830 Douglas Ave.
Dunedin, FL 34698
(727)733-2447
Fax:(727)733-3962
www.OceanOptics.com

A HALMA GROUP COMPANY

QE *Pro* Data Sheet

Intended Audience

This data sheet is intended for Ocean Optics customers looking for specific technical details for the QE *Pro* spectrometer. It contains information such as product specifications, a mechanical diagram, connector pinouts, QE *Pro* protocol messages, etc. For basic operation information, see the *QE Pro Installation and Operation Manual*. For more specific information on triggering, see *New External Triggering Options Instruction for Spectrometers with Firmware Version 3.0 and Above*.

Description

The Ocean Optics QE *Pro* Spectrometer is a scientific-grade spectrometer that is ideal for researchers and industrial customers. Its broadband sensitivity, from UV to NIR, makes it suitable for a wide range of applications, while its high sensitivity and thermoelectric cooler enable effective measurements at very low light levels. The QE *Pro* also has the highest dynamic range of any fiber optic CCD spectrometer in its class Onboard buffering and improved TEC performance are also features that help to set the QE *Pro* apart from other spectrometers.

The QE *Pro* interfaces to PCs, PLCs and other embedded controllers through USB 2.0 or RS-232 communications. The information included in this data sheet provides detailed instructions on the connection and operation of the QE *Pro*.

The detector used in the QE *Pro* spectrometer is a scientific-grade, back-thinned, TE Cooled, 1044x64 element CCD array from Hamamatsu (product number S7031-1006). For complete details on this detector, visit **www.Hamamatsu.com**.

Features

- Hamamatsu S7031-1006S Detector:
  - Typical dynamic range ~85,000:1
  - Peak QE: 90%
  - Back-thinned for enhanced sensitivity
  - Integration times from 8 ms to 60 minutes
  - Thermo Electric Cooled
- Scientific-grade Optical Bench:
  - Symmetrical Crossed Czerny Turner
  - 101mm focal length
  - F number: f/4
  - Interchangeable slits
  - 14 gratings (H1 – H14); HC1
  - 6 slit widths, plus no slit in SMA or FC bulkhead
- Communications
  - USB 2.0 Full Speed (12 Mbps)
  - RS232 up to 460K Baud
- Thermo Electric Cooler (TEC)
  - Software-controlled set-point
  - Software queries available for whether TEC is enabled, and TEC setpoint and stability
  - LED indicator to show when the TEC is stable and accurate
  - Temperature stability: <0.1ºC
  - Response time to reach setpoint:
  - Continuous TEC setpoint control from 40ºC below ambient up to 50 ºC
- GPIO
  - Single strobe
  - Continuous strobe
  - 10 user-programmable digital I/O pins
  - SPI/I2C for controlling peripherals
- Nonvolatile storage
  - Wavelength calibration coefficients
  - Linearity correction coefficients

    - Absolute irradiance calibration (optional)
  - Buffering
  - Triggering (4 modes)
  - Resets
    - Watchdog timer for reliability
    - Hardware power recycle via reset pin or software command
  - Kensington® security slot
  - LEDs for feedback on TEC readiness and general spectrometer health
  - Kinematic mounts used to position optical elements to increase accuracy and reliability
  - Software support:
    - OceanView
    - OmniDriver
    - SeaBreeze
- CE certification

Specifications

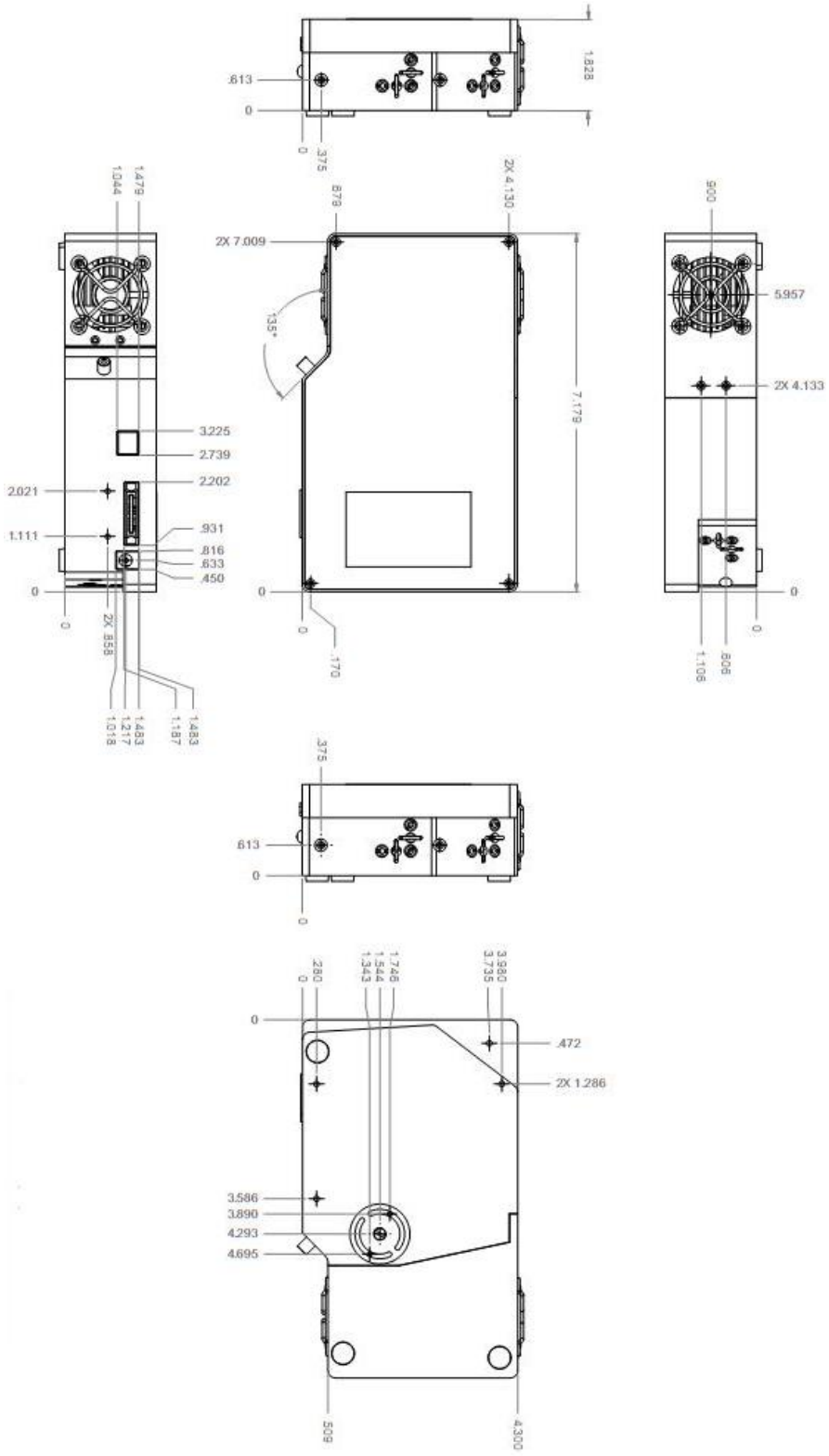
Specifications	Criteria
Performance:	
Integration Time	8 ms – 60 minutes
Dynamic Range	
Typical	~85000:1
Single Integration Period	85,000:1 (min)
100 Averages	850,000:1 (min)
10,000 Averages	8,500,000:1 (min)
Signal-to-Noise	
Single Integration Period	1,000:1 (typical)
Readout Noise	2.5 counts RMS (typical)
Stray light	<0.08% at 600 nm; 0.4% at 435 nm
Linearity	
Corrected	0.5% nonlinearity (max)
Spectrometer:	
Design	Symmetric crossed Czerny-Turner
F number	f/4
Input Fiber Connector	SMA 905 and Ocean Optics FC
Gratings	14 different gratings (H1 – H14); HC1 grating
Entrance Slit	5, 10, 25, 50, 100, or 200 µm slits. (or SMA/FC bulkhead with no slit)
Pixels	1024 active
Spectral range	185 – 1100nm available

Specifications	Criteria
Thermo-electric Cooler:	
Minimum Possible Delta[1]	
Above ambient	+20°C
Below ambient	-40°C
Temperature Limits[2]:	
Maximum	from -17 to 45 ºC in less than 10 seconds
Minimum	from 45 to -15 ºC in less than 10 seconds
Ramp Rate:	
Warming	+50°C
Cooling	-40° C
Temperature Stability	±0.1°C
Power:	
Supply voltage	4.5 – 5.5 V
Input/Output Logic	3.3 V CMOS
Power Consumption	
TEC On @ 40° below ambient	15W (Max)
TEC Off	2.5W (Max, Typical)
Physical Specifications:	
Physical Dimensions (LxWxH)	182 mm (7.17 in.) x 110 mm (4.33 in.) x 47 mm (1.85 in.)
Spectrometer Weight	1.15 kg (2.6 lbs.)
Power Supply Weight	0.45 kg (1 lb.)
Operating Temperature	0 to 50°C
Humidity	≤ 90% (noncondensing)
Regulatory Compliance	CE, FCC, UL (external power supply)

[1] This is the guaranteed range of the TEC input

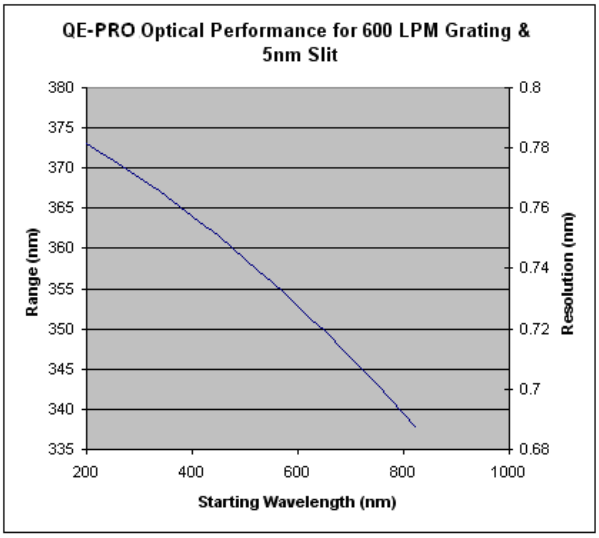
[2] Do not set your TEC setpoint to more than 20°C above ambient temperature

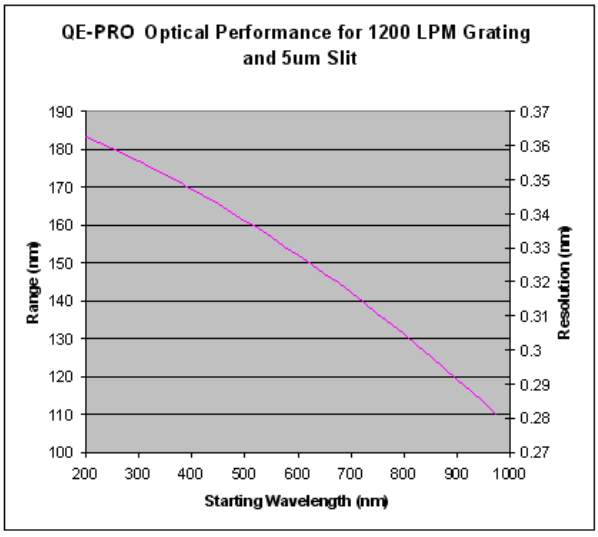
Mechanical Diagram

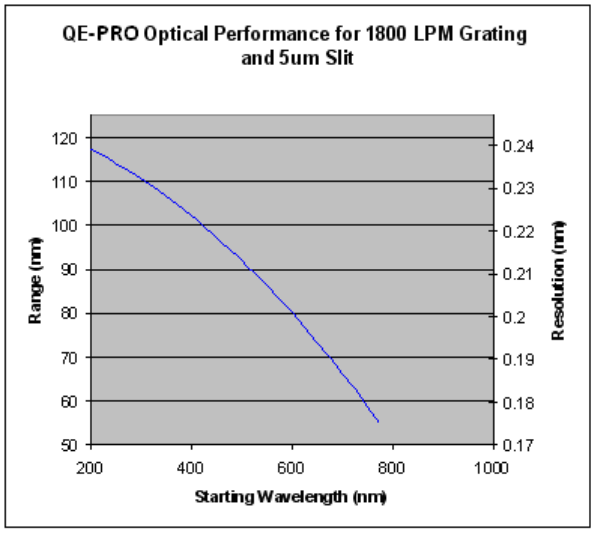


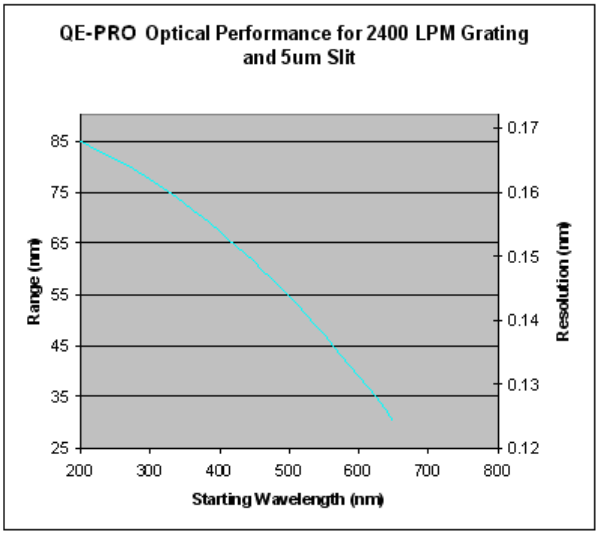
Optical Performance

Below are the graphs showing the range and resolution for the various gratings when configured with a 5µm slit for some grating options. For more information, please contact info@oceanoptics.com.









The following table shows the resolution for various slit sizes.

QE *Pro* Resolution

Slit							
		5	10	25	50	100	200
Pixel Resolution Factor							
Grating	Avg Range	2	2.2	2.6	3.3	4.6	8.9
300	750	1.46	1.61	1.90	2.42	3.37	6.52
600	360	0.70	0.77	0.91	1.16	1.62	3.13
1200	150	0.29	0.32	0.38	0.48	0.67	1.30
1800	90	0.18	0.19	0.23	0.29	0.40	0.78
2400	70	0.14	0.15	0.18	0.23	0.31	0.61

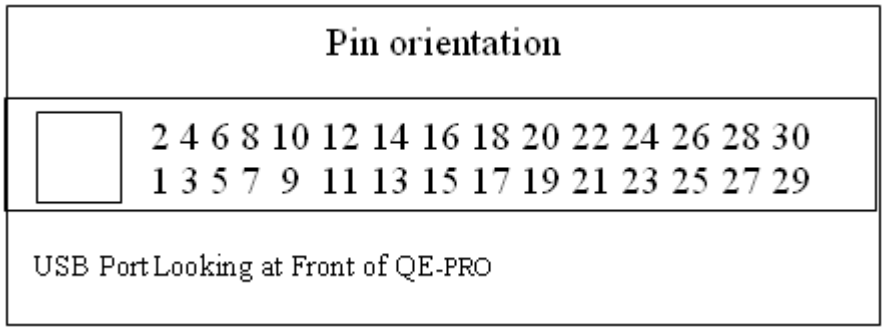
External Interface

The QE *Pro* includes an external connector that can be used to interface with external peripherals, such as light sources.

All digital signals on the external connector are 3.3V CMOS logic, except the I2C signals, which are open drain lines, internally pulled up to 3.3V. For protection, all signals have 33 ohm minimum output impedance and transient overvoltage suppression. Additionally, all signals except the I2C and SPI signals are pulled down to ground by 100K resistors.

Electrical Pinout

Listed below is the pin description for the QE *Pro* Accessory Connector (J3) located on the front vertical wall of the unit. The connector is a Pak50TM model from 3M Corp. Headed Connector Part# P50-030P1-RR1-TG. Mates with part# P50-030S-TGF (requires two: 1.27mm (50 mil) flat ribbon cable: Recommended 3M 3365 Series, HR4-CBL-DB15).



Pin #	Function	Input/Output	Description
1	RS232 Rx	Input	RS232 receive signal – Communicates with a PC over DB9 Pin 3
2	RS232 Tx	Output	RS232 transmit signal – Communicates with a PC over DB9 Pin 2
3	GPIO (2)	Input/Output	General Purpose Input Output
4	-	-	Unused
5	Ground	Input/Output	Ground
6	I2C SCL	Input/Output	I2C clock signal for communication to other I2C peripherals
7	GPIO (0)	Input/Output	General Purpose Input Output
8	I2C SDA	Input/Output	I2C data signal for communication to other I2C peripherals
9	GPIO (1)	Input/Output	General Purpose Input Output
10	Ext. Trigger In	Input	CMOS input trigger tolerant from 3-5V
11	GPIO (3)	Input/Output	General Purpose Input Output
12	VOUT	Output	Output power pin for QE *Pro*
13	SPI_MOSI	Output	SPI Master Out Slave In (MOSI) signal for communication to other SPI peripherals
14	VOUT	Output	Output power pin for QE *Pro*
15	SPI MISO	Input	SPI Master In Slave Out (MISO) signal for communication to the other SPI peripherals
16	GPIO (4)*	Input /Output	General Purpose Input Output
17	Single Strobe	Output	CMOS (3.3V) output pulse used as a strobe signal – Has a programmable delay relative to the beginning of the spectrometer integration period
18	GPIO (5)	Input/Output	General Purpose Input Output
19	SPI Clock	Output	SPI clock signal for communication to other SPI peripherals
20	Continuous Strobe	Output	CMOS output signal used to pulse a strobe – Divided down from the master clock signal
21	SPI CS	Output	External SPI chip select (active low)
22	GPIO (6)	Input/Output	General Purpose Input Output
23	RESET	Input	This pin is pulled up to 5V by a 10K internal resistor. Pull down to ground to reset. Leave open for normal operation.
24	RS-232 CTS	Output	RS-232 Clear to Send control logic signal – used to enable or suspend host transmission to the QE *Pro*
25	Lamp Enable	Output	CMOS signal driven Active HIGH when the Lamp Enable command is sent to the spectrometer
26	GPIO (7)	Input/Output	General Purpose Input Output

Pin #	Function	Input/Output	Description
27	RS-232 RTS	Input	RS-232 Request To Send control logic signal -- used to enable QE *Pro* transmission to the host
28	GPIO (8)	Input/Output	General Purpose Input Output
29	Ground	Input/Output	Ground
30	GPIO (9)	Input/Output	General Purpose Input Output

SPI

The QE *Pro* has the ability to function as a SPI master through the SPI port, which comprises the SPI-CS, SPI-MOSI, SPI_MISO, and SPI-CLK pins. To send messages over the SPI port, use the *SPI Full Duplex Transfer* message. The QE *Pro* does not send or receive any SPI data without direction from its host PC.

Because SPI is a full-duplex transaction, the *SPI Full Duplex Transfer* message both reads and writes at the same time. For instance, a four byte write will return four bytes of dummy read data, and a four byte read requires four bytes of dummy write data.

The maximum SPI clock rate can be configured with the *Set SPI Clock Limit* message. Chip select is active low and remains low for the entire transfer. MOSI and MISO should be sampled on rising clock edge and change on the falling clock edge.

Refer to Freescale's SPI format with SPO for more information.

I2C

The QE *Pro* has the ability to function as an I2C master through the I2C port, which comprises the I2C-SDA, and I2C-SCL pins. To send messages over the I2C port, use the *I2C Bus Write* and *I2C Bus Read* messages. The maximum I2C clock rate can be configured with the *Set I2C Clock Limit* message. Note that QE *Pro* does not send or receive any I2C data without direction from its host PC. The I2C lines are pulled up internally to 3.3V by 10K resistors.

GPIO

The QE *Pro* includes 10 user-programmable GPIO pins. These pins can be individually configured as either inputs or outputs by the *GPIO Set Output Enable Vector* message. When a pin is configured as an output, its value can be set through the *GPIO Set Value Vector* message. When a pin is configured as an input, its value can be read by the *GPIO Get Value Vector* message. Pins are configured as inputs (output is disabled) by default.

External Trigger

The external trigger input is a 3V to 5V CMOS signal.

Strobe / Lamp Outputs

Three signals are included to enable control of external lamps:

- **Lamp Enable** – This is a 3.3V signal that is used to turn lamps on or off. When lamp enable is low, lamps should be off. When lamp enable is high, continuous lamps should be on and strobed lamps should be responsive to continuous or single strobe inputs.

- **Continuous strobe** – This signal is used to continuously trigger pulsed lamps, such as the PX2. This signal is run off of an internal timer and is not synchronized to any activity inside the spectrometer. The shape of the continuous strobe pulse itself can be defined by software messages: *Set Continuous Strobe Period* and *Set Continuous Strobe Pulse Width* (not available in OceanView software). Continuous strobe must be enabled in software before it can be used.
- **Single strobe** – This signal is used to trigger light sources (or other equipment) that need to be synchronized to the beginning of the spectrometer's integration period. The single strobe signal goes high after a certain delay after the beginning of each integration period. Single strobe delay is referenced to the trigger in all external triggered modes. In the Normal/Continuous mode the single strobe delay is referenced to start of integration. The shape of the single strobe pulse is defined by software messages: *Set Single Strobe Delay* and *Set Single Strobe Pulse Width*. Single strobe must be enabled in software before it can be used.

Power Out

The QE *Pro* can supply power to external peripherals through the GPIO connector. The output voltage on these pins is connected to the input power of the QE *Pro* (5V), through a load switch that limits the output current. Due to this current limiting, peripherals are required to draw less than 100mA from this pin, and the peripheral should avoid drawing large transient currents, even during its power up sequence.

Reset

See *Resets* for more information about the external reset pin.

QE *Pro* TE Cooler

The Thermo Electric Cooler (TEC) is used for thermal noise reduction. At very low integration times, noise is primarily determined by "readout noise", which is constant across all integration times and detector temperatures. As integration times increase, thermal noise becomes dominant. However, a reduction in temperature from 25°C to -10°C may reduce thermal noise by almost a factor of 10. So, for low light applications that require long integration times, cooling the detector is critical.

The TEC can also be used to stabilize the detector temperature. This reduces baseline drift due to ambient temperature changes and self-heating.

Setting the TEC Temperature

When the instrument starts up, the TEC is enabled and the setpoint is set to -10°C by default. Users can disable the TEC or set a new setpoint using the messages listed in this document or via OceanView. The setpoint is set as an absolute temperature.

TEC LED and Is TEC Stable Message

For both the LED and the *Is TEC Stable* message, the TEC is considered to be stable when the following conditions are met:

- The TEC temperature is within 1°C of the setpoint

- Once the TEC has settled to within 0.1°C of its final value, the device waits for 10 more seconds and stores the settled value. The TEC is considered stable as long as the temperature is within 0.1°C of the settled value.

Precautions for Using the TEC

It is important to note that although the TEC setpoint is set as an absolute temperature, the TEC itself operates as a differential element; its ability to cool or heat is determined by its environmental conditions. As listed in the *Specifications*, the TEC is guaranteed to cool to down to 40°C below ambient or up to 20°C above ambient. For instance, at a 25°C ambient temperature, the TEC is guaranteed to be able to cool to -15°C or heat to 45°C. If the ambient temperature were to increase to 29°C, and the setpoint had been set to -15°C, then the device may not be able to reach the setpoint.

The TEC circuitry includes a thermal cutoff, in order to protect the detector from overheating. When the temperature of the TEC exceeds approximately 56°C, the TEC will shut off until the temperature has dropped below 51°C. The TEC LED will continue to be orange. The fan will start and stop intermittently.

Spectral Output

The QE *Pro* response to a *Get Buffered Spectra with Metadata* message includes more data than just the spectrum itself. The response begins with a metadata block that provides information about the spectrum that is being returned. The spectrum also includes both dummy and optical dark pixels that are located at the edges of the detector. Only the 1024 spectrum pixels represent valid spectral data. Pixels are 4 bytes each.

The format of the return is shown below:

Name	Length	Description
Meta Data	32 Bytes	See description of meta data later in this document
4 Dummy Pixels	16 Bytes	These pixels are not optically active. Use these pixels for electronic dark correction.
6 Optical Dark Pixels	24 Bytes	These pixels are optically active, but they are masked by the bevel. Do not use these pixels.
1024 Spectrum Pixels	4096 Bytes	These are the optically active pixels that constitute the spectrum.
6 Optical Dark Pixels	24 Bytes	These pixels are optically active, but they are masked by the bevel. Do not use these pixels.
4 Dummy Pixels	16 Bytes	These pixels are not optically active. Use these pixels for electronic dark correction.

Metadata

Name	Length	Description
Spec Count	Unsigned, 4 bytes	Increments every time a spectrum is digitized, regardless of whether it is kept.
Tick Count	Unsigned, 8 bytes	Taken at the time of acquisition.
Int Time	Unsigned, 4 bytes	In microseconds.
Reserved	2 bytes	
Trigger Mode	Unsigned, 4 byte	See *QE Pro Trigger Modes* for details.
Reserved	13 bytes	

Acquisition Control

Buffer

The QE *Pro* supports buffering.

- Maximum Buffer Time – 126 seconds @8 ms integration time
- Maximum Buffered Spectra – 15,698 spectra
- Maximum Buffered Pixels – 16,388,712 pixels

Abort/Enable Acquisition

The QE *Pro* supports an *Abort Acquisition* message that can be used to escape from long integration times or unfulfilled trigger modes. After writing the *Abort Acquisition* message to the device, the device will enter "idle mode," which means that it won't acquire spectra. To get the device out of idle mode, issue an Enable Acquisition – currently called *Acquire Spectra into Buffer* message.

Clear Buffer

The *Clear All Buffered Spectra* message clears the device's buffer.

Acquire Spectra Into Buffer Timing

The timing of the *Acquire Spectra into Buffer* message is important, especially for applications where conditions are dynamic. This message does not begin an integration time; it only retrieves existing data from a buffer. Control over when an integration time begins and ends is determined by the trigger mode settings.

When the PC issues an *Acquire Spectra into Buffer* message, one of the following conditions results:

Conditions	Result	Notes
The spectrometer is in idle mode.	NAK	Idle mode may happen if the user issues an *Abort Acquisition* message.
The spectrometer is not in idle mode and there is a spectrum in the buffer.	The spectrum is available in the endpoint buffer immediately.	
The spectrometer is not in idle mode but the buffer is empty.	The device waits until a spectrum is available, then transfers it to the endpoint buffer.	This may happen if the spectrometer is in a long integration time or if it is waiting for a trigger.

Be aware that the timing of acquiring spectra differs with the QE *Pro* due to its buffering capability. With other Ocean Optics spectrometers, a request to get the spectrum would retrieve the current spectrum in the process of completing. With the QE *Pro*, you retrieve the most recently completed spectrum.

Consider the following sequence of events:

1. Place the spectrometer into triggered mode and issue a *Clear All Buffered Spectra* message.
2. Issue a trigger.
3. Send an *Acquire Spectra into Buffer* message (select play/pause in the spectrometer operating software)
4. Issue a second trigger.

In this example, the *Acquire Spectra into Buffer* message returns immediately with the results from the first trigger, even if that trigger occurred an hour ago. This differs from other Ocean Optics spectrometers which wait until the *Acquire Spectra into Buffer* message is issued to "arm" the trigger and do not return until after the second trigger's subsequent integration period.

To arm the trigger appropriately for the QE *Pro* spectrometer, use the following procedure:

► *Procedure*

1. Abort Acquisition
2. Clear All Buffered Spectra
3. Set Trigger Mode
4. Get Buffered Spectra with Metadata

Note

In some cases (extremely short integration times or rapid multiple triggers) the request in Step 4 may not return the first spectrum that was acquired after Step 3.

QE *Pro* Trigger Modes

The QE *Pro* supports four trigger modes including the standard free running mode. These modes are described below. The following paragraphs describe these modes.

- **Normal (Free-run) Mode (0)**: In this mode, the spectrometer begins an integration period as soon as the previous integration period is complete.
- **External Hardware Level Trigger Mode (1)**: The spectrometer operates as in Normal mode while the trigger level is high.
- **External Synchronous Mode (2)**: In this mode, each rising edge of the trigger ends the previous integration period and begins the next. The period between rising edges must exceed the minimum integration time.
- **External Hardware Edge Trigger Mode (3):** In this mode, the device begins an integration period when it detects a rising edge of the trigger.

Image Skew

The QE *Pro* uses a two-dimensional detector instead of a linear detector. This improves dynamic range and sensitivity, but also introduces the effect of image skew into triggering and readouts. Image skew is a result of column binning that occurs after a trigger (or at the end of an integration period in free-running mode). During column binning, the columns of the two dimensional array are collapsed (binned) into a single row to prepare for a linear readout. Theoretically, during this period of column binning, half of the light will be added to the concluding integration period and half will be added to the integration period that is just beginning.

In practice, this balance may not be perfect, so it is best not to make any assumptions about whether light that is incident on the detector during column binning will show up on the previous or next integration period. In other words, the column binning period should be considered a period of indeterminate exposure. If there are events that are to be captured, such as light pulses, it is best to use timing offsets to ensure that these events happen outside of the column binning period.

Trigger Resolution

The column binning period lasts approximately 700 µsec and begins either due to the end of an integration period (in Normal or Level Trigger modes) or due to the rising edge of a trigger (in Hardware Edge Trigger or Synchronous modes).

The trigger delay has a resolution of one microsecond. However, the External Hardware Edge Trigger is sampled and detected at a higher rate. Jitter between the External Hardware Edge Trigger and the start of column binning (or trigger delay) is 40 ns.

Note

It is possible to turn off the trigger.

Trigger Parameters

In addition to the trigger mode, it is also possible to set an acquisition delay. The acquisition delay is the amount of time that the device will wait after detecting a trigger rising edge before registering the trigger and taking action. This can be important for synchronization with external systems.

Temperature Sensors

The QE *Pro* includes three separate temperature sensors. These can be queried by the *Read All Temperature Sensors* message.

Name	Description
Get TEC Temperature Or Read Temperature Sensor (3)	The temperature of the TEC, in degrees C. This is measured by a thermistor that is embedded in the detector itself. Due to nonlinearity in this thermistor, the accuracy of this reading deteriorates outside of the range from -20 to 40°C.
Read Temperature Sensor (2)	The temperature of the PCB, in °C. This is read by a temperature sensor IC that is mounted to the PCB. Due to heating from the electronics, the PCB temperature sensor generally reads a few degrees higher than ambient temperature.
Read Temperature Sensor (0)	The temperature of the microprocessor in °C. The microprocessor is one of the hottest components on the board, typically running about 15°C above ambient.

Resets

Watchdog Timer

The QE *Pro* utilizes an internal hardware-based timer that is maintained by the firmware. This timer is set to expire at a fixed interval that is reloaded by the firmware. If the firmware fails to reload the timer, the QE *Pro* will be forced into a reset condition causing the hardware to be power-cycled, ensuring a rapid recovery from any firmware fault that would otherwise leave the device unresponsive.

External Reset Pin

Pin 23 on the external connector is the external reset pin. This pin is pulled up internally by a 10K resistor to the input supply voltage, 5V. To reset the device, pull it down to GND. The device will remain in reset for at least 140ms after the pin has been released.

This pin can be also be used to hold the unit in shutdown mode. While the device is held in reset, it draws approximately 5mA. Most of this current draw is due to leakage in ESD protection circuits.

Reset Message

The message *0x000 000 00* causes the device to perform a hard reset. The device will remain in reset for at least 140ms after the message has been received.

Onboard Memory

The QE *Pro* stores information about its identification, programs, settings, and calibration information in its non-volatile memory. Users who will not be writing their own device drivers are encouraged to understand how the calibration-related parameters are used, in order to ensure measurements that are consistent with off-the-shelf performance.

Wavelength Calibration

The QE *Pro* outputs spectra as an array of pixel values. To translate from pixel index to actual wavelength, an equation is used to map pixel index to wavelength:

$$\lambda_p = I + C_1 p + C_2 p^2 + C_3 p^3$$

Where:

λ = the wavelength of pixel p

I = the wavelength of pixel 0

C_1 = the first coefficient (nm/pixel)

C_2 = the second coefficient (nm/pixel^2)

C_3 = the third coefficient (nm/pixel^3)

p = Pixel Number (starting at 0)

C_1 through C_3 are wavelength calibration constants stored in the QE *Pro*'s memory.

The wavelength calibration constants can be queried through the *Get Wavelength Coefficient* message. They can also be set manually through the *Set Wavelength Coefficient* message, although the procedure to determine the correct coefficients is outside of the scope of this document. The *QE Pro Installation and Operation Manual* contains the procedure for calibrating the wavelength of the QE *Pro*.

Nonlinearity Correction Calibration

The pixel intensity output of the QE *Pro* ranges from 0 to 200,000. Although mostly linear over that range, the native linearity does degrade slightly at the limits of its range. Therefore, applying the nonlinearity correction is strongly recommended.

The process for applying the nonlinearity correction to each pixel is:

$$L = D + (S-D) / (C_0 + C_1(S-D)^1 + C_2(S-D)^2 + \ldots + C_7(S-D)^7),$$

Where:

L =corrected pixel value,

D = dark pixel value,

S = raw pixel value, and

C0 through C7 are linearity constants that are stored in the QE *Pro*'s memory.

The nonlinearity calibration constants can be queried through the *Get Nonlinearity Coefficient* message. They can also be set manually through the *Set Nonlinearity Coefficient* message, although the procedure to determine the correct coefficients is outside of the scope of this document.

Stray Light Coefficients

Stray light coefficients are stored in the device's flash memory.

Irradiance Calibration

Irradiance calibration coefficients are stored in the device's flash memory.

QE *Pro* Communication and Interface

USB 2.0

The primary data interface between the QE *Pro* and a host computer is via USB. On the microprocessor, the interface is USB 2.0 Full Speed, which provides 12Mbit/s of bandwidth. The maximum update rate is ~ 100 Hz. The endpoints provided by the USB interface are divided up such that it is possible to request a spectrum and query the status of the device (or provide other messages) while waiting for the spectrum to be returned (see *Messages* for the USB message set).

USB Endpoints (any data query on either OUT will cause a response on the corresponding IN):

EP1 OUT ↔ EP1 IN

EP2 OUT ↔ EP2 IN

RS-232

Also known as serial port communication, RS-232 is a standard in PC and industrial device communications. Using transmit and receive signals this option allows the QE *Pro* to be a standalone device, which can output data to other logic devices/controllers such as a PLC or microcontroller.

QE *Pro* USB/RS-232 Port Interface Communications and Control Information

Overview

The QE *Pro* can communicate via the Universal Serial Bus (USB) or RS-232. This section contains the necessary message information for controlling the QE *Pro* via the USB or RS-232 interface. This information is only pertinent to users who wish to not utilize Ocean Optics drivers to interface to the QE *Pro*. Only experienced programmers should attempt to interface to the QE *Pro* via these methods.

Note

After start-up or reset, wait ~7 seconds before sending messages to the QE *Pro*.

USB Information

Ocean Optics Vendor ID number is 0x2457 and the Product ID is 0x4004.

Protocol Design

The binary message protocol for the QE *Pro* Spectrometer has the following design characteristics:

- Provides information so that the host does not need to know the state of the device to read the message.
- Contains a distinct header and footer to fully bracket transfers.
- Provides an abstract interface to the device. All timing is represented in standard units rather than clock divisors. A specific outcome is achieved via a single mechanism.
- Stores calibration information (wavelength, nonlinearity coefficients, etc.) in distinct messages rather than EEPROM slots.

QE Pro Message Protocol

There are two types of messages in this protocol:

- "commands" that do not return any information (except is requested)
- "queries" that cause the device to return information

When developing a device driver that will communicate with the QE *Pro*, the fact that some messages generate a response (including a status indication) and others do not can cause design problems. The simplest approach to creating a driver for this protocol is to have all message types generate a reply. This allows a single message read to be performed after every message write, and if the response indicates an error, then the driver can recover immediately rather than finding the error later when it expects a response to some new query.

The "flags" field in the message header (starting at byte offset 4) has an "acknowledgment (ACK) requested" bit (bit 2). If this bit is set to 1 for every message, then all communications with the QE *Pro* will become predictable read/write transactions. The immediate reply allows the host driver to avoid changing its state until it has received confirmation that the last operation succeeded or failed. This makes driver design much easier than the alternative.

It is recommended that a QE *Pro* protocol driver implement two functions:

- send_command_to_device() which takes a message type and an optional payload, and returns a simple pass/fail result based on the ACK or NACK flag in the response. This should set the "ACK requested" bit in every message it emits;
- query_device() which takes a message type and optional payload and returns a payload (e.g. a byte array) which can be NULL if the response was a NACK. Setting the "ACK requested" bit will not cause an extra response message; the ACK flag will be set in the normal response it requested.

By using these two functions to encapsulate all transfers to the QE *Pro*, the programming model is kept very simple.

Message Layout

All multi-byte fields are little-endian (LSB first). Each message in the binary protocol is laid out as follows:

1. A 44-byte header
2. An optional payload
3. A 16-byte checksum block
4. A four byte footer

The header, checksum, and footer are 64 bytes total. For simple messages, the command or response is embedded in the header so only a single packet is required. For more complex messages, the header and footer add a single USB packet as overhead to the transfer.

Header

The message header is structured as follows:

Offset (Bytes)	Field	Size (Bytes)	Valid Values	Notes
0	Start Bytes	2	0xC1, 0xC0	**Host:** When concatenated with the tail of a previous message, this field will create a distinct sequence.
2	Protocol Version	2	0x0000 – 0xFFFF	**Host:** Initially set to 0x1100. The host should only send messages known to be supported in the reported version of the protocol. The device may reject messages with a specified protocol it does not recognize.
4	Flags	2	0x0000 – 0xFFFF	The following bits represent what the device may include in the response and what the host may include in the request: **Bit 0 (Device):** response to earlier request (message type is set equal to request type). **Bit 1 (Device):** acknowledgment (ACK) if previous message included request for ACK. **Bit 2 (Host):** acknowledgment (ACK) requested. This can be used to cause a single reply to be generated for every command or query to the device; if the command would not normally generate a response, then a response will be created just to convey the ACK. Otherwise, the ACK bit will be set in the normally generated reply. Setting this bit is recommended as it simplifies driver development (though at a cost of bandwidth). **Bit 3 (Device):** negative acknowledgment (NACK). May be sent if previously sent message type is unknown or otherwise invalid. Message type and regarding fields will be set to the type that caused the error. Error Number field contains reason for NACK. **Bit 4 (Device):** exception occurred. Indicates that although the message itself was valid, the device encountered a hardware problem that may have invalidated the result. Error Number will be set to explain, if possible. **Bit 5 (Device):** The protocol version is deprecated. The device may communicate normally, but in future updates some changes may occur to how the device responds to certain messages. **Bit 6 (Device):** The message is deprecated. The device

Offset (Bytes)	Field	Size (Bytes)	Valid Values	Notes
				may communicate normally, but the message may be removed in future updates.
6	Error Number	2	0x0000 – 0xFFFF	**(Device):** Indicates whether the previous request was successful. Only set to be non-zero if at least one of the following flags is set: NACK or exception. Only one value can be set, even if multiple errors were detected. Application Values: **0**: Success (no detectable errors) **1**: Invalid/unsupported protocol **2**: Unknown message type **3**: Bad checksum **4**: Message too large **5**: Payload length does not match message type **6**: Payload data invalid **7**: Device not ready for given message type **8**: Unknown checksum type **9**: Device reset unexpectedly **10**: Too many buses. Messages have come from too many different bus interfaces. **11**: Out of memory. Failed to allocate enough space to complete the request. **12**: Message is valid, but requested information does not exist. **13**: Internal Error. May be unrecoverable. **14**: Message did not end properly **15:** Current scan interrupted Firmware Reprogramming Values: **100**: Could not decrypt properly **101**: Firmware layout invalid **102**: Data packet was wrong size (not 64 bytes) **103**: Hardware revision is not compatible with downloaded firmware. **104**: Existing flash map not compatible with downloaded firmware.
8	Message Type	4	0x00000000 – 0xFFFFFFFF	**Host:** Each message type represents a command. See *Message Types*.
12	Regarding	4	0x00000000 – 0xFFFFFFFF	**Host/Device:** Arbitrary host-defined data. Any response generated by the device will have the same value in its Regarding field. This can be used by the host to match responses to requests if transactions are split up.
16	Reserved	6		For future expansion.
22	Checksum Type	1	0x00 – 0x01	**Host:** Valid types: **0**: no checksum (must still provide a block of 16 bytes after the payload, but they can be zero).

Offset (Bytes)	Field	Size (Bytes)	Valid Values	Notes
				1: MD5 (fully fills the 16 byte checksum block)
23	Immediate Data Length	1	0x00 – 0x10	**Host/Device:** Total number of bytes used in the Immediate Data field (see below).
24	Immediate Data	16		**Host/Device:** Provides an alternative to specifying a payload so messages with small operands can fit within a single USB packet. If this field is used, the number of bytes containing valid data must be set in the Immediate Data Length field, and there is no payload. If a payload is used, this field is ignored.
40	Bytes Remaining	4	0x00000000 – 0xFFFFFFFF	**Host/Device:** Includes the payload, if any, plus the checksum and footer. This is included for buses like RS-232, such that a constant-sized header could be read (including this field) followed by another read for the remainder of the message. Payload length must be computed as this field minus the checksum and footer length. QE *Pro* may reject any message too long for it to process internally and return a NACK.

The header can be represented as a C struct as follows (assuming that the int type is 32 bits long):

```
struct ooi_binary_protocol_header {
    unsigned char start_bytes[2];     /* = { 0xC1, 0xC0 } */
    unsigned short protocol_version;  /* = 0x1100 */
    unsigned short flags;
    unsigned short error;
    unsigned int message_type;
    unsigned int regarding;
    unsigned char reserved[6];
    unsigned char checksum_type;
    unsigned char immediate_data_length;
    unsigned char immediate_data[16];
    unsigned int bytes_remaining;
};
```

Payload

After the standard header, a payload may be provided. The payload contains data required by the given message type. The format of the data within the payload depends on the message type. A payload is not required if operands will fit in the Immediate Data field of the header. The length of the payload must be computed from the Bytes Remaining field in the header, given that the checksum and footer are of a constant length:

Payload length = Bytes remaining – 20

Checksum

A 16-byte block must appear after the payload (if any) to contain checksum data. This block is required even if no checksum is used (according to the Checksum Type field). This protocol does not support checksums longer than 16 bytes, but the intent of the checksum is to detect bit errors. The checksum may not be necessary for USB but may be useful for buses that do not provide data integrity guarantees, such as RS-232.

If a checksum is used, it will be computed starting with the start byte of the header and continuing through the last byte of the payload. The length of the checksum and footer will not be included in the checksum (i.e., for MD5, which includes the total data length as an input).

Footer

After the checksum block, a 4-byte footer is provided. The footer has a constant value of 0xC5C4C3C2. This provides a distinct pattern when followed by a valid header (0xC1C0).

Message Types

The binary protocol divides up the 4.29 billion possible message types into categories and subcategories in a hierarchy. The most significant bits represent the more abstract categories, while the least significant bits represent subcategories and the messages. The 32-bit message type is split into three blocks, 0xXXX YYY ZZ, as follows:

XXX: top-level category or feature. 4096 of these may exist.

YYY: subcategories within the feature. 4096 of these may exist for each category.

ZZ: specific messages for the subcategory. 256 of these may exist for each subcategory.

The top-level categories (XXX) are initially defined as follows.

0x000: General device characteristics

0x001: Spectrometer feature (control of detector and ADC, pixel calibrations and corrections)

0x002: GPIO feature (configuration and control)

0x003: Strobe features (single and continuous strobe timing)

0x004: Temperature features (board temperature, thermo-electric cooling)

0x005: SPI feature

0x006: I^2C feature

The subcategories and messages for each of these categories are described in the tables that follow. Input and output data lengths that can be computed from the header (Bytes Remaining field) are not shown. All multi-byte integer types will be returned in little-endian format (least significant byte first).

Message Examples

The following is an example of how the Set Integration Time message type (0x001 100 10) can be constructed based on the information provided in this data sheet:

Header

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0xC1	0xC0	0x00	0x11	0x00	0x00	0x00	0x00
Start bytes		Protocol version		Flags		Error number	

Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
0x10	0x00	0x11	0x00	*x*	*x*	*x*	*x*
Message type (0x00110010)				Regarding (user-specified)			

Byte 16	Byte 17	Byte 18	Byte 19	Byte 20	Byte 21	Byte 22	Byte 23
						0x00	0x04
Reserved						Checksum type	Immediate length

Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	...	Byte 39
x LSB	*x*	*x*	*x* MSB	0	0	0
Integration time (immediate data)				Unused		

Byte 40	Byte 41	Byte 42	Byte 43

0x14	0	0	0
Bytes remaining			

Optional Payload	Byte 44…Byte 59	Byte 60	Byte 61	Byte 62	Byte 63
Not used for this message	Checksum	0xC5	0xC4	0xC3	0xC2
		Footer			

The following is an example of how the Get Buffered Spectrum with Metadata message type (0x001 009 28) can be constructed based on the information provided in this data sheet:

Request

- Unless otherwise noted, all fields are little-endian
- Acknowledgment not requested as this message is a query, not a command
- Message Opcode: 0x001 009 28
- No checksum was used in this example

Byte 00	Byte 01	Byte 02	Byte 03	Byte 04	Byte 05	Byte 06	Byte 07	Byte 08	Byte 09	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
Start Bytes (not little-endian)		Protocol Version		Flags		Error Number		Message Opcode				Regarding (user-specified)			
0xC1	0xC0	0x00	0x11	0x00	0x00	0x00	0x00	0x28	0x09	0x10	0x00	0x00	0x00	0x00	0x00

Byte 16	Byte 17	Byte 18	Byte 19	Byte 20	Byte 21	Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31
Reserved						Checksum Type	Immediate Data Length	Immediate Data ….							
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Byte 32	Byte 33	Byte 34	Byte 35	Byte 36	Byte 37	Byte 38	Byte 39	Byte 40	Byte 41	Byte 42	Byte 43	Byte 44	Byte 45	Byte 46	Byte 47
…. Immediate Data								Bytes Remaining				Checksum …. (not little-endian)			
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x14	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Byte 48	Byte 49	Byte 50	Byte 51	Byte 52	Byte 53	Byte 54	Byte 55	Byte 56	Byte 57	Byte 58	Byte 59	Byte 60	Byte 61	Byte 62	Byte 63
…. Checksum (not little-endian)												Footer (not little-endian)			
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0xC5	0xC4	0xC3	0xC2

Response

- Unless otherwise noted, all fields are little-endian
- Flags field (bit 0) shows a response to an earlier request
- Error Number field shows a positive acknowledgment (no error)
- Payload: 4208 bytes (Metadata + Pixel Data)
  - Metadata: 32 bytes
    - Spectrum Count (unsigned, 4 bytes)
    - Tick Count (unsigned, 8 bytes)
    - Integration Time (unsigned, 4 bytes)
    - Reserved (2 bytes)
    - Trigger Mode (unsigned, 1 byte)
    - Reserved (13 bytes)
  - Pixel Data: 4176 bytes
    - 1044 pixels
    - each pixel (unsigned, 4 bytes)
      - bits 31-18 (unused)
      - bits 17-0 (valid pixel information)

Byte 00	Byte 01	Byte 02	Byte 03	Byte 04	Byte 05	Byte 06	Byte 07	Byte 08	Byte 09	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
Start Bytes (not little-endian)		Protocol Version		Flags		Error Number		Message Opcode				Regarding (user-specified)			
0xC1	0xC0	0x00	0x11	0x01	0x00	0x00	0x00	0x28	0x09	0x10	0x00	0x00	0x00	0x00	0x00

Byte 16	Byte 17	Byte 18	Byte 19	Byte 20	Byte 21	Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31
Reserved						Checksum Type	Immediate Data Length	Immediate Data ….							
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Byte 32	Byte 33	Byte 34	Byte 35	Byte 36	Byte 37	Byte 38	Byte 39	Byte 40	Byte 41	Byte 42	Byte 43	Byte 44	Byte 45	Byte 46	Byte 47
…. Immediate Data								Bytes Remaining				Payload ….			
												Metadata ….			
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x84	0x10	0x00	0x00	0xXX	0xXX	0xXX	0xXX

Byte 48	Byte 49	Byte 50	Byte 51	Byte 52	Byte 53	Byte 54	Byte 55	Byte 56	Byte 57	Byte 58	Byte 59	Byte 60	Byte 61	Byte 62	Byte 63
…. Payload ….															
…. Metadata ….															
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

Byte 64	Byte 65	Byte 66	Byte 67	Byte 68	Byte 69	Byte 70	Byte 71	Byte 72	Byte 73	Byte 74	Byte 75	Byte 76	Byte 77	Byte 77	Byte 78
…. Payload ….															
…. Metadata												Pixel Data ….			
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

Byte 79	Byte 80	Byte 81	Byte 82	Byte 83	Byte 84	Byte 85	…. ….	Byte 4233	Byte 4234	Byte 4235	Byte 4236	Byte 4237	Byte 4238	Byte 4239
…. Payload ….							…. ….	…. Payload ….						
…. Pixel Data ….							…. ….	…. Pixel Data ….						
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	…. ….	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

Byte 4240	Byte 4241	Byte 4242	Byte 4243	Byte 4244	Byte 4245	Byte 4246	Byte 4247	Byte 4248	Byte 4249	Byte 4250	Byte 4251	Byte 4252	Byte 4253	Byte 4254	Byte 4255
…. Payload												Checksum …. (not little-endian)			
…. Pixel Data															
0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0xXX	0x00	0x00	0x00	0x00

Byte 4256	Byte 4257	Byte 4258	Byte 4259	Byte 4260	Byte 4261	Byte 4262	Byte 4263	Byte 4264	Byte 4265	Byte 4266	Byte 4267	Byte 4268	Byte 4269	Byte 4270	Byte 4271
…. Checksum (not little-endian)												Footer (not little-endian)			
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0xC5	0xC4	0xC3	0xC2

Messages

General Device Messages

Message Type	Purpose	Input Data	Output Data	Notes
Reset Messages				
0x000 000 00	Reset	N/A	N/A	Forces a reset of the device. Wait approximately 7 seconds before restarting communications.
0x000 000 01	Reset and clear settings			Forces a reset and restores factory settings. Erases RS-232 parameters.
Revision Information Messages				
0x000 000 80	Get hardware revision	N/A	Unsigned byte	This value is sensed from the hardware itself. May be used to verify compatibility of firmware before reprogramming and for diagnostics. Input has no payload. Output is 2-digit binary coded decimal.
0x000 000 90	Get host interface firmware revision	N/A	Unsigned short	Firmware version as binary coded decimal. The same value should be available through the USB descriptor as the bcdDevice field. Input has no payload. Output is a 4-digit binary coded decimal (LSB first).
0x000 000 91	Get FPGA firmware revision	N/A	Unsigned short	Firmware version as 4-digit binary coded decimal (LSB first) for FPGA.
Device Identification Messages				
0x000 001 00	Get serial number	N/A	String	Device serial number. The string used to enumerate a USB connection is limited to 9 bytes.
0x000 001 01	Get serial number length	N/A	Unsigned byte	Output is maximum length of serial number in bytes.
0x000 002 00	Get device alias	N/A	String	User-defined name for the device (e.g., station number)
0x000 002 01	Get device alias length	N/A	Unsigned byte	Output is maximum length of alias in bytes
0x000 002 10	Set device alias	String	N/A	If string length is 0, alias will be deleted
User Storage Messages				
0x000 003 00	Get number of available user strings	N/A	Unsigned byte	User-defined strings for storing small amounts of arbitrary data (count)

Message Type	Purpose	Input Data	Output Data	Notes
0x000 003 01	Get user string length	N/A	Unsigned short	Output is maximum length in bytes for each user string
0x000 003 02	Get user string	Unsigned byte	String	Input is a string index
0x000 003 10	Set user string	Unsigned byte, String	N/A	Input is string index followed by data. If string data is of zero length, user string will be deleted.
RS232 Configuration Messages				
0x000 008 00	Get RS-232 baud rate	N/A	Unsigned 32-bit integer	Returned value is actual baud rate as an integer, e.g. 115200.
0x000 008 04	Get RS-232 flow control mode	N/A	Unsigned byte	Byte value is hardware flow control mode: 0 = none 1 = CTS/RTS flow control
0x000 008 10	Set RS-232 baud rate	Unsigned 32-bit integer	N/A	Target baud rate as an integer, e.g. 115200.
0x000 008 14	Set RS-232 flow control mode	Unsigned byte	N/A	Byte value is hardware flow control mode: 0 = none 1 = CTS/RTS flow control
0x000 008 F0	Save current RS-232 settings	N/A	N/A	Stores current settings as new power-on defaults. It is unlikely that a strictly RS232 host would be able to save default parameters that would prevent it from communicating with the QE *Pro*. If it can correctly send this message to save the settings, then it has proven that it can communicate with these settings.
Status LED Messages				
0x000 010 10	Set user LED pattern	Unsigned byte, Unsigned byte (optional)	N/A	If no payload is given, LED will reset to default behavior. If two bytes are provided, the first must be zero. LED behavior may be overridden by other blink patterns based on relative priorities. Controls only the top LED. Argument 1 LED index: LED index = 0 Argument 2 Mode: 0 = Default behavior 1 = High priority. LED will blink 3 long, 3 short, and repeat. High-priority pattern. 2 = Low priority. LED will fade in intensity up and down.

Message Type	Purpose	Input Data	Output Data	Notes
Firmware Reprogramming Messages				
0x000 FFF 00	Put device in reprogramming mode	N/A	N/A	After receiving this message, the device will transition out of its operational state into a reprogramming state. After reprogramming, the device will reset. Only to be used when reprogramming the microcontroller, not the FPGA.

Spectrometer Messages

Message Type	Purpose	Input Data	Output Data	Notes
General Messages				
0x001 000 00	Abort acquisition	N/A	N/A	If spectrometer is acquiring, causes it to dump the ongoing scan and transition to idle state in anticipation of a new request. The acquisition must be restarted before requesting spectra.
Buffering Messages				
0x001 008 20	Get maximum buffer size	N/A	Unsigned 32-bit integer	Maximum number of spectra that can be stored due to hardware limits. Provides an upper bound for configuring buffer depth.
0x001 008 22	Get current buffer size	N/A	Unsigned 32-bit integer	Programmable limit on number of spectra that can be stored at once.
0x001 008 30	Clear all buffered spectra	N/A	N/A	Causes scan buffer to be purged of any accumulated spectra.
0x001 008 31	Remove oldest spectra	Unsigned 32-bit integer	N/A	Causes the given number of spectra to be dumped from the buffer, starting with the oldest. Specifying a value larger than the current buffer length causes the entire buffer to be cleared.
0x001 008 32	Set buffer size: active	Unsigned 32-bit integer	N/A	Specifies the maximum number of spectra that may be buffered before an overflow condition occurs. May not be set larger than hardware buffer size limit. Minimum size is 1. Causes the entire buffer to be cleared.

Message Type	Purpose	Input Data	Output Data	Notes
Spectral Acquisition Control Messages				
0x001 009 00	Get number of spectra in buffer	N/A	Unsigned 32-bit integer	Number of spectra currently stored in data buffer
0x001 009 02	Acquire spectra into buffer	N/A	N/A	Prompts the device to begin acquiring and correcting spectra according to buffer settings, processing settings and trigger mode.
0x001 009 08	Query whether the device is idle	N/A	Unsigned byte	Returns 1 if the device is idle, otherwise 0. This can be used to determine if the device is acquiring a spectrum.
Buffered Spectrum Retrieval Messages				
0x001 009 28	Get buffered spectra with metadata (32 bits per pixel)	N/A	Metadata and spectral data (LSB, …, MSB)	Output is 4208 bytes. It consists of a 32-byte metadata block and a 4176-byte spectral data (32 bits/pixel) block. If no spectra are available but device is acquiring, readout will block until next scan is finished. **Metadata block**: 1. (Unsigned 32-bit integer) Spec Count: Increments every time a spectrum is digitized, regardless of whether it is kept. 2. (Unsigned, 8 bytes) Tick Count: Taken at time of acquisition. 3. (Unsigned 32-bit integer) Int Time: In micro-seconds. 4. (Unsigned, 1 byte) Reserved 5. (Unsigned, 1 byte) Reserved 6. (Unsigned, 1 byte) See *QE Pro Trigger Modes*. 7. (Unsigned, 1 byte) Reserved 8. (Unsigned 32-bit integer) Reserved 9. (8 bytes) Reserved **Spectral Data block:** 1. 1044 pixels 2. 32 bits (4 bytes) per pixel 3. relevant bits: 0-17

Message Type	Purpose	Input Data	Output Data	Notes
Integration Time Messages				
0x001 100 00	Get integration time (in microseconds)	N/A	Unsigned 32-bit integer	
0x001 100 01	Get integration time: Minimum	N/A	Unsigned 32-bit integer	Minimum integration time = 8,000 µsec
0x001 100 02	Get integration time: Maximum	N/A	Unsigned 32-bit integer	Maximum integration time = 3,600,000,000 µsec (60 minutes)
0x001 100 03	Get integration time: Increment	N/A	Unsigned 32-bit integer	Integration increment = 1 µsec
0x001 100 10	Set integration time (µs)	Unsigned 32-bit integer value	N/A	
Triggering Messages				
0x001 101 00	Get trigger mode	N/A	Unsigned byte	Valid trigger modes[1]
0x001 101 10	Set trigger mode	Unsigned byte	N/A	Valid trigger modes[1]
[1]Trigger modes: Mode 0 (default): Normal. Integration begins as soon as possible after request. Mode 1: Level trigger. Integration begins with rising edge and continues to restart as long as the level is held high. Mode 2: Synchronization. Each trigger pulse ends the previous integration period and starts the next. Pulses must not be closer together than minimum allowed integration time. Mode 3: Edge trigger. Integration begins with rising edge. Edges occurring while device is integrating may be ignored.				
Other Acquisition Parameter Messages				
0x001 104 00	Get lamp enable	N/A	Unsigned byte	Flag: 0: disable 1: enable
0x001 104 10	Set lamp enable	Unsigned byte	N/A	Refers to the external enable pin. Changes take effect at the beginning of the next spectral acquisition. If unsynchronized control is required, connect to a GPIO instead. Flag: 0: disable 1: enable

Message Type	Purpose	Input Data	Output Data	Notes
0x001 105 00	Get Acquisition Delay	N/A	Unsigned 32-bit integer	In microseconds
0x001 105 01	Get Acquisition Delay: Minimum	N/A	Unsigned 32-bit integer	Minimum delay is 0 µsec
0x001 105 02	Get Acquisition Delay: Maximum	N/A	Unsigned 32-bit integer	Maximum delay is 1,360 µsec
0x001 105 03	Get Acquisition Delay: Increment	N/A	Unsigned 32-bit integer	Acquisition delay increment is 1 µsec
0x001 105 10	Set Acquisition Delay	Unsigned 32-bit integer	N/A	In microseconds
Wavelength Calibration Messages				
0x001 801 00	Get number of wavelength coefficients	N/A	Unsigned byte	
0x001 801 01	Get wavelength coefficient	Unsigned byte	Single-precision floating point	Input is the order of the coefficient to retrieve. Calibration only refers to the pixels that will be returned when a corrected spectrum is provided.
0x001 801 11	Set wavelength coefficient	Unsigned byte, single-precision floating point	N/A	Input is the order of the coefficient to set (indexing starts with wavelength intercept at index 0), followed by the coefficient (IEEE single-precision float). Calibration only refers to pixels that will be returned when a corrected spectrum is provided.
Nonlinearity Correction Calibration Messages				
0x001 811 00	Get nonlinearity coefficient count	N/A	Unsigned byte	Output has 1-byte output data with the number of coefficients.
0x001 811 01	Get nonlinearity coefficient	Unsigned byte	Single-precision floating point	Input has 1-byte input for coefficient index to retrieve. Output has 4-byte floating-point, single-precision coefficient.

Message Type	Purpose	Input Data	Output Data	Notes
0x001 811 11	Set nonlinearity coefficient	Unsigned byte, single-precision floating point	N/A	Input is the order of the coefficient to set, followed by an IEEE single-precision, floating-point coefficient.
Irradiance Calibration Messages				
0x001 820 01	Get irradiance factors for all pixels	N/A	Single-precision floating point	Output is µJ/count (4176 bytes). **Data:** 1. 1044 pixels 2. 32 bits (4 bytes) per pixel 3. IEEE single precision, floating point
0x001 820 02	Get number of calibration factors	N/A	Unsigned 32-bit integer	Determines the size of a buffer to create before reading back calibration.
0x001 820 03	Get irradiance collection area	N/A	Single-precision floating point	Retrieves the collection area (in units of cm^2) associated with the irradiance calibration factors. If a collection area has not been set, it returns a NACK.
0x001 820 11	Set irradiance factors for all pixels	Single-precision floating point	N/A	Output is µJ/count (4176 bytes). **Data:** 1. 1044 pixels 2. 32 bits (4 bytes) per pixel 3. IEEE single precision, floating point
0x001 820 13	Set irradiance collection area	Single-precision floating point	N/A	Sets the collection area (in units of cm^2) associated with the irradiance calibration factors.
Stray Light Coefficients				
0x001 831 00	Get number of stray light coefficients	N/A	Unsigned byte	
0x001 831 01	Get stray light coefficient	Unsigned byte	Single-precision floating point	Input is the order of the coefficient to retrieve.
0x001 831 11	Set stray light coefficient	Unsigned byte, Single-precision floating point	N/A	Input is the order of the coefficient to set, followed by an IEEE single-precision, floating-point coefficient.
Bench Information Messages				
0x001 B02 00	Get slit width	N/A	Unsigned short	Represents a value given by 10^{-6} m.
0x001 B04 00	Get bench grating description	N/A	String	Returns the name or category of the grating.

Message Type	Purpose	Input Data	Output Data	Notes
0x001 B05 00	Get bench filter description	N/A	String	Returns the description of filter, if any.
0x001 B07 00	Get detector serial number	N/A	String	Returns the serial number for the detector.

GPIO Messages

The "mask" bits (8-15) are associated with the "value" bits 0-7. To set/clear individual "value" bits, the corresponding "mask" bits must be set to 1.

Setting the Output Enable Vector:

GPIO pin direction: 0=input, 1=output

Setting the Value Vector:

GPIO output pins: 0=logic low, 1=logic high

Message Type	Purpose	Input Data	Output Data	Notes
0x002 000 00	Get number of GPIO pins	N/A	Unsigned byte	Output is the I/O pin count.
0x002 001 00	Get output enable vector	N/A	Unsigned short	Output is the data direction definition.
0x002 001 10	Set output enable vector	Unsigned long, unsigned long	N/A	Argument 1: bit vector value Argument 2: bit mask.
0x002 003 00	Get value vector	N/A	Unsigned short	Bits set to 1 correspond to pins set as outputs.
0x002 003 10	Set value vector	Unsigned long, unsigned long	N/A	Argument 1: bit vector value Argument 2: bit mask. Bits set to 1 correspond to pins being driven to a logic high.

Strobe Messages

Message Type	Type/Purpose	Input Data	Output Data	Notes
Single Strobe				
0x003 000 00	Get single-strobe pulse delay (µs)	N/A	Unsigned 32-bit integer	In microseconds
0x003 000 01	Get single-strobe pulse width (µs)	N/A	Unsigned 32-bit integer	In microseconds
0x003 000 02	Get single-strobe enable	N/A	Unsigned byte	0: disabled 1: enabled

Message Type	Type/Purpose	Input Data	Output Data	Notes
0x003 000 03	Get single-strobe pulse delay minimum	N/A	Unsigned 32-bit integer	In microseconds
0x003 000 04	Get single-strobe pulse delay maximum	N/A	Unsigned 32-bit integer	In microseconds
0x003 000 05	Get single-strobe pulse delay increment	N/A	Unsigned 32-bit integer	In microseconds
0x003 000 10	Set single-strobe pulse delay	Unsigned 32-bit integer	N/A	In microseconds
0x003 000 11	Set single-strobe pulse width	Unsigned 32-bit integer	N/A	In microseconds
0x003 000 12	Set single-strobe enable	Unsigned 32-bit integer	N/A	0: disabled 1: enabled
Continuous Strobe Messages				
0x003 100 00	Get continuous-strobe period	N/A	Unsigned 32-bit integer	In microseconds
0x003 100 01	Get continuous-strobe enable	N/A	Unsigned byte	0: disabled 1: enabled
0x003 100 02	Get continuous-strobe period minimum	N/A	Unsigned 32-bit integer	In microseconds
0x003 100 03	Get continuous-strobe period maximum	N/A	Unsigned 32-bit integer	In microseconds
0x003 100 04	Get continuous-strobe period increment	N/A	Unsigned 32-bit integer	In microseconds
0x003 100 05	Get continuous-strobe width	N/A	Unsigned 32-bit integer	In microseconds
0x003 100 10	Set continuous-strobe period	Unsigned 32-bit integer	N/A	In microseconds

Message Type	Type/Purpose	Input Data	Output Data	Notes
0x003 100 11	Set continuous-strobe enable	Unsigned byte	N/A	0: disabled 1: enabled
0x003 100 15	Set continuous-strobe width	Unsigned 32-bit integer	N/A	In microseconds

Temperature Messages

Notes

The microcontroller sensor will report values much higher than the detector board thermistor because the microcontroller integrated circuit runs at a higher temperature.

The QE *Pro* contains three memory locations for the temperature sensor as follows:

0 = Microcontroller Temperature Sensor

1 = Reserved/Internal Use

2 = Main Board Temperature Sensor

3 = Detector Thermistor

Message Type	Purpose	Input Data	Output Data	Notes
0x004 000 00	Get temperature sensor count	N/A	Unsigned byte	Provides the number of temperature sensors available.
0x004 000 01	Read temperature sensor	Unsigned, 1 byte	Single-precision floating point	Provides the temperature in °C of the indexed sensor. As an alternate, the **Get TEC Temperature** message can be used to retrieve a reading from the detector thermistor. 0: µP on-chip temp sensor 1: Reserved 2: main board temp sensor 3: detector thermistor
0x004 000 02	Read all temperature sensors	N/A	Single-precision floating point	All temperature readings are returned as an array.
TEC Messages				
0x004 200 00	Get TEC enable	N/A	Unsigned byte	Queries whether thermo electric cooler attached to detector is enabled: 0 : disabled 1 : enabled

Message Type	Purpose	Input Data	Output Data	Notes
0x004 200 01	Get TEC setpoint	N/A	Single-precision floating point	Queries the setpoint of the TEC (in °C).
0x004 200 03	Is TEC stable	N/A	Unsigned byte	Queries whether the TEC temperature has reached a stable setpoint or whether it is still changing. 0: not stable 1: stable
0x004 200 04	Get TEC temperature	N/A	Single-precision floating point	Provides the temperature (in °C) of the detector thermistor. As an alternate, the Read Temperature Sensor message can be used by including the input argument of sensor index 3.
0x004 200 10	Set TEC enable	Unsigned byte	N/A	Enables/disables thermo-electric cooler attached to detector. 0: disabled 1: enabled
0x004 200 11	Set TEC setpoint	Single-precision floating point	N/A	Specifies the setpoint (in °C) of the TEC.

SPI Messages

Message Type	Purpose	Input Data	Output Data	Notes
0x005 000 00	Get number of SPI buses	N/A	Unsigned byte	Output is number of available SPI buses. Values less than this can be used as the bus index for duplex transfers.
0x005 000 01	Get number of SPI chip selects for selected bus	Unsigned byte	Unsigned byte	Input is index of SPI bus to query. Output is number of chip selects available for that bus. 0: external bus
0x005 000 10	SPI full-duplex transfer	Unsigned byte, Unsigned byte, string	String	Transfers the input data through the external SPI bus. Input is bus index, then chip select index, then transmit buffer. The length of the returned data stream is equal to that of the transmit buffer. Argument 1 : Bus index Argument 2: chip select Argument 3: byte stream

Message Type	Purpose	Input Data	Output Data	Notes
0x005 000 90	Set SPI clock limit	Unsigned byte, Unsigned 32-bit integer	N/A	Clock limit is maximum allowed clock rate in Hz to be used on subsequent transfer operations. QE *Pro* will compute closest available rate not exceeding this amount. Input is bus index followed by rate in Hz. Argument 1: Bus index 0 = external Argument 2: Rate (in Hertz)

I²C Messages

Message Type	Purpose	Input Data	Output Data	Notes
0x006 000 00	Get number of I²C buses	N/A	Unsigned byte	Output is number of available I2C buses. Values less than this can be used as the bus index in the read and write message types.
0x006 000 10	I²C bus read	Unsigned byte, Unsigned byte, Unsigned short	String	Input is bus index (0), then device address (0-127), followed by number of bytes to read. Output is bytes actually read.
0x006 000 20	I²C bus write	Unsigned byte, Unsigned byte, String	Unsigned short	Input is bus index (0), then device address (0-127), followed by bytes to write. Output is number of bytes written.
0x006 000 90	Set I²C clock limit	Unsigned byte, Unsigned 32-bit integer	N/A	Clock limit is maximum allowed clock rate in Hz to be used on subsequent read/write operations. QE *Pro* will compute closest available rate not exceeding this amount. Input is bus index (0) followed by rate (in Hz).