

# Programming Manual

---

## About this manual

This manual describes how to control active Qontrol modules. It describes the communication protocol, and both human-readable and binary command structure.

Binary commands allow your application to communicate with your Qontrol modules more quickly and efficiently, with maximum precision.

## Module compatibility

All Qontrol driver modules manufactured before 2018 are compatible. Specific device compatibility, if different, is noted throughout.

- Q8 (HW04–06)
- Q8iv (all versions)
- Q8b (all versions)

## Contents of this document

Communication protocol.....	2
Human-readable control .....	2
‣ <i>Command Reference</i>	2
‣ <i>Error Codes</i>	5
Binary control.....	6
‣ <i>Data structure</i>	6
‣ <i>Header</i>	6
‣ <i>Command index</i>	7
‣ <i>Address</i>	8
‣ <i>Data</i>	9
Example ASCII-binary translations.....	9
Command processing flowchart.....	11
Notes and disclaimer.....	12
Revision History .....	12

# Programming Manual

---

## Communication protocol

Modules communicate with a PC and with other Qontrol devices via a serial port and the RS-232/UART protocol, with the following fixed settings:

Table 1: Serial port parameters.

115200 baud

8 data bits

1 stop bit

no flow control

Qontrol backplanes convert this RS-232 port into a standard mini-USB port. Qontrol uses genuine FTDI parts; drivers for all major operating systems can be found online at [ftdichip.com/FTDrivers](http://ftdichip.com/FTDrivers).

## Human-readable control

Qontrol modules support a simple, easy-to-code communications language based on human-readable ASCII-formatted commands and responses.

### ► Command Reference

Commands either set or get a value, with the operators = and ?, respectively. Valid commands have the following format, [command][channel] [= or ?] [value]\n, where: [command] is the command's name; [channel] is the target channel index, or is omitted if the command does not target a channel; and [value] is the command's input in the case of a set operation, or is absent in the case of a get operation. Space characters (0x20) are ignored. All commands must be terminated with either a line feed character \n (0x0A), a carriage return character \r (0xD0), or both. Some global commands allow the ? or = operator to be omitted.

Table 2: Command reference.

Command Syntax	Function
<code>V[ch] = [float]</code>	Set the voltage on channel [ch] to [float] volts.
<code>I[ch] = [float]</code>	Set the current on channel [ch] to [float] milliamperes (software-controlled).
<code>V[ch]?</code>	Get the voltage on channel [ch] in volts. When channel is in voltage control mode, this command returns the currently set voltage value. In current and power control modes, it returns the software-controlled voltage value.
<code>I[ch]?</code>	Get the current on channel [ch] in milliamperes.
<code>P[ch]?</code>	Get the power on channel [ch] in milliwatts.

# Programming Manual

Table 2: Command reference.

Command Syntax	Function
<b>VMAX[ch] = [float]</b>	Set the over-voltage limit on channel [ch] to [float] volts.
<b>IMAX[ch] = [float]</b>	Set the over-current limit on channel [ch] to [float] milliamperes.
<b>VMAXALL = [float]</b>	Set the over-voltage limit on all channels to [float] volts.
<b>IMAXALL = [float]</b>	Set the over-current limit on all channels to [float] milliamperes.
<b>VCAL[ch] = [int]</b>	Set the raw voltage calibration for [ch]. See <i>Application Information</i> ▶ <i>Calibration</i> for more information.
<b>ICAL[ch] = [int]</b>	Set the raw current calibration for [ch]. See <i>Application Information</i> ▶ <i>Calibration</i> for more information.
<b>VCAL[ch]?</b>	Get the raw voltage calibration for [ch]. See <i>Application Information</i> ▶ <i>Calibration</i> for more information.
<b>ICAL[ch]?</b>	Get the raw current calibration for [ch]. See <i>Application Information</i> ▶ <i>Calibration</i> for more information.
<b>VERR</b>	
<b>IERR[ch]?</b>	Estimate the current uncertainty (standard deviation) for [ch]. <i>Firmware v1.63 and later.</i>
<b>VIPALL?</b>	Get the voltages, currents, and powers sourced by each channel.
<b>VFULL?</b>	Get the full-scale voltage.
<b>IFULL?</b>	Get the full-scale voltage (Q8iv only)
<b>NCHAN?</b>	Get the number of output channels. <i>Firmware v1.63 and later.</i>
<b>FIRMWARE?</b>	Get the firmware version.
<b>ID?</b>	Get the unique device identifier.
<b>LIFETIME?</b>	Get the total number of seconds for which the device has been powered.

**Table 2:** Command reference.

Command Syntax	Function
<b>NVMALL?</b>	Get the contents of each word stored in non-volatile memory. See module data sheet for an overview of the memory's contents.
<b>NVMALL = 0</b>	Erase the NVM. Safety lock must be disabled (see <b>SAFE</b> ).
<b>LOG?</b>	Get the device log. <i>Firmware v1.63 and later</i> .
<b>ECHO = [1/0]</b>	If <b>[1/0] = 1</b> then the device will echo back characters it receives, otherwise, it will not (default is <b>[1/0] = 0</b> ). <i>Warning: activating echo can disrupt machine-machine communication protocols.</i>
<b>LED = [1/0]</b>	If <b>[1/0] = 1</b> (the default) then the device LEDs will be used to display device status. If <b>[1/0] = 0</b> then the device LEDs will be disabled (e.g. to aid single-photon counting).
<b>NUP = 0</b>	Reset the daisy-chain configuration by telling the first module in the chain that there are zero devices upstream.
<b>NUPALL?</b>	Read out the daisy chain configuration.
<b>ADCT = [int]</b>	Set the number of instruction cycles for which to integrate current (ADC) readings between 1 and 31.
<b>ADCT?</b>	Retrieve the number of instruction cycles for current readings.
<b>ADCN = [int]</b>	Set the number of averages collected for each current (ADC) reading. The application software must wait at least 1 s before requesting a current reading after sending this command. <i>Firmware v1.63 and later</i> : this value is rounded down to the nearest power of two.
<b>ADCN?</b>	Retrieve the number of averages collected for each current reading.
<b>CCFN = [int]</b>	Set the current control filter response to have a characteristic number of samples of <b>[int]</b> . (Q8 only)

# Programming Manual

Table 2: Command reference.

Command Syntax	Function
<b>RESET</b>	Perform a soft reset on the device. Safety lock must be disabled (see <b>SAFE</b> ).
<b>HELP</b>	Get a brief listing of available commands.
<b>SAFE = [0/1]</b>	Disable or enable the software safety lock. This lock protects critical internal values from accidental modification. The lock is always active after a reset. Use with caution.
<b>ROCOM = [0/1]</b>	Deactivate or activate robust communication mode. (Not implemented)

## › Error Codes

If a set command is completed successfully, the device will respond with **OK\n**, otherwise it will return an error, of the form **E[code]:[ch]\n**, where **[code]** is a two-digit hexadecimal error code which identifies the source of the error, and **[ch]** is a two-digit index of the affected output channel, or '**00**' if no channel is affected.

Table 3: Error code reference.

Error Code	Fault
<b>E00:[ch]</b>	Uncategorised error. May relate to channel <b>[ch]</b> if index > 0.
<b>E01:[ch]</b>	Over-voltage fault on channel <b>[ch]</b> . Power to channel is cut off.
<b>E02:[ch]</b>	Over-current fault on channel <b>[ch]</b> . Power to channel is cut off.
<b>E03:00</b>	Power cycling. Under normal circumstances, this error is returned when the device is powered down.
<b>E10:00</b>	Unrecognised command received.
<b>E11:00</b>	Unrecognised or invalid input parameter received.
<b>E12:[ch]</b>	Unrecognised channel specified.
<b>E13:[ch]</b>	Operation denied.
<b>E14:00</b>	There may be a problem with the NVM. If this persists, see Troubleshooting.

**Table 3:** Error code reference.

Error Code	Fault
<b>E90:00</b>	Information notice that the device powered up successfully. Added to the device log, but not transmitted.

## Binary control

Binary control can be challenging to implement, but greatly reduces communication time and latency. A one-to-one correspondence exists between the human-readable and binary control methods. Internally, each human-readable command is parsed into its binary counterpart; chains of modules communicate in binary. Binary data input avoids rounding errors, improving precision. Binary data output is not yet implemented.

### ► Data structure

Each binary command contains: a header byte, with information about the command; a command index, specifying which command should be executed; an address, to specify the target device or port; and one or more data blocks.

**Figure 1:** Binary data structure.



### ► Header

The header is 1 byte long. For a binary command, the first (most significant) bit is always 1.

# Programming Manual

Table 4: Header bits.

Bit number	Bit name	Hex mask	Bit function
0	BIN	0x80	Always 1; indicates binary command header
1	BCAST	0x40	Broadcast to all devices, before parse, execution
2	ALLCH	0x20	All channels should respond to command
3	ADDM	0x10	Addressing mode (unimplemented, set as 0)
4	RW	0x08	Read (1) or write (0) data
5	ACT	0x04	Act on a port without reading or writing data
6	DEXT	0x02	Data extension for vector commands
7	PBIT	0x01	Parity bit (even); prevents header data corruption

## › Command index

The command index is a single byte. Valid commands are listed in Table 4. Refer to Table 2 for full command descriptions.

Table 5: Command indices.

Index (decimal)	Index (hex)	ASCII equivalent	Allowed header modes
0	0x00	V	81, 82, 88, A0, A9
1	0x01	I	81, 82, 88, A0, A9
2	0x02	VMAX	81, 82, 88, A0, A9
3	0x03	IMAX	81, 82, 88, A0, A9
4	0x04	VCAL	81, 84, 88, A0, A9
5	0x05	ICAL	81, 84, 88, A0, A9
6	0x06	VERR	88, A9
7	0x07	IERR	88, A9
10	0x0A	VIP	A9
32	0x20	VFULL	A9
33	0x21	IFULL	A9
34	0x22	NCHAN	A9
35	0x23	FIRMWARE	88, A9

**Table 5:** Command indices.

Index (decimal)	Index (hex)	ASCII equivalent	Allowed header modes
36	0x24	<b>ID</b>	88, A9
37	0x25	<b>LIFETIME</b>	88, A9
38	0x26	<b>NVM</b>	A9, A0
39	0x27	<b>LOG</b>	88, A9
48	0x30	<b>ECHO</b>	81, 88, A9, A0
49	0x31	<b>LED</b>	81, 88, A9, A0
50	0x32	<b>NUP</b>	81, 88
51	0x33	<b>ADCT</b>	81, 88, A9, A0
52	0x34	<b>ADCN</b>	81, 88, A9, A0
53	0x35	<b>CCFN</b>	81, 88, A9, A0
54	0x36	<b>INTEST</b>	84
55	0x37	<b>OK</b>	81, 88, A9, A0
64	0x40	<b>RESET</b>	84
65	0x41	<b>HELP</b>	88
66	0x42	<b>SAFE</b>	81, 88, A9, A0
67	0x43	<b>ROCOM</b>	81, 88, A9, A0

### ‣ **Address**

The address determines which pin on which device (in the daisy-chain) is targeted by the command. It contains three bytes, with the most-significant byte transmitted first. Channel and Device ID values must be split into bytes for transmission. In the channel-wise addressing mode (ADDM = 0), two bytes are reserved for the channel number; in the device-wise addressing mode (ADDM = 1), 2 bytes are reserved for the device identifier, and 1 byte is reserved for the channel number. If all channels are being addressed (ALLCH = 1), the 3 address bytes must still be transmitted, but their values are ignored.

**Table 6:** Address bytes.

Byte 1	Byte 2	Byte 3	Setting
X	Channel high	Channel low	ADDM = 0
Device ID high	Device ID low	Channel low	ADDM = 1

# Programming Manual

Table 6: Address bytes.

Byte 1	Byte 2	Byte 3	Setting
X	X	X	ALLCH = 1

For example, to address channel 26 (hex 1A) in channel-wise addressing mode (ADDM = 0), the 3 address bytes are hex 00 00 1A.

## ‣ Data

Data is transmitted in 16-bit words, of 2 bytes each. Most commands accept just one word as input, but other commands can be *vectorised*, transmitting multiple words in the same command. Vectorised commands must have DEXT = 1.

For vectorised commands (DEXT = 1), the first data word is a count of the following data words. If there are  $N$  data words in total,  $N > 1$ , then the data block should take up  $2(N + 1)$  bytes, including the leading word count.

For normal commands (DEXT = 0), the two data bytes contain the high and low parts of the 16-bit data integer. For example, the ASCII command “V7=20” indicates that the voltage on channel 7 should be set to 20 V. On the Q8, VFULL = 20, so the requested voltage is the full-range voltage, and the corresponding data word is 0xFFFF. The voltage data word is calculated as:

$$(2^{16} - 1) V / VFULL =$$
$$0xFF00 \& \{(2^{16} - 1) V / VFULL\} / 256 + 0x00FF \& \{(2^{16} - 1) V / VFULL\}$$

*first data byte (MSB)* + *second data byte (LSB)*

A similar calculation applies to current setting commands, but  $V$  and  $VFULL$  are replaced by  $I$  and  $IFULL$ .

## Example ASCII-binary translations

Table 4 shows the correspondence between a selection of common ASCII commands with their binary equivalents, for Q8 example hardware (with  $VFULL = 20V$  and  $IFULL = 100 mA$ ).

Table 7: ASCII-binary command examples.

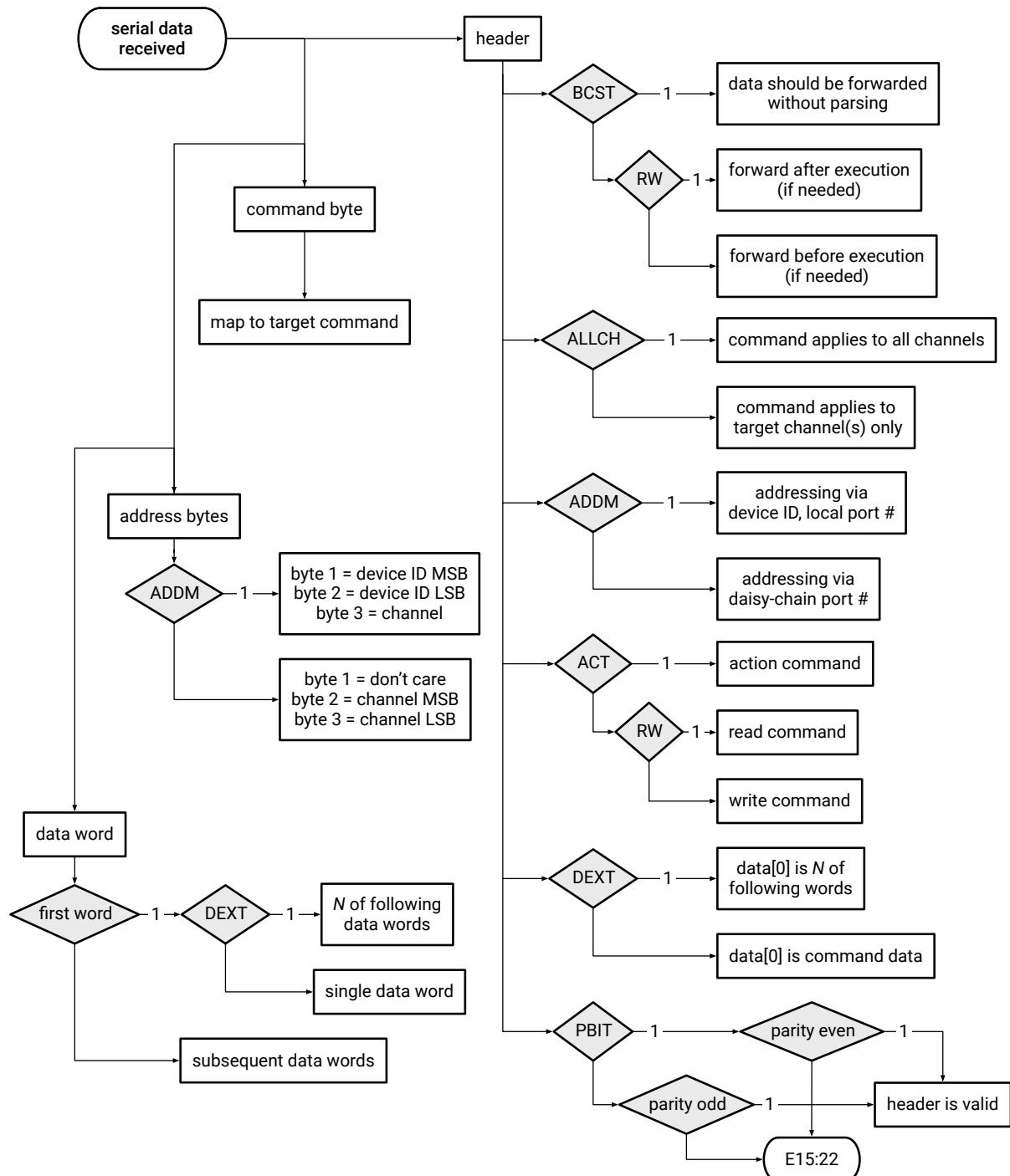
ASCII command		Binary command (hex)
V0 = 0	↔	81 00 000000 0000
V1 = 5.0	↔	81 00 000001 4000
V1?	↔	88 00 000001 0000
VALL = 5.0	↔	A0 00 FFFFFF 4000

**Table 7:** ASCII-binary command examples.

ASCII command	↔	Binary command (hex)
<b>VALL?</b>	↔	A9 00 FFFFFF 0000
<b>VMAX7 = 10.0</b>	↔	81 02 000007 8000
<b>VMAXALL = 10.0</b>	↔	A0 02 000007 8000
<b>VCAL18</b>	↔	84 04 000012 0000
<b>RESET</b>	↔	84 40 000000 0000
<b>NUP?</b>	↔	88 32 000000 0000
<b>LED = 1</b>	↔	81 31 000000 0001
<b>VVEC1 = 5.004, 5.009</b>	↔	82 00 000001 0002 4010 4020

# Programming Manual

## Command processing flowchart



## Notes and disclaimer

If you find an error in this manual, or have suggestions for how we could improve it, we would be very happy to hear from you. Contact us at [support@qontrol.co.uk](mailto:support@qontrol.co.uk) with your comments.

This manual is believed accurate at the time of writing. It is provided for information only, 'as is', and without guarantee of any kind. Qontrol Systems LLP accepts no liability for damage to equipment, hardware, or the customer application, or labour costs incurred due to application of information contained within this document.

## Revision History

Date	Hardware	Firmware	Notes	Author
Sept 2017	Q8 [0x06]	2.0.4	First revision.	JWS
Oct 2017	Q8 [0x06]	2.1.0	Second revision	JWS
Mar 2018	Q8 [0x06]	2.1.1	Clarified examples, bit order	JWS
Dec 2018	Q8 HW06, Q8iv HW05, Q8b HW01	2.1.1	Merged data sheet command reference and binary programming guide	JWS