birgerCTRL User Manual

Manual Rev 0.8
3/14/26

Birger Engineering, Inc.
42 Chauncy St Suite 1A
Boston, MA 02111
birger.com

1 Table of Contents

2 Quick Reference Tables

Table 2.1. Legacy-Type Command Quick Reference

Command	***Arguments***	***Description***
bv		Print the bootloader version.
da		Print aperture information.
de		Dump EEPROM.
ds		Print distance stops.
dz		Print the zoom range.
eh	pos,checksum	Set absolute lens focus position (0…0x3FFF).
ex		Exit to the bootloader.
fa	position	Move focus to absolute position.
fd		Print focus distance range.
ff	position	Fast focus.
fp		Print the raw focus positions.
gs		Echo current device and lens statuses.
hv		Print the hardware version.
id		Print basic lens identification (zoom and f-number).
im	{0, 1}	Ignore manual focus
in		Initialize the aperture motor. Aperture will fully open.
is	{0, 1}	Turn image stabilization off/on.
la	[0,1,2,3,4]	Learn the focus range.
lc		Print cached lens status
ll		Library loaded check.
ln		Lens name.
lp		Lens presence.
ls		Query lens for status immediately and print.
lv		Print the library version string.
ma	stop	Move aperture to absolute position.
mc		Move aperture to the fully stopped down limit.
mf	delta_value	Move focus incremental.
mi		Move focus to the infinity stop.
mn	num_stops	Move aperture incremental.
mo		Move aperture to completely open.
mz		Move focus to the zero stop.
pa		Print the aperture position.
pf		Print the focus position.
pl	[0, 1]	Lens power
pr	digits	Set floating point precision for reporting.
ps	[0,1,2]	Print aperture stops
pz		Print zoom position

qp		Query port ID.
rm	verbose[,new]	Set response modes.
se	byte,val	Temporarily set non-volatile (EEPROM) byte.
sf	count	Set the focus counter.
sm	mode_flags	Set special modes.
sn		Print the device serial number.
sr	{0, 1}	Set spontaneous responses off/on.
vs		Print the short version string.
wc		Write console settings to EEPROM.
we	serial_number	Write non-volatile parameters to EEPROM.
za	position	Move zoom absolute.
zn	delta_value	Move zoom incremental.
zt		Move zoom to telephoto position.
zw		Move zoom to wide position.

Table 2.2. New Command Quick Reference

Command	***Arguments***	***Description***
announce	{0,1}	Send Ethernet broadcast packet with device name.
boot	number	Boot an application loaded in the device.
cmds		List all commands in short form.
comp	{0,1}	Set/clear EF compatibility mode.
creset		Reset the console.
cuser		Create a user login.
dhcp		DHCP client commands.
dnsc		DNS client commands
echo	"string" [port]	Echo characters
help		List all commands in long form.
lctrl		Control login options.
list		List the applications loaded in the device.
mkapp	number	Mark the application to be run at boot.
netinfo		Print network information.
qapp		Query the number of the running application.
qsec		Query port security.
quit		Quit the command processor
reset		Reset the device.
route	port	Route output to another port
sdr		Set/print detailed response modes
setdns		Set DNS address.
setbaud		Set the baud rate for the serial port.
setip		Set IP address and mask.

setgw		Set gateway address.
sock		List all open TCP/IP sockets.
spt		Set TCP/IP port/protocol configurations.
stack		Control the TCP/IP stack.
temp		Print the chip temperature.
time		Enable/disable legacy command timestamps.
usbstat		Print USB port status

Table 2.3. Bootloader Command Quick Reference

Command	***Arguments***	***Description***
bl		Boot latest application.
boot	app_index	Boot a particular application.
help		Print command help.
hv		Print the hardware version.
list		Print a list of the applications loaded in memory.
mkapp	app_index	Mark an application for boot.
reset		Reset the device.
rm	verbose[,new]	Set response modes.
sn		Print the unit serial number.
vs		Print the short version string.

3 General Notes on Using the Application

This section describes the differences between legacy-type commands and new commands, the differences in command behavior, and the differences between the bootloader and application(s). It is also the reference for the basic behaviors of the Lens Controller application.

3.1 Bootloader vs. Application

There is a secure bootloader in the device that can communicate over the serial or USB interfaces. The bootloader is capable of basic functionality and encrypted firmware updates. You should not normally need to use the bootloader functionality.

The bootloader transfers execution to the application for normal operation, and the application handles normal command processing. Commands can be blocking or non-blocking (they return immediately while lens processing occurs “in the background”). Firmware upgrading can be done through the bootloader or application.

The current bootloader no longer behaves exactly as the bootloader in the original Birger lens controller, it contains a very limited command set and those commands require a space between the command and its arguments. If you have developed an application for the original lens controller that required using the bootloader, contact Birger engineering for instructions on how to migrate to the new platform.

3.2 Command Interfaces

The window into the device is provided by the command console. This is how information and control is sent and received from the device. In this manual, it will be often be referred to in shorthand as the ‘console.’

A command console is provided over several interfaces (or ports to match the language of the Canon EF-232 product). These interfaces are provided over directly hardwired ports such as USB and serial, and also over virtual dynamic ports such as can be provided via TCP/IP protocols over Ethernet.

All command interfaces share the same process so that commands from different interfaces can be coordinated, however this means that one interface will hold off the others when a command is being executed unless it is a command that can be transferred to background processing.

3.2.1 Serial

Command interfaces provided by device serial port(s) are connectionless and always available, i.e. there is no protocol negotiation phase to begin communication with the device. The serial interface(s) operate with standard asynchronous signaling with 8 data bits, 1 stop bit, no parity, and configurable baud rate.

3.2.2 USB

There may be a single USB interface implementing a Hi-Speed USB 2.0 device. Command console interfaces are implemented as standard CDC virtual serial ports. If the USB interface is available then the device may implement one or more virtual serial ports in this manner.

3.2.3 TCP/IP

There is a single wired network interface implementing Ethernet+TCP/IP. Connecting to this interface requires a Birger Ethernet adapter module. Virtual ports are created over this interface using Telnet over TCP. The Telnet port number is configurable. Birger has extended this interface to operate with standard TLS 1.2 security. Security can be turned on or off for all Telnet connections (they must be all set the same).

Connecting devices must initiate the TLS connection protocol without prompting, e.g. there is no equivalent of a STARTTLS message. The device ships with security enabled and with a default certificate and key for secure connections; this certificate and key can be replaced with the user's own certificate and key.

3.2.4 CAN

There is a single wired network interface implementing CAN available in the hardware. Connecting to this interface requires a Birger CAN adapter module. Software support is not currently enabled for the CAN interface.

3.3 Legacy-Type Commands vs. New

The application provides two different command paradigms. These are legacy-type commands and new commands. Legacy-type commands are two-character commands that duplicate the original Birger Canon EF-232 command paradigm. These commands do not need a space between the command and the arguments. There is a units extension that can be added to certain legacy-type commands. This unit extension is part of the command itself and cannot be separated from the command by a space.

New commands are longer than two characters and require a space between the command and arguments. New commands can offer a richer command interface than legacy-type commands.

For both types of commands, arguments can be separated by a comma or space.

3.4 Legacy Aperture Positioning

Aperture positioning is generally measured in ¼ stop increments. However, the input to the aperture positioning commands is an integer and the reported aperture positions are f-number times ten or a corresponding integer number of ¼ stops. For example, if f28 is reported for an aperture position, then it is indicating that the actual f-number is 2.8. If this is the wide-open position then it is considered the zero position, and the input to the absolute aperture position commands would be zero. If aperture measurement is not reported as ¼ stops then the exception will be noted where applicable.

A value of one would indicate ¼ stop from the full-open position (2.8) towards the closed position, which would be approximately 3.1. If the aperture position is then reported in this state, the number one would be reported as the position in number of ¼ stops from the wide-open (zero) position, and the f-number would be reported as f30. Note that due to round-off error in certain cases, the f-number reported may be close to, but not exactly the mathematical f-number.

3.5 Legacy Focus Positioning

Focus positioning for legacy commands is performed with respect to the encoder positions directly reported by the connected lens. Each lens will have a different encoder count for its focusable range, and the encoder count for the zero position will almost never be zero, in fact it will usually be random, changing with each power cycle of the lens. However, the focus counter command 'sf' can be used to make any number the reference for the zero position. The lens can be focused anywhere from zero to infinity by using commands to learn the full encoder count range and then applying those values to the positioning commands. For example, a move to zero and then a set focus counter to zero will establish the value of zero as the position for the minimum focus distance of the lens. A move to infinity will then report the full number of encoder counts that it took to get there.

3.6 Mapped Absolute Focus Positioning

The application can perform absolute positioning on a fixed scale. This fixed scale is a 14-bit range from 0 to 16383 (hex 3FFF) when in EF compatibility mode. The range of the lens from the zero stop to the infinity stop is mapped into this numerical scale. The 14-bit range is generally much larger than the encoder range returned by the lens, and therefore a small change in the mapped position often will not result in a change to the focus position. If compatibility mode is turned off, this scale can range to 65535 (hex FFFF) for lenses with extended encoder ranges. The value of this scale can be learned with a command.

The mapped focusing feature is highly recommended when using a control surface to alter focusing. It allows use of a scale that is always fixed between the zero and infinity positions. However, if the control surface is not capable of sending small enough changes in the mapped position, then jumpy focusing behavior can occur. Ideally the control surface should also have a 14-bit range, but a smaller range is okay as long as it is several times larger than the lens' encoder range. The control surface should have at least 12-bit resolution to meet this requirement for all lenses tested to date. A smaller resolution will still work, but focusing will not be as smooth or precise.

3.7 Focus Drift

Many lenses will experience drift of the focus mechanism. This results in a change of the encoder position associated with a particular stop. Generally the lens will lose or gain a few encoder counts when hitting a stop. The lens may also lose positioning just during normal focusing, and it can be made worse if the focus move is small. The better-constructed, more-expensive lenses will drift far less than the cheaper lenses. Drift must be accounted for by relearning the encoder positions of the zero and infinity stops.

3.8 Canon EF-232 Library Command Compatibility

The library as shipped emulates the original Canon EF-232 product when the command interface was configured in verbose, new-language mode and when using legacy-type commands. Commands that are marked as 'New (Legacy-Type)' in this manual will still conform to this behavior. This provides maximum compatibility with any user-written applications targeting the original Birger device.

3.8.1 Blocking vs. Non-Blocking Commands

The application duplicates the command behavior of the Canon EF-232 product. Blocking commands will still behave in the same way, and will block processing on the command interface. In other words, they needed to finish before any other command can be sent. There are also commands that do not block, such as the servo focus commands. These commands return immediately and the application handles the necessary lens processing in the background.

If you intend to have multiple devices talking to the lens controller over several interfaces, then lens commands and status queries should use commands that operate in the background if they exist. Otherwise, a device on one interface will hold off the other devices from getting their communications through in a timely manner. The lens controller application attempts to coordinate the lens processing so no conflicts occur from the lens' point-of-view, but from your application's point-of-view, you are responsible for coordination among different applications and different interfaces.

3.8.2 Terse vs. Verbose Reporting Modes

The device can be configured to report in two different modes in addition to legacy/new protocols using the response modes ('rm') command. In terse mode, characters are not echoed back from the device, and all commands return a single number, zero for no error or a non-zero error value. Note that even commands that would normally return a status string will not do so in terse mode as long as they are marked as legacy-type commands. New commands will always echo their output if they have status to print.

In verbose mode, all characters input to the device are echoed back out, and commands will print their full status strings as usual. If set to both verbose and new protocol modes, then a question mark will be emitted by the device for unknown commands. Verbose mode is most useful when manually controlling the device through a terminal program. See section 3.8 for a more detailed breakdown of command console configurations.

3.9 Command Console Configurations

The command console behavior can be configured on a per-interface basis. The following configurations can be set:

Table 3.9.1. Command Console Settings

Setting	***Description***
Character Echo	When set to true, all characters received by the device are echoed back out to the sender as each character is received.
Line-Feeds	When set to true, carriage-returns will be followed by line-feeds, otherwise the device will only emit carriage-returns as line ends.
Suppress Prompt	When set to true, no prompt character (>) will be emitted when the device is ready to receive a command.
New Language	When set to true, puts the command interface in the new-language mode provided by the original Canon EF-232 product with libraries version 17 or later, otherwise the interface behaves as the original product with libraries earlier than version 17.
Verbose Mode	When set to true, puts the command interface in the verbose mode provided by the original Canon EF-232 product with libraries version 17 or later, otherwise the interface behaves as the original product with libraries earlier than version 17.
Compatibility Mode	When set to true, the interface and command output will emulate the original Canon EF-232 product. This is likely required for compatibility with any pre-existing user applications targeting the original product. When set to false, certain commands will emit more information than their original counterparts.
VT100 Mode	When set to true, the device will emit VT100 control strings for richer text and terminal control.

3.10 Status Strings

There are a standard set of status strings that can be queried for or spontaneously emitted from the device. These status strings are of the format <designator><option><:> where <designator> is a character that has been chosen so that it should not appear in normal output, making it easier for user application software to parse the device output.

The spontaneous emissions are off by default but can be turned on with the command 'sr'. Several status strings are compatible with the original Canon EF-232 product; new status strings will have different designators applied so that compatibility is maintained.

3.10.1 Focus Position Status

Format: %:xxxx

Where xxxx is a 4-digit hexadecimal number indicating the current focus position of the lens in the mapped range (0...3FFF).

3.10.2 Aperture Position Status

Format: &:xx

Where xx is a 2-digit hexadecimal number indicating the current aperture position in 1/8 stops. Note that if the aperture position is unknown, for example if a lens has been connected but not initialized, then the status string will be:

&:?

3.10.3 Focal Length and Aperture Range Status

Format: @:<focal_length>mm,f<min_f>,<num_stops>,f<max_f>

Where <focal_length> is a decimal number for the current focal length in millimeters, <min_f> is a decimal number for the current minimum f-number times ten, <num_stops> is a decimal number for the number of ¼ stops available, and <max_f> is a decimal number for the current maximum f-number times ten.

3.10.4 Miscellaneous Status Flags

Format: #:xxxx

Where xxxx is a 4-digit hexadecimal number indicating the current value of the flags. The flags are given in the following table.

Table 3.10.4.1 Status Flags

Bit(s)	***Description***
0	Reserved.
1	Reserved.
2	Reserved.
3	Reserved.
4	Aperture initialized; 0 = not initialized, 1 = initialized.
5	Manual focus; 0 = lens is in autofocus, 1 = lens in in manual focus.
6	Reserved.
7	Focus stops learned; 0 = not learned, 1 = learned.
8-15	Lens module-specific flags, see individual sections for different lens families.

3.10.5 Lens Connection Messages

When in new-protocol mode, a spontaneous string will be emitted when a lens is connected or disconnected from the unit. These strings are:

Lens Connected.

and

Lens Disconnected.

3.11 Errors in Command Processing

If a command fails it will report a numerical error code. In terse mode the number is returned by itself, while in verbose mode and new-protocol mode the return string will be:

ERRx

Where x is the numerical error code. The possible error codes that can be returned are given in the following table.

Table 3.11.1. Error Codes

Error Code	***Description***
0	No error.
1	Unrecognized or bad command.
2	Lens is in manual focus mode.
3	No lens connected.
4	Lens distance stop error.
5	Aperture not initialized.
6	Invalid baud rate specified.
7	Reserved.
8	Reserved.
9	A bad parameter was supplied to the command.
10	XModem timeout.
11	XModem error.
12	XModem unlock code incorrect.
13	Lens is in manual zoom mode.
14	Invalid port.
15	License unlock failure.
16	Invalid license file.
17	Invalid application file.
18	Reserved.
19	Lens is in manual aperture mode.
20	Not used.
21	Application not ready for lens communications.
22	Application not ready for commands.
23	Command not licensed.
24	Invalid focus range in memory. Try relearning the range.
25	Distance stops are not supported by the lens.
26	Fast learn algorithm distance stop error.
27	Distance stops not learned.
28	EEPROM is busy.
29	Not enough parameters supplied to command.

30-49	Reserved.
50	Miscellaneous EEPROM error.
51	Function not supported.
52	Zoom failed to reach target.
53	Units not supported.
54	Mixed units not allowed.
55	FLASH program error
56	Not used.
57	Bad port specified.
58	Reserved.
59	Reserved.
60	Baud rate error greater than one percent.

4 How to Read the Command References

Each command page is organized as follows:

Applicable Versions

Command Name (command)

Command Type	**: Legacy / New (Legacy-Type) / New**
Aliases	**: Command aliases if available**
Syntax	**: What is entered as the complete command**
Returns	**: The information or exact string returned**

Description

A description of how the command is used.

Examples:

One or more examples of how the command is used. The command syntax is shown as normal text, information returned from the device is shown in italic text. The command confirmation for legacy behavior ('OK') is not shown.

5 Legacy-Type Command Reference

The following pages contain descriptions of the legacy and legacy-type commands that are available from within the lens controller application. Legacy commands are two-character commands that existed in the original Birger lens controller device. Legacy-type commands are new commands provided by this platform that follow the same paradigm as the original legacy commands. The commands in this section do not need a space between the command and its arguments.

v1.30.0+

5.1 Bootloader Version (bv)

Command Type	**: Legacy**
Syntax	**: bv**
Returns	**: Bootloader version number**

Description

This command will print the version of the bootloader code from within the application. Therefore exiting to the bootloader is not needed to read the revision information.

Example:

```
bv <CR>
```

```
50
```

v1.30.0+

5.2 Define Aperture (da)

Command Type	**: Legacy**
Aliases	**: ar**
Syntax	**: da[e, f, t]**
Returns	**: f<min>,<num_stops>,f<max>**
Returns	**: <units><min>,<units><max>**

Description

This command will print the the aperture range of the connected lens. The range is printed in three parts as follows:

fmin**,**num_stops**,f**max

The item printed is the letter 'f' followed by the current minimum f-number. This is followed by a comma and the total number of stops in ¼ stops that can be commanded to the lens. This is then followed by another comma, the letter 'f', and the current maximum f-number. If the command is suffixed with a units modifier then the output will be in the desired units without the number of stops.

Example:

```
da <CR>
```

```
f43,22,f293
```

Example:

```
daf <CR>
```

```
f4.3,f29.3
```

v1.30.0+

5.3 Dump EEPROM (de)

Command Type	**: Legacy**
Syntax	**: de**
Returns	**: EEPROM parameters as hexadecimal bytes**

Description

This command will dump a hexadecimal stream of bytes that corresponds to the non-volatile parameters stored in EEPROM. The commonly configured parameters are almost always available through other commands, so this command is used to gain access to the remaining parameters. These parameters are often low-level timing configurations. Contact Birger Engineering if you think your programming needs might require modification to these parameters.

v1.30.0+

5.4 Distance Stops (ds)

Command Type : Legacy
Syntax : ds

Returns : $:<far>:<near>:<encoder>

Description

This command will execute a move to the zero stop of the lens. It will then advance the lens through its focus range and print the distance stops as reported by the lens. The distance stops take the form:

$:<far>:<near>:<encoder>

Where <far> is a 4-digit hexadecimal number representing the far distance of the stop in centimeters, <near> is another 4-digit hexadecimal number representing the near distance of the stop in centimeters, and <encoder> is a 4-digit hexadecimal number of the raw encoder count of the lens. The command finishes with a move to the focus motor infinity stop to read its position followed by a return to the original focus position. Issuing this command while it is already in progress will cancel further distance reporting. Other lens control commands will not work or should not be issued while this command is in progress.

The infinity position is reported as ffff hexadecimal. The infinity position reported as a distance stop may not actually correspond with the infinity focus position resulting from the focus motor's infinity stop. These are often two different stops in the lens construction. Not all lenses are capable of reporting distance stops.

Example:

```
ds <CR>

$:0021:0020:fe01
$:0023:0021:feb4
$:0025:0023:ff23
$:0026:0025:ffb1
$:0029:0026:0066
$:0035:0029:0216
$:005f:0035:0436
$:008b:005f:04f6
$:017a:008b:05e8
$:045d:017a:062f
$:ffff:045d:0655
$:ffff:ffff:06bf
$:DONE
```

v1.30.0+

5.5 Define Zoom (dz)

Command Type : Legacy
Syntax : dz

Returns : <min_focal_length>mm,<max_focal_length>mm

Description

This command will print the zoom range of the lens. The first number printed is the minimum focal length in millimeters followed by 'mm'. This is followed by a comma, the maximum focal length of the lens, and the letters 'mm'. Prime lenses will print the same value for minimum and maximum focal length.

Example:

```
dz <CR>
```

```
18mm,55mm
```

```
dz <CR>
```

```
50mm,50mm
```

v1.30.0+

5.6 Servo Focus with Checksum (eh)

Command Type	**: Legacy**
Syntax	**: eh <absolute_position>,<checksum>**
Returns	**: Nothing**

Description

This command is used to set the focus position of the lens on an absolute 14-bit scale from 0 to 3FFF hexadecimal. The command requires a checksum that is the bit-wise exclusive-OR of the four 4-bit digits of the position. This command executes through the servo focus routine and returns immediately.

All four digits of the position must be specified, including leading zeros. If the correct number of digits is not sent or the checksum is incorrect, then the command will be ignored.

Examples:

```
eh0000,0 <CR>

eh1000,1 <CR>

eh3fff,c <CR>
```

v1.30.0+

5.7 Exit to Bootloader (ex)

Command Type : Legacy
Syntax : ex

Returns : In bootloader!

Description

This command exits application processing and hands control to the bootloader. The application can be recovered by re-programming through the bootloader if needed, however, the bootloader should not be needed for any normal operation..

The return response from this command depends on which protocol mode is active, legacy vs. new, and terse vs. verbose, but the string 'In bootloader!' will always be returned.

Examples:

```
ex <CR>

BEI Boot Executive, v50
(c)2023 Birger Engineering, Inc.
In bootloader!
```

v1.30.0+

5.8 Focus Absolute (fa)

Command Type : **Legacy**
Syntax : **fa[c, f, i, m] <position>**

Returns : **DONE<rposition>,<flag>**
Returns : **DONE<rposition><units>,<flag>**

Description

This command makes the focus mechanism move to the specified focus count position, specified as an absolute position given in <position>. This position is relative to the reference point specified by the 'sf' command. If an input value would move the focus out of the legal range the value is rounded to the boundary (i.e. min/max count). <rposition> is the actual position that the count has been moved to, <flag> is 1 if the lens reports having hit a stop, and 0 if it hasn't. Note that some lenses do not return a 1 until the second time the stop is hit.

This command can also take distance units (inches, feet, centimeters, meters) if focusing by distance is available.

Examples:

```
fa100 <CR>
```

DONE100,0

```
fa2000 <CR>
```

DONE1759,1

```
fam20 <CR>
```

DONE20m,1

v1.30.0+

5.9 Focus Distance (fd)

Command Type	**: Legacy**
Syntax	**: fd[c, f, i, m]**
Returns	**: <near>cm,<far>cm**
Returns	**: <near><units>,<far><units>**

Description

The number is returned as two distances, <near> the near distance, and <far> the far distance, both in centimeters, where 65535 represents infinity. Not all lenses return distance information. This information corresponds to what the lens reports as the stops that bracket the current focus position, not as the near and far focal distances as determined by the actual optics involved, which are calculations based on the lens focal length, aperture, focusing distance, and circle-of-confusion. This command can take a units modifier to change the reporting to inches, feet, centimeters, or meters.

Examples:

```
fd <CR>

45cm,48cm

fd <CR>

65535cm,65535cm

fdm <CR>

655.35m,655.35m
```

v1.30.0+

5.10 Fast Focus (ff)

Command Type	**: Legacy**
Syntax	**: ff <position>**
Returns	**: DONE**

Description

This command has been updated to use the servo focus routine. The position supplied uses the legacy focus positioning, which are encoder counts as reported by the lens with the applied offset set through the focus counter command 'sf'. This command is retained for legacy compatibility but is not recommended for new designs.

Example:

```
ff1000 <CR>
```

DONE

5.11 Focus Positions (fp)

Command Type : Legacy
Aliases : fr
Syntax : fp[c, i, f, m]

Returns : fmin:<fmin> fmax:<fmax> current:<current_pos>
Returns : fmin:<fmin><units> [fmax:<fmax><units>] current:<current_pos><units>

Description

This command will print the focus positions in raw encoder counts as reported by the lens. The positions are reported as signed decimals, and as they are raw encoder counts, do not have the focus counter offset applied. <fmin> is the zero position of the lens and is updated every time the lens is commanded to the zero stop with the 'mz' command. <fmax> is the infinity position of the lens and is updated every time the lens is commanded to the infinity position with the 'mi' command. Both positions are updated if the learn focus range ('la') command is issued. <current_pos> is the current focus position of the lens. If a lens supports it then this command can take a distance units modifier for reporting (inches, feet, centimeters, meters), in which case the maximum distance is assumed to be infinity unless an fmax is printed.

Example:

```
fp <CR>

fmin:-1303  fmax:1228  current:696

fpm <CR>

fmin:1.2m, current:5.8m
```

v1.30.0+

5.12 Get Statuses (gs)

Command Type : **Legacy**
Syntax : **gs[a]**

Returns : **Status strings**

Description

This command prints complete status information in the form of the new status strings. The status strings are described in Section 3.9 of this manual. This command can take a modifier character to print extended status for the new platform, in which case aperture position is printed by exposure value and focus position is printed by distance if available.

Example:

```
gs <CR>

%:1948
&:0a
@:100mm,f28,28,f320
#:0010
```

v1.30.0+

5.13 Hardware Version (hv)

Command Type : Legacy
Syntax : hv

Returns : Device hardware version

Description

This command will print the hardware version of the device.

Example:

```
hv <CR>
```

```
5
```

v1.30.0+

5.14 Lens Identification (id)

Command Type : Legacy
Syntax : id

Returns : <current_focal_length>mm,f<current_min_f_number>

Description

This command prints the basic lens identity. For a zoom lens it will print the current focal length, otherwise it prints the fixed focal length of the lens. This is followed by a comma and the letter 'f' followed by the current minimum f-number that the lens is capable of multiplied by ten.

Examples:

```
id <CR>
```

38mm,f47

```
id <CR>
```

100mm,f28

v1.30.0+

5.15 Ignore Manual Focus (im)

Command Type	**: Legacy**
Syntax	**: im {0, 1}**
Returns	**: OK**

Description

This command is for internal use and is subject to deprecation.

v1.30.0+

5.16 Initialize Aperture (in)

Command Type	**: Legacy**
Syntax	**: in**
Returns	**: DONE**

Description

This command initializes the aperture motor and position. This opens up the aperture to its minimum f-number (maximum opening). This position is then set as the zero position for all subsequent aperture movement commands. This is required for most lenses before any of the aperture movement commands can be used, as driving the aperture motor too far without knowing its position could damage the aperture mechanism.

There are some lenses for which the aperture range will be set to a working default so that this command does not need to be used, however using this command may learn a more open aperture than the default. If necessary, the library prevents you from issuing aperture movement commands unless the position has first been initialized with this command. Optionally, the device can be configured to automatically initialize the aperture motor when a lens is attached.

Example:

```
in <CR>
```

DONE

v1.30.0+

5.17 Image Stabilization (is)

Command Type : Legacy
Syntax : is {0, 1}

Returns : DONE

Description

Turn image stabilization on or off. Does nothing if the lens does not support image stabilization.

Examples:

```
is0 <CR>
```

DONE

```
is1 <CR>
```

DONE

v1.30.0+

5.18 Learn Absolute Focus Range / Detect Lens (la)

Command Type	**: Legacy**
Syntax	**: la [0, 1, 2, 3, 4]**
Returns	**: DONE:LA**

Description

This command learns the focus range of the lens for the servo focus routine and also for any legacy commands, in a single step. No other lens movement commands can be issued while this command is in progress. This command does not block, so other commands that don't require lens communications can be issued while the lens is focusing. A confirmation will be emitted by the device when the lens has completed focusing. The servo focusing and mapped focus range will not function unless the lens focus range is learned.

This command can also be supplied with an argument that starts the lens detection process at different stages.

Option:	0	Same as LA without arguments – learn the lens' focal range.
	1	Resynchronize communications with the lens, and learn range.
	2	Re-detect and resynchronize with the lens and learn range.
	3	Resynchronize with the lens as (1), preserving the lens' current mechanical focus position.
	4	Same as (2) but power the lens down and back up first; equivalent to sending pl0 and la2.

Examples:

```
la <CR>
```

```
DONE:LA
```

```
la1 <CR>
```

```
DONE:LA
```

```
la2 <CR>
```

```
DONE:LA
```

v1.30.0+

5.19 Cached Extended Lens Status (lc)

Command Type : **Legacy**
Syntax : **lc**

Returns : **@:<focal_length>mm,f<min_f>,<num_stops>,f<max_f>**

Description

This command will print the extended lens status from cached information. The status string is of the form outlined in Section 3.9. Cached information is updated periodically by the library if background querying is turned on (see command 'sm'). If multiple devices are interfacing to the lens controller over multiple serial ports, then this command should be used to retrieve status, as it will avoid conflicts.

Example:

```
lc <CR>

@:38mm,f47,21,f293
```

v1.30.0+

5.20 Library Loaded (ll)

Command Type : Legacy
Syntax : ll

Returns : OK

Description

This command simply prints 'OK'. It can be used as a test to see if the device is executing application code or bootloader code. The command does not exist in the bootloader, and therefore will not work unless application code is being executed.

Example:

```
ll <CR>
```

OK

v1.30.0+

5.21 Lens Name (ln)

Command Type : **New (Legacy-Type)**
Syntax : **ln**

Returns : **The name of the lens**

Description

This command will print the name of the connected lens.

Examples:

```
ln <CR>
```

```
Canon EF-S 17-55mm f/2.8 IS USM
```

v1.30.0+

5.22 Lens Presence (lp)

Command Type : **Legacy**
Syntax : **lp**

Returns : **0 or 1**

Description

This command will indicate if a lens is currently connected or not. It prints the number '0' when no lens is connected and '1' when a lens is connected.

Examples:

```
lp <CR>

0

lp <CR>

1
```

v1.30.0+

5.23 Immediate Extended Lens Status (ls)

Command Type	**: Legacy**
Syntax	**: ls**
Returns	**: @:<focal_length>mm,f<min_f>,<num_stops>,f<max_f>**

Description

This command will print the extended lens status by immediately querying the lens. The status string is of the form outlined in Section 3.9. This command should be used if background querying is turned off (see command 'sm').

Example:

```
ls <CR>
```

```
@:38mm,f47,21,f293
```

v1.30.0+

5.24 Library Version (lv)

Command Type	**: Legacy**
Syntax	**: lv**
Returns	**: Full application identity string**

Description

This command prints the full application identification string. In compatibility mode the application version will be taken from the three version components to create a single number and letter version.

Compatibility Mode Example:

```
lv <CR>
```

`Canon EF-232 Library v30a`

Normal Mode Example:

```
lv <CR>
```

`Birger Lens Controller V2 v1.30.0`

v1.30.0+

5.25 Move Aperture Absolute (ma)

Command Type	**: Legacy**
Aliases	**: aa**
Syntax	**: ma[e, f, t] <pos>**
Returns	**: DONE<rpos>,f<f_number>**
Returns	**: DONE<approx_rpos>,<units><position>**

Description

This command moves the aperture mechanism to the specified encoder position, specified in ¼ stops from the fully-open position, indicated by the user in <pos>. If an input value would move the aperture out of the legal range the value is set to the boundary (i.e. min/max aperture). <rpos> is the actual position the aperture moved to in ¼ stops from the fully-open position, and <f_number> is the absolute position given as the lens f-number times ten.

This command can take a units modifier to specify the aperture position in terms of exposure value, f/number, or T-number. In this case the output contains the nearest legacy indexing value to the desired aperture position along with the actual aperture position that was achieved.

Examples:

```
ma0 <CR>
```

DONE0,f43

```
ma5 <CR>
```

DONE5,f67

```
maf6.7 <CR>
```

DONE5,f6.7

v1.30.0+

5.26 Move Aperture to Closed Position (mc)

Command Type	**: Legacy**
Aliases	**: ac**
Syntax	**: mc[e, f, t]**
Returns	**: DONE<signed_num_steps>,f<f_number>**
Returns	**: DONE<units><signed_num_steps>,<units><position>**

Description

This command will move the aperture to the fully closed position. The number of steps in ¼ stops that the aperture actually moved is given in <signed_num_steps>. The final position is given in <f_number> as the lens f-number times ten.

This command can take a units modifier to specify the aperture position in terms of exposure value, f/number, or T-number. In this case the output contains the amount moved and final position.

Example:

```
mc <CR>
```

DONE14,f216

```
mcf <CR>
```

DONEf4.0,f22.6

v1.30.0+

5.27 Move Focus Incremental (mf)

Command Type	**: Legacy**
Aliases	**: fn**
Syntax	**: mf[c, f, i, m] <signed_value>**
Returns	**: DONE<signed_num_counts>,<flag>**
Returns	**: DONE<signed_value><units>,<flag>**

Description

This command makes the focus mechanism move incrementally the number of counts specified in <signed_counts>. If the number specified is positive then the focus moves towards the infinity position. If the number is negative then the focus moves towards the zero position. If the number of counts specified is out of range, the mechanism will move as far as it can, <signed_num_counts> is the actual number of counts moved, and <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that some lenses do not return a 1 until the second time the stop is hit.

This command can take a distance unit modifier if the lens supports focusing by distance.

Examples:

```
mf100 <CR>

DONE100,0

mf-50 <CR>

DONE-50,0

mfm-3.4 <CR>

DONE-3.4m,0
```

v1.30.0+

5.28 Move Focus to Infinity (mi)

Command Type	**: Legacy**
Aliases	**: fi**
Syntax	**: mi[c, f, i, m]**
Returns	**: DONE<signed_num_counts>,<flag>**
Returns	**: DONE<signed_value><units>,<flag>**
Returns	**: DONE<inf>,<flag>**

Description

This command moves the lens focus to the infinity position. The actual number of counts moved as reported by the lens' encoder is given in <signed_num_counts>. <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that some lenses do not return a 1 until the second time the stop is hit.

This command can take a units modifier for printing if the lens supports distance reporting. If the lens is focused at infinity then 'inf' will be printed as the distance only if a units modifier were specified, to maintain backward compatibility.

Example:

```
mi <CR>

DONE246,1

mim <CR>

DONE320m,1

mim <CR>

DONEinf,1
```

v1.30.0+

5.29 Move Aperture Incremental (mn)

Command Type	**: Legacy**
Aliases	**: ai**
Syntax	**: mn[e, f, t] <signed_values>**
Returns	**: DONE<num_signed_steps>,f<f_number>**
Returns	**: DONE<units><signed_value>,<units><position>**

Description

This command moves aperture mechanism the specified number of steps, in the specified direction, in ¼ stops given by <signed_steps>. A positive step is a move towards the fully closed position and a negative step is a move towards the fully opened position. If an input value would move the aperture out of the legal range the value is clipped to the boundary (i.e. min/max aperture). <num_signed_steps> is the actual number of steps the aperture was moved, and <f_number> is the absolute position of the aperture after executing the move, in the lens f-number times ten.

This command can take a units modifier to move the desired amount in the specified number system. The actual amount moved is printed along with the final position.

Examples:

```
mn2 <CR>

DONE2,f54

mn-1 <CR>

DONE-1,f49

mnf0.25 <CR>

DONEf0.25,f4.8
```

v1.30.0+

5.30 Move Aperture to Opened Position (mo)

Command Type	**: Legacy**
Aliases	**: ao**
Syntax	**: mo[e, f, t]**
Returns	**: DONE<signed_num_steps>,f<f_number>**
Returns	**: DONE<units><signed_value>,<units><position>**

Description

This command will move the aperture to the fully open position. The number of steps in ¼ stops that the aperture actually moved is given in <signed_num_steps>. The final position is given in <f_number> as the lens f-number times ten.

This command can take a units modifier to specify the aperture position in terms of exposure value, f/number, or T-number. In this case the output contains the amount moved and final position.

Example:

```
mo <CR>

DONE-21,f35

mof <CR>

DONEf-3.4,f4.0
```

v1.30.0+

5.31 Move Focus to Zero (mz)

Command Type	**: Legacy**
Aliases	**: fz**
Syntax	**: mz[c, f, i, m]**
Returns	**: DONE<signed_num_counts>,<flag>**
Returns	**: DONE<signed_value><units>,<flag>**

Description

This command moves the lens focus to the minimum object distance position. The actual number of counts moved as reported by the lens' encoder is given in <signed_num_counts>. <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that some lenses do not return a 1 until the second time the stop is hit.

This command can take a units modifier for printing if the lens supports distance reporting.

Example:

```
mz <CR>

DONE-1246,1

mzm <CR>

DONE-12.3m,1
```

v1.30.0+

5.32 Print Aperture Position (pa)

Command Type	**: Legacy**
Aliases	**: ap**
Syntax	**: pa[e, f, t]**
Returns	**: <pos>,f<f_number>**
Returns	**: <pos>,<units><position>**

Description

This command prints the current position of the aperture. <pos> is the absolute position given in ¼ stops from the fully-open position and <f_number> is the absolute position given as the lens f-number times ten.

This command can take a units modifier to specify the reporting in terms of exposure value, f/number, or T-number. In this case the output contains the nearest legacy indexing value to the desired aperture position along with the actual aperture position.

Example:

```
pa <CR>
```

5,f52

```
paf <CR>
```

5,f4.7

v1.30.0+

5.33 Print Focus Position (pf)

Command Type **: Legacy**
Syntax **: pf[c, f, i, m]**

Returns **: Signed focus position in encoder counts.**
Returns **: Focus position in desired units.**

Description

This command prints the current position of the focus mechanism given in encoder counts with the user's desired focus counter offset applied.

This command can take a units modifier for distance reporting if the lens supports it.

Examples:

```
pf <CR>

-1247

pf <CR>

743

pfm <CR>

3.2m
```

V1.30.0+

5.34 Lens Power (pl)

Command Type	**: Legacy**
Syntax	**: pl [0, 1]**
Returns	**: Current setting**

Description

Report or control the lens power state. If sent with no arguments, reports the current state, 0 = powered down, 1 = powered up. The lens will be powered down gracefully according to the manufacturer's recommended procedure, which may involve parking the motors (aperture, focus, zoom) in default positions.

Option:	0	Park and gracefully power down the lens.
	1	Power up and re-detect the lens.

Examples:

```
pl <CR>
```

1

```
pl1 <CR>
```

1

```
pl0 <CR>
```

0

5.35 Set Floating Point Precision (pr)

Command Type : New (Legacy-Type)
Syntax : pr <digits>

Returns : OK

Description

Set the number of digits of floating point precision for printing. The internal floating point precision used for calculations is not affected by this setting.

Examples:

```
pr1 <CR>

OK

maf4 <CR>

DONE0,f4.0

pr2 <CR>

OK

maf4 <CR>

DONE0,f4.04

pr3 <CR>

OK

maf4 <CR>

DONE0,f4.035
```

5.36 Print Aperture Stops (ps)

Command Type : **New (Legacy-Type)**
Syntax : **ps [0, 1, 2]**

Returns : **!<type>: Followed by list of aperture stops.**

Description

Prints the list of aperture stops. These stops may not necessarily line up with the legacy aperture positions that are arranged on ¼ stops, and they may be emulated for lenses, including for lenses with continuous apertures. This gives a discrete list of stops that can be accessed with the 'ma' command plus the zero-based stop index.

Option:	None or 0	Print stops as exposure values (floating-point).
	1	Print stops as f-numbers (floating-point).
	2	Print stops as exposure values times 1000 (integer).

Examples:

```
ps

!0:4.25,4.50,4.75,5.00,5.25,5.50,5.75,6.00,6.25,6.50,6.75,7
.00,7.25,7.50,7.75,8.00,8.25,8.50,8.75,9.00,9.25,9.50,9.75

ps1

!1:4.36,4.75,5.18,5.65,6.16,6.72,7.33,7.99,8.71,9.50,10.36,
11.30,12.32,13.44,14.65,15.98,17.42,19.00,20.72,22.60,24.64
,26.87,29.30

ps2

!2:4246,4496,4746,4996,5246,5496,5746,5996,6246,6496,6746,6
996,7246,7496,7746,7996,8246,8496,8746,8996,9246,9496,9746
```

V1.30.0+

5.37 Print Zoom Position (pz)

Command Type	**: New (Legacy-Type)**
Aliases	**: zp**
Syntax	**: pz[m]**
Returns	**: Current zoom position**

Description

Report the current focal length of the lens in raw encoder counts. This command can take a units modifier (m) indicating in this case to print the results in millimeters if the lens is capable of reporting focal length by length units.

Examples:

```
pz <CR>

20000

pzm <CR>

55mm
```

V1.30.0+

5.38 Query Port ID (qp)

Command Type : **New (Legacy-Type)**
Syntax : **qp**

Returns : **Numerical port number**

Description

Print the number assigned to the current interface. This number can be used in port routing. Port numbers are fixed for hardware interfaces, but are assigned first-come-first-served for dynamic interfaces like TCP/IP connections.

Examples:

```
qp <CR>
```

```
2
```

v1.30.0+

5.39 Set Response Modes (rm)

Command Type : **Legacy**
Syntax : **rm <verbose{0, 1}>[, <new{0, 1}>]**

Returns : **OK or 0**

Description

This command is used to set the response modes. The first argument can be 0 or 1, representing terse mode or verbose mode, respectively. In verbose mode all characters sent to the device are echoed back out and most commands return confirmations and full status strings. In terse mode, no characters are echoed and command responses are limited to error codes for legacy behavior. The second argument is optional (if not supplied it defaults to 0), and indicates the protocol mode – legacy (0) or new (1). Legacy mode maintains protocol compatibility with legacy Canon EF-232 libraries version 15 and earlier.

The response modes are set on a per-port basis, the change being effective for the port over which the command is sent.

Examples:

```
rm0,1 <CR>
```

OK

```
rm0,0 <CR>
```

0

v1.30.0+

5.40 Set EEPROM Byte (se)

Command Type : **Legacy**
Syntax : **se <byte_number>,<value>**

Returns : **DONE**

Description

This command will temporarily change an EEPROM setting. The setting is not actually committed to non-volatile memory with this command, and therefore if the device is given a bad setting that causes it to malfunction, removing power will clear the condition. The index of the byte to change is given in <byte_number> and the new value is given by <value>. The index is given as a decimal number and the value is given as a hexadecimal number.

The commonly configured parameters are almost always available through other commands, so this command is used to alter the remaining parameters. These parameters are often low-level timing configurations. Contact Birger Engineering if your programming needs might require modification to these parameters.

Example:

```
se4,0 <CR>
```

DONE

v1.30.0+

5.41 Set Focus Counter Offset (sf)

Command Type : Legacy
Syntax : sf <count>

Returns : OK

Description

This command sets the focus counter to mark the current lens focus position with the value specified. For example, moving the focus to the zero position using the 'mz' command will yield the lens' encoder position for the zero stop, which will almost always be non-zero. Using this command with an argument of zero at that point allows you to mark that position as zero and then zero would be the position input to the focus absolute commands to return to that same position. This offset is internally set whenever the ‘la’ command is used to learn the focus range, such that the near focus of the lens is marked as the zero reference.

Example:

```
sf0 <CR>
```

```
OK
```

5.42 Set Special Modes (sm)

Command Type : **Legacy**
Syntax : **sm <mode_flags>**

Returns : **DONE**

Description

This command is used to control special modes of the device. <mode_flags> is given as a decimal number which is the sum of the individual flags to be set. If you need to know more about how the modes work, contact Birger Engineering, Inc.

Table 5.43.1. Special Mode Flags

Bit	***Decimal Value***	***Mode***
0	1	Fast focus mode.
1	2	Compatible focus mode (temporary if bit 0 is also set).
2	4	Switch 'ma' command to use the servo aperture routine.
3	8	Turn on background lens querying.
4	16	Show distance stop progress.
5	32	Reserved.
6	64	Reserved.
7	128	Reserved.

Examples:

```
sm4 <CR>
```

DONE

```
sm12 <CR>
```

DONE

V1.30.0+

5.43 Print Serial Number (sn)

Command Type : **Legacy**
Syntax : **sn**

Returns : **Device serial number**

Description

This command prints the device serial number. The serial number is used as an input to various commands as an unlock code.

Compatibility Mode Example:

```
sn <CR>
```

```
67890
```

Normal Mode Example:

```
sn <CR>
```

```
1234567890
```

v1.30.0+

5.44 Set Spontaneous Responses (sr)

Command Type	**: Legacy**
Syntax	**: sr {0, 1}**
Returns	**: OK**

Description

This command turns spontaneous responses on or off for the port over which the command is sent; 0 for off, 1 for on. For status updates, focus and aperture changes will be automatically broadcast to every other port in response to a change on any particular port. If background lens querying is also turned on (see command 'sm') then changes in lens status (focal length, etc.) are broadcast to all ports. Finally, GPIO changes will also be broadcast to all ports.

Note that this does not cause commands to be echoed out, rather updates take the form of status strings as given in Section 3.9. Spontaneous updates are set on a port-by-port basis. The status changes are broadcast, but only the communications ports over which this command has been sent to turn on responses will actually receive them.

Examples:

```
sr0 <CR>

OK

sr1 <CR>

OK
```

V1.30.0+

5.45 Short Version String (vs)

Command Type : **Legacy**
Syntax : **vs**

Returns : **s:<module>v<version>**

Description

This command prints the short version string that describes the code currently executing. This string is given with the identifier “s:” followed by the short module ID in <module>, followed by the letter 'v', followed by the decimal revision number. The legacy Canon EF-232 Library will return 'C2' as the module ID, and this is also what is returned when the interface is running in legacy compatibility mode.

Compatibility Mode Example:

```
vs <CR>
```

```
s:C2v17
```

Normal Mode Example:

```
vs <CR>
```

```
s:B1v1.30.0
```

5.46 Write Console Settings to Non-Volatile Memory (wc)

Command Type	**: New (Legacy-Type)**
Syntax	**: wc**
Returns	**: OK**

Description

This command writes only the console configurations to non-volatile memory. The console configurations written are the settings: baud rate (if applicable), output verbosity, new language output mode, character echo, prompt suppression, line feeds sent, VT100 terminal mode, and legacy compatibility mode.

v1.30.0+

5.47 Write EEPROM Parameters (we)

Command Type	**: Legacy**
Syntax	**: we <serial_number>**
Returns	**: DONE**

Description

This command will commit the temporary non-volatile settings to EEPROM. This will make the settings permanent. You may damage your device, which will require sending it back to Birger Engineering, if you commit bad parameters to the non-volatile memory. This command requires the device serial number as a confirmation. In compatibility mode the serial number is parsed numerically, so leading zeros are not significant, while in normal mode the full serial number is required.

Example:

```
we1234 <CR>

DONE
```

v1.30.0+

5.48 Move Absolute Zoom position (za)

Command Type	**: New (Legacy-Type)**
Syntax	**: za[m] <position>**
Returns	**: DONE<rposition>,flag**
Returns	**: DONE<rposition>mm,flag**

Description

This command makes the zoom mechanism move to the specified zoom count position, specified as an absolute position given in <position>. If an input value would move the zoom out of the legal range the value is rounded to the boundary (i.e. min/max count). <rposition> is the actual position that the count has been moved to, <flag> is 1 if the lens reports having hit a stop, and 0 if it hasn't. Note that the behavior of this flag may be emulated if the lens cannot report hitting a stop.

This command can also take a distance unit modifier if the lens can support moving by focal length – [m] which means mm (millimeters) in this case.

The target position may not be reached exactly, but will be within a few encoder counts of the target position.

Examples:

```
za5000 <CR>

DONE5000,0

zam65 <CR>

DONE65mm,0
```

5.49 Move Zoom Incremental (zn)

Command Type	**: New (Legacy-Type)**
Aliases	**: zi**
Syntax	**: zn[m] <signed_value>**
Returns	**: DONE<signed_return_value>,flag**
Returns	**: DONE<signed_return_value>mm,flag**

Description

This command makes the zoom mechanism move incrementally the number of counts specified in <signed_value>. If the number specified is positive then the zoom moves towards the telephoto position. If the number is negative then the zoom moves towards the wide position. If the number of counts specified is out of range, the mechanism will move as far as it can, <signed_returned_value> is the actual number of counts moved, and <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that the behavior of this flag may be emulated if the lens cannot report hitting a stop.

This command can also take a distance unit modifier if the lens can support moving by focal length – [m] which means mm (millimeters) in this case.

The ending position may not be reached exactly, but will be within a few encoder counts of the desired position.

Examples:

```
zi1000 <CR>

DONE1000,0

zi-5000 <CR>

DONE-5000,0

zim10 <CR>

DONE10mm,0
```

5.50 Move Zoom to Telephoto Position (zt)

Command Type : **New (Legacy-Type)**
Syntax : **zt[m]**

Returns : **DONE<rposition>,<flag>**
Returns : **DONE<rposition>mm,<flag>**

Description

This command will move the zoom mechanism to the full telephoto position. The numerical position will be returned in rposition. <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that the behavior of this flag may be emulated if the lens cannot report hitting a stop.

This command can also take a distance unit modifier if the lens can support moving by focal length – [m] which means mm (millimeters) in this case.

The ending position may not be reached exactly, but will be within a few encoder counts of the desired position.

Example:

```
zt <CR>

DONE65535,1

ztm <CR>

DONE85mm,1
```

v1.30.0+

5.51 Move Zoom to Wide Position (zw)

Command Type	**: New (Legacy-Type)**
Syntax	**: zw[m]**
Returns	**: DONE<rposition>,<flag>**
Returns	**: DONE<rposition>mm,<flag>**

Description

This command will move the zoom mechanism to the full wide position. The numerical position will be returned in rposition. <flag> is 1 if the lens reports having hit a stop, 0 if it hasn't. Note that the behavior of this flag may be emulated if the lens cannot report hitting a stop.

This command can also take a distance unit modifier if the lens can support moving by focal length – [m] which means mm (millimeters) in this case.

The ending position may not be reached exactly, but will be within a few encoder counts of the desired position.

Example:

```
zw <CR>

DONE0,1

zwm <CR>

DONE50mm,1
```

6 New Command Reference

This section describes the enhanced commands that are new with the introduction of the Birger V2 Lens Controller platform. All of the commands in this section require a space between the command and its arguments.

v1.30.0+

6.1 Send Ethernet Broadcast Packet (announce)

Command Type : **New**
Syntax : **announce {0,1}**

Returns : **{Limited|Directed} broadcast sent**

Description

Use this command to generate a packet from the device on your local network. This packet can often be used by routers for device detection. It is also useful for diagnostics.

The packet is constructed with the following details:

Type: UDP
Source and destination ports: 30303
Destination MAC: FF:FF:FF:FF:FF:FF (broadcast)
Destination IP (Limited): 255.255.255.255
Destination IP (Directed): x.x.x.255 (source IP address with 255 as the host address)

Examples:

```
announce 0 <CR>
```

Limited broadcast sent

```
announce 1 <CR>
```

Directed broadcast sent

v1.30.0+

6.2 Boot Application (boot)

Command Type : New
Syntax : boot <app_index>

Returns : Error or boots the application specified

Description

This command will boot any valid application that is loaded into the device memory. The application is specified by the application index number retrieved from the 'list' command.

Examples:

```
boot 1 <CR>
```

v1.30.0+

6.3 List Commands (cmds)

Command Type : New
Syntax : cmds [group][alias]

Returns : List of all commands in short-form

Description

Use this command to print a list of all of the available commands without their description. An optional command group can be specified to list only that group. If the keyword 'alias' is used then the commands will be printed with their aliases.

Examples:

```
cmds <CR>

cmds group <CR>

cmds focus <CR>

cmds focus alias<CR>
```

6.4 Set/Clear/Display EF Compatibility Mode (comp)

Command Type	**: New**
Syntax	**: comp [0, 1]**
Returns	**: Current setting or OK**

Description

This command sets the EF compatibility mode for lens families other than EF. It is on by default as shipped. This setting is part of the parameter set that can be saved to non-volatile memory with the 'we' command. If given with no arguments it prints the current setting; 0 = compatibility mode off, 1 = compatibility mode on.

As an example of how compatibility mode works, for the MFT lens family aperture stops will be auto-generated or emulated on ¼ stops if they can be, and raw encoder numerical ranges for legacy commands will have the same polarity as they do on EF lenses, even if that may not be the actual case. This ensures maximum compatibility with any pre-existing application user software that was written for the Birger Canon EF-232 platform.

Examples:

```
comp <CR>

1

comp 0 <CR>

OK
```

6.5 Reset Console Configurations (creset)

Command Type : New
Syntax : creset

Returns : OK

Description

Restores the console settings (the settings for the current interface over which this command is sent) to their defaults. This will also reset the line-end auto-detection.

Table 6.5.1. Console Default Settings

Interfaces	***USB and Serial***	***TCP/IP***
Setting:		
Character Echo	True	False
Line-Feeds	False	True
Suppress Prompt	False	False
New Language	True	True
Verbose Mode	True	True
Compatibility Mode	True	True
VT100 Mode	False	True

Example:

```
creset <CR>

OK
>
```

6.6 Create User Login (cuser)

Command Type : New
Syntax : cuser <index> <name> <password>

Returns : Error or login account is created

Description

Use this command to create a user/password combination that can be used to authorize a TCP/IP connection to the device if authorized connections are enabled. The name and password are case-sensitive and spaces are not allowed in the name or password. You will be prompted for an existing user/password in order to authorize the creation of a new login account.

Each account is stored in an index and creating a login account over an existing index will overwrite that login. The number of available login accounts is limited, **<TBD refer to earlier section describing TCP/IP interfaces and accounts>**

Note that these names as passwords are normally sent in plaintext, therefore it is not recommended to use this command unless it is over one of the non-network hardwired ports (USB/serial) or over a network interface that is secured with TLS.

Example:

```
cuser 1 newuser passw29$ <CR>

Enter a user name for authentication
User:

Password:

User newuser created
```

6.7 DHCP Client Commands (dhcp)

Command Type : New
Syntax : dhcp [info, off, on, renew, request]

Returns : Various, see below

Description

This command can be used to access the DHCP client facilities built into the device. The current state of the DHCP system (on or off) can be saved to non-volatile memory so that the device always boots with the system on or off as desired.

Table 6.7.1. DHCP Commands

Command	*Description*
info	Prints all of the available information for the current DHCP lease.
off	Turns off DHCP. This setting can be saved to non-volatile memory.
on	Turns on DHCP. This setting can be saved to non-volatile memory.
renew	Renew the DHCP lease.
request	Request the DHCP server for a particular IP address. The assignment of the requested address is not guaranteed.

Examples:

```
dhcp on <CR>

dhcp on success

dhcp renew <CR>

dhcp renew success

dhcp request 192.168.1.203 <CR>

dhcp request success
```

6.8 DNS Client Commands (dnsc)

Command Type	**: New**
Syntax	**: dnsc [del, info, off, on, lookup]**
Returns	**: Various, see below**

Description

This command can be used to access the DNS client facilities built into the device.

Table 6.8.1. DNS Client Commands

Command	***Description***
del	Remove any or all of the specified hosts from the current directory.
info	Prints all of the available information in the host directory.
off	<TBD>
on	<TBD>
lookup	Lookup the IP address for the specified hostname.

Examples:

<TBD>

6.9 Echo Characters (echo)

Command Type : New
Syntax : echo "characters to echo" [port_id]

Returns : Characters to echo or nothing

Description

This command can be used to print information to a port.

Examples:

```
echo "this is a test" <CR>
This is a test

echo "characters sent to port 1" 1 <CR>
```

v1.30.0+

6.10 Command List with Brief Descriptions (help)

Command Type	**: New**
Syntax	**: help [command, group][alias]**
Returns	**: Command name(s) and description(s)**

Description

Use this command to print a list of all commands with brief descriptions, a list of groups, a command group, or an individual command. If ‘alias’ is specified then the commands are printed with their aliases.

Examples:

```
help <CR>

help group <CR>

help focus <CR>

help focus alias <CR>

help ma <CR>
```

v1.30.0+

6.11 Set Login Control Options (lctrl)

Command Type : New
Syntax : lctrl [disable, enable, attempts <num_attempts>]

Returns : Error or OK

Description

This command is used to set options for the behavior of the TCP/IP Telnet console connections. Login authorization can be disabled or enabled and the number of login attempts before the connection is closed can be specified. If login authorization is turned on, the user must enter a name and password corresponding to any of the accounts stored in the device with the 'cuser' command before the connection can proceed to the command interpreter. Login control is enabled by default and the default username and password are both 'birger' without the quotes.

For security purposes, this command can only be executed over a non-network hardwired interface (USB/serial).

Examples:

```
lctrl disable <CR>

OK

lctrl enable <CR>

OK

lctrl attempts 5 <CR>

OK
```

v1.30.0+

6.12 List Loaded Applications (list)

Command Type : New
Syntax : list

Returns : List of loaded applications

Description

Example:

```
list <CR>

App Index     : 0
App Name      : Birger Lens Controller V2
App Version   : 1.30.0
Required HW   : 5
Valid         : Yes
```

v1.30.0+

6.13 Mark an Application for Boot (mkapp)

Command Type : **New**
Syntax : **mkapp <app_index>**

Returns : **Error or OK**

Description

This command is used to mark the application that will be automatically booted when the device is turned on. The application index to be specified is the same that is learned with the 'list' command. If the application index is specified as 255 then the application in memory with the highest version number will be automatically booted. The default setting is 255.

Examples:

```
mkapp 0 <CR>

OK

mkapp 255 <CR>

OK
```

6.14 Print Network Information (netinfo)

Command Type : New
Syntax : netinfo

Returns : Network information

Description

This command prints the current network settings, which includes the manually or dynamically assigned IP addresses, device MAC address, DHCP state, link state, and device network name.

Example:

```
netinfo <CR>

Host Name    : BirgerController45
IPv4 Address: 192.168.1.203
Mask         : 255.255.255.0
Gateway      : 192.168.1.1
DNS1         : 192.168.1.1
DNS2         : 0.0.0.0
MAC Address : 58:47:ca:80:00:00
DHCP is ON
Link is UP
Status: Ready
```

v1.30.0+

6.15 Query Index of Running Application (qapp)

Command Type : **New**
Syntax : **qapp**

Returns : **Index of currently running application**

Description

This command reports the index of the currently running application. The application indices match those that are returned by the 'list' command.

Examples:

```
qapp <CR>
```

0

v1.30.0+

6.16 Query Port Security (qsec)

Command Type : New
Syntax : qsec

Returns : Open or Secure

Description

This command will print the security level of the interface over which it is sent. Hardwired ports (USB/serial) will always be shown as 'Secure.' TCP/IP connections will be shown as 'Secure' if a TLS connection has been established, otherwise the interface will be shown as 'Open.' A port that is open is a port whose communications can be easily intercepted. Even though hardwired ports are shown as secure, the user is responsible for the actual final security of the physical device.

Examples:

```
qsec <CR>

Secure

qsec <CR>

Open
```

v1.30.0+

6.17 Quit the Command Processor (quit)

Command Type : New
Syntax : quit

Returns : * Quitting the Command Processor *** or nothing**

Description

This command will close the command processor and shut down the interface over which it is issued. This command is only applicable for dynamic (e.g. TCP/IP) connections and not for hardwired connections (USB/serial). It may not always be possible to receive the final closing text from this command as the socket connection is closed quickly after the final text is sent.

Example:

```
quit <CR>

 *** Quitting the Command Processor ***
```

v1.30.0+

6.18 Reset the Device (reset)

Command Type : New
Syntax : reset

Returns : Nothing

Description

This command causes a device reset, simulating as if the power was removed and restored, although not actually being a true hardware reset. All temporary settings are lost and all settings are restored from non-volatile memory.

Examples:

```
reset <CR>
```

v1.30.0+

6.19 Set Port Routing (route)

Command Type : **New**
Syntax : **route [port_id]**

Returns : **Various, see below**

Description

This command is used to set up a port to receive an echo of all legacy commands and their output. Note that it applies to legacy commands only, all new commands (commands in this Section 6) will not be echoed. The port ID is a number that is obtained by issuing the query port command 'qp.' A command is only echoed after it has been accepted, it will not be echoed character-by-character. This command is useful for developing and debugging user applications by allowing a second port to observe what is happening on the application communication port.

Examples:

```
route <CR>
```

```
There is no port routing set up
```

```
route 0 <CR>
```

```
Echoing all legacy communications on port 0 to port 1 (this
port)
```

```
route -1 <CR>
```

```
Port routing canceled
```

6.20 Set/Print Detailed Response Modes (sdr)

Command Type : New
Syntax : sdr [value[, port_id[, modifier]]]

Returns : Current settings or OK

Description

This command is used to set or print the configurations for the console interfaces, including the default template that is used for all new TCP/IP connections. It has several variations:

sdr value – sets the hex value as a bitmask for the settings
sdr x,port – print the settings as a hex word for the specified port
sdr x,254 – print all of the settings for all ports including the default template
sdr value,254 – sets the hex value as the default connection template directly to EEPROM
sdr value,port – sets the hex value as a bitmask for the settings for the specified port
sdr value,port,1 – same as above, but writes-through to the EEPROM instead of setting the current working settings

Table 6.19.1. Command Console Settings Bitmask Values

Setting	***Value***	***Description***
Verbose Mode	0x01	When set to true, puts the command interface in the verbose mode provided by the original Canon EF-232 product with libraries version 17 or later, otherwise the interface behaves as the original product with libraries earlier than version 17.
New Language	0x02	When set to true, puts the command interface in the new-language mode provided by the original Canon EF-232 product with libraries version 17 or later, otherwise the interface behaves as the original product with libraries earlier than version 17.
Character Echo	0x04	When set to true, all characters received by the device are echoed back out to the sender as each character is received.
Suppress Prompt	0x08	When set to true, no prompt character (>) will be emitted when the device is ready to receive a command.
Line-Feeds	0x10	When set to true, carriage-returns will be followed by line-feeds, otherwise the device will only emit carriage-returns as line ends.
VT100 Mode	0x20	When set to true, the device will emit VT100 control strings for richer text and terminal control.

Compatibility Mode	0x40	When set to true, the interface and command output will emulate the original Canon EF-232 product. This is likely required for compatibility with any pre-existing user applications targeting the original product. When set to false, certain commands will emit more information than their original counterparts.

Examples:

```
sdr <CR>

Current settings:
Verbose           : true
New language      : true
Echo characters   : true
Suppress prompt   : false
Send line feed    : false
VT100 mode        : false
Compatibility     : true

sdr 4a <CR>

OK
```

6.21 Set DNS Address (setdns)

Command Type	**: New**
Syntax	**: setdns {1, 2} <ip_addr>**
Returns	**: Error or success**

Description

Set the addresses for DNS lookup. Two addresses can be specified, a primary and a backup. This is a live update, the changes will be made immediately without needing to restart the TCP/IP stack. These settings can be saved to non-volatile memory using the 'we' command.

Example:

```
setdns 1 192.168.1.1 <CR>

Set DNS 1 address success

setdns 2 192.168.1.2 <CR>

Set DNS 2 address success
```

v1.30.0+

6.22 Set Serial Port Baud Rate (setbaud)

Command Type	**: New**
Syntax	**: setbaud [baud_rate][, port_id][, w]**
Returns	**: Current baud rate or OK**

Description

This command prints or sets the baud rate for a serial port. Without any arguments it will print the baud rate for the port over which it was issued. If it was issued over a USB port, the baud rate is not applicable and will be returned as zero. Arguments can be specified in any order, low numbers are interpreted as ports and high numbers as baud rates.

It can be issued over a hardware serial port, and the command's output will be issued at the new baud rate unless 'w' is specified, in which case the change is made to the EEPROM setting for the serial port instead of being applied instantly. This setting is also saved to non-volatile memory using the 'we' command.

System defined ports are as follows:

For firmware versions 1.30.0 through 1.30.9:

UART 0 (camera interface serial port) = 0
USB CDC 0 = 1
USB CDC 1 = 2

For firmware versions 1.30.10 and later:

UART 0 (camera interface serial port) = 0
UART 1 (accessory serial port) = 1
USB CDC 0 = 2
USB CDC 1 = 3

Examples:

```
setbaud <CR>
```

115200

```
setbaud 9600 <CR>
```

```
setbaud 9600,1 <CR>
```

OK

v1.30.0+

6.23 Set IP Address and Mask (setip)

Command Type	**: New**
Syntax	**: setip <ip_address> <network_mask>**
Returns	**: Error or success**

Description

This command is used to set the device's IP address and network mask. This is a live update, the changes will be made immediately without needing to restart the TCP/IP stack. These settings can be saved to non-volatile memory using the 'we' command.

Example:

```
setip 192.168.1.200 255.255.255.0 <CR>
```

v1.30.0+

6.24 Set Gateway Address (setgw)

Command Type : New
Syntax : setgw <gateway_ip_address>

Returns : Error or success

Description

This command is used to set the device's gateway address. This is a live update, the changes will be made immediately without needing to restart the TCP/IP stack. This setting can be saved to non-volatile memory using the 'we' command.

Example:

```
setgw 192.168.1.1 <CR>

Set gateway address succeeded
```

6.25 List All Open Sockets (sock)

Command Type : New
Syntax : sock

Returns : List of all open sockets

Description

This command can be used to list all of the open TCP/IP sockets and their information, which includes the remote IP address, remote MAC address, remote port, local port, the security state (TLS is in use or not), and the state of the connection.

Example:

```
sock <CR>

Socket number : 0
Remote IP Addr: 192.168.1.243
Remote MAC    : 5C:52:30:A1:A9:62
Remote Port   : 57613
Local Port    : 992
Secure        : Yes
State         : Connected

Socket number : 1
Remote IP Addr: 0.0.0.0
Remote MAC    : 00:00:00:00:00:00
Remote Port   : 0
Local Port    : 992
Secure        : No
State         : Listening
```

v1.30.0+

6.26 Set TCP/IP Port/Protocol Configurations (spt)

Command Type	**: New**
Syntax	**: spt [icmp, telnet] [port] [disable, enable] [open, secure]**
Returns	**: Error or OK**

Description

This command is used to configure the TCP/IP interfaces. An interface can be disabled or enabled, be configured to operate over a specific port, and be open or require TLS security. The ICMP protocol can only be enabled or disabled. Note that disabling ICMP (network ping) may cause management issues with routers or other devices that use it. These are not live updates, they require restarting the TCP/IP stack to take effect. These settings are saved to non-volatile memory using the 'we' command.

Examples for Telnet:

```
spt telnet disable <CR> - disables telnet

spt telnet 23 enable <CR> - sets telnet to work on port 23
and enables it

spt telnet 23 enable secure <CR> - sets telnet to work on
port 23 and enforces TLS for connections.
```

Examples for ICMP:

```
spt icmp disable <CR>

spt icmp enable <CR>
```

v1.30.0+

6.27 TCP/IP Stack Commands (stack)

Command Type	**: New**
Syntax	**: stack [down, restart, up] [preserve]**
Returns	**: Error or OK**

Description

Use this command to turn the TCP/IP stack on or off. This setting can be saved to non-volatile memory using the 'we' command or by specifying 'preserve' as an argument. The stack can also be restarted which is required when changing the configurations using the 'spt' command. A restart is equivalent to shutting the stack down and turning it back on. Note that the stack cannot be restarted or shut down and turned on an unlimited amount of times due to the way the device manages memory. Eventually the device will need to reset to reclaim memory resources. If the stack is shut down or restarted over a TCP/IP connection, that connection will be closed.

Examples:

```
stack down <CR>

Stack down succeeded

stack down preserve <CR>

Stack down succeeded

stack up <CR>

Restarting the stack
Stack up succeeded
```

v1.30.0+

6.28 Print Chip Temperature in Celsius (temp)

Command Type	**: New**
Syntax	**: temp**
Returns	**: Microprocessor die temperature in degrees Celsius**

Description

Issue this command to print the internal temperature of the microprocessor.

Example:

```
temp <CR>
```

```
temp:49.3
```

v1.30.0+

6.29 Enable/Disable Legacy Command Timestamps (time)

Command Type	**: New**
Syntax	**: time {off, on}**
Returns	**: OK**

Description

Turn time stamps for legacy commands on or off. When time stamps are turned on, all legacy-type commands will have their execution time measured and the results printed. **<TBD – refer to section on timestamps>**

Examples:

```
time off <CR>

OK

time on <CR>

OK
```

6.30 Print USB Port Status (usbstat)

Command Type	**: New**
Syntax	**: usbstat**
Returns	**: OK**

Description

Print the status of the USB port connection.

Examples:

```
usbstat <CR>
```

```
USB Roles :Direction is UFP(500mA), Cable Orientation is
Normal, Connected Device Type is USB Host
```

7 Bootloader Command Reference

This section describes commands available while the bootloader is active. These commands are not available while the application is executing, although the application may have some of the same commands also implemented.

Unlike the legacy Birger product and unlike legacy-type commands emulated in the application, all bootloader commands must have a space between the command and its arguments.

7.1 Boot Library (bl)

Command Type : New
Syntax : bl

Returns : TBD

Description

This command boots the application in memory with the highest version number. Control is transferred to the application and the application handles all further command processing.

Example:

```
bl <CR>
```

<TBD - update possible output>

7.2 Boot Application (boot)

Command Type	**: New**
Syntax	**: boot <app_index>**
Returns	**: Error or boots the application specified**

Description

This command will boot any valid application that is loaded into the device memory. The application is specified by the application index number retrieved from the ‘list’ command.

Examples:

```
boot 1 <CR>
```

7.3 Command List with Brief Descriptions (help)

Command Type : New
Syntax : help

Returns : Command names and descriptions

Description

Use this command to print a list of all commands with brief descriptions.

Examples:

```
help <CR>
```

7.4 Hardware Version (hv)

Command Type : Legacy
Syntax : hv

Returns : Device hardware revision

Description

This command prints the hardware revision of the device.

Example:

```
hv <CR>
```

```
5
```

7.5 List Applications (list)

Command Type : **New**
Syntax : **list**

Returns : **List of loaded applications**

Description

Example:

```
list <CR>

App Index     : 0
App Name      : Birger Lens Controller V2
App Version   : 1.30.0
Required HW   : 5
Valid         : Yes
```

7.6 Mark an Application for Boot (mkapp)

Command Type : **New**
Syntax : **mkapp <app_index>**

Returns : **Error or OK**

Description

This command is used to mark the application that will be automatically booted when the device is turned on. The application index to be specified is the same that is learned with the ‘list’ command. If the application index is specified as 255 then the application in memory with the highest version number will be automatically booted. The default setting is 255.

Examples:

```
mkapp 0 <CR>

OK

mkapp 255 <CR>

OK
```

7.7 Reset the Device (reset)

Command Type : New
Syntax : reset

Returns : Nothing

Description

This command causes a device reset, simulating as if the power was removed and restored, although not actually being a true hardware reset. All temporary settings are lost and all settings are restored from non-volatile memory.

Examples:

```
reset <CR>
```

7.8 Set Response Modes (rm)

Command Type **: Legacy/New**
Syntax **: rm <verbose{0, 1}>[, <new{0, 1}>[,<all_values_hex>]]**

Returns **: OK or 0**

Description

This command is used to set the response modes. The first argument can be 0 or 1, representing terse mode or verbose mode, respectively. In verbose mode all characters sent to the device are echoed back out and most commands return confirmations and full status strings. In terse mode, no characters are echoed and command responses are limited to error codes for legacy behavior. The second argument is optional (if not supplied it defaults to 0), and indicates the protocol mode – legacy or new. Legacy mode maintains protocol compatibility with legacy libraries version 15 and earlier.

The response modes set in the bootloader are copied forward into the application when the application is booted, but only for the hard-wired serial port and the first instance of any USB virtual serial port. This is done as a convenience and to maintain backward compatibility with the legacy Birger device. Unlike the legacy device, they cannot be committed to non-volatile memory in the bootloader, that function must be performed from within the application.

An optional third argument, if specified, will override the first two arguments and set all of the response modes as a bitmask. The third argument shall be specified as a hexadecimal number. The bitmask values are the same as those for the 'sdr' command listed in the application command section.

The response mode is set on a per-port basis, the settings being active for the port over which the command is sent.

Examples:

```
rm 0 <CR>

0

rm 0,1 <CR>

OK
```

7.9 Serial Number (sn)

Command Type	**: Legacy**
Syntax	**: sn**
Returns	**: Device serial number**

Description

This command prints the full device serial number. This command does not have compatibility mode capability.

Example:

```
sn <CR>

1234567890
```

7.10 Short Version String (vs)

Command Type : **Legacy/New**
Syntax : **vs**

Returns : **s:<module>v<version>**

Description

Legacy: Prior to bootloader version 44, this command prints an extended version string.

New: This command prints the short version string that describes the code currently executing. This string is given with the identifier "s:" followed by the short module ID in <module>, followed by the letter 'v', followed by the decimal revision number. The bootloader will return 'BE' as the module ID.

Example:

```
vs <CR>
```

`s:BEv50`