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

A HALMA GROUP COMPANY

STS Data Sheet

Description

This document addresses the latest protocol version (0x1100). Previous versions will be deprecated.

The Ocean Optics STS Spectrometer is a family of compact, low-cost, highly reproducible (unit-to-unit) instruments ideal for embedding into OEM devices. STS includes the linear ELIS1024 detector in a footprint that is less than 50 mm (2") square optical bench, plus all the circuits necessary for spectrometer operation.

The STS provides full spectral analysis with low stray light, high signal-to-noise ratio, and optical resolution of ~1.5 nm (FWHM) and is especially useful for high-intensity applications such as LED characterization and absorbance/transmission/reflection measurements. Available STS models include:

- STS-VIS Microspectrometer, 350 – 800 nm
- STS-NIR Microspectrometer, 650 – 1100 nm
- STS-UV Microspectrometer, 190 – 650 nm

The STS spectrometer interfaces to PCs, PLCs and other embedded controllers through USB 2.0 or RS-232 communications. The information included in this document provides detailed instructions on the connection and functionality of the STS.

The detector used in the STS spectrometer is a high-sensitivity 1024-element CCD array from Panavision, product number ELIS-1024. (Ocean Optics applies a coating to the detector, so the optical sensitivity could vary from that specified by the manufacturer.)

The STS can be powered by a USB interface or GPIO port. The STS is a microcontroller-controlled spectrometer, thus all operating parameters are specified through software interfacing to the unit.

Note

To achieve faster transfers than 17ms for 1024-pixel scans or 8ms for 128-pixel scans, it is necessary to insert a USB2.0 High-Speed hub between the STS and the host (PC). With the hub in place, it is possible to reach 13ms (75Hz) for 1024 pixels and less than 4ms (>250Hz) for 128 pixels. This is a USB limitation that is not specific to either the STS or the Ocean Optics device drivers.

Features

- ELIS1024 Detector
    - 1024 pixel linear CMOS
    - Pixel size: 7.8 x 125 µm

- Detector range (coated): 350-800 nm for STS VIS-NIR and 650-1100 nm for STS-NIR, 190-650 for STS-UV
- Pixel well depth: 800k e-
- Defective pixels: up to 5 below 300ms integration time; up to 20 at ≥1s integration time
- Sensitivity: 6.74V/lux-sec typical (555nm)
- Absolute Quantum Efficiency at Peak (675nm) 60% (typical)

- Optics
  - Custom-molded collimating and focusing mirrors
  - Grating: 600 g/mm
  - Focal length: 28mm
  - Shaped entrance aperture
- Spectroscopic
  - Wavelength range: VIS (350-800nm), NIR (650-1100nm), UV (190-650nm)
  - Approximate Optical resolution (FWHM): 1.0nm (10µm slit), 1.5nm (25µm slit), 6nm (100µm slit), 12nm (200µm slit)
- Signal-to-noise ratio: >1500 (at max signal)
- A/D resolution: 14 bits
- Dark noise: ≤3 counts rms
- Dynamic range: $5x10^9$ (system, 10s max integration), ~4600 single acquisition
- Integration time: 10µs – 10s
- 3 triggering modes
- Strobe functions: Single/Continuous
- Embedded microcontroller allows programmatic control of all operating parameters:
  - USB 2.0 Full speed
  - RS-232 300-460K baud (defaults to 9600 baud)

  NOTE: Allow a >=500ms delay after changing baud rate
- Onboard GPIO
  - 4 user programmable digital I/O
- Flash storage for
  - Wavelength Calibration Coefficients
  - Linearity Correction Coefficients
  - Stray light coefficients
  - Irradiance calibration
  - User data (4 x 348 bytes)
  - Bench configuration
  - Device alias (16 bytes)
- Plug-n-Play Interface for PC applications
- Gated Delay feature

- 19-pin MHDMI connector for interfacing to external products. A custom interface adapter to 15-pin high density D-sub connector is available.
- CE and RoHS Certification

Specifications

Specifications	Criteria
Absolute Maximum Ratings:	
V_{CC}	+ 5.5 VDC
Voltage on any pin	Vcc
Physical Specifications:	
Physical Dimensions	40 mm x 42 mm x 24 mm
Weight	68 g
Power	5V supply < 500mA inrush, 150mA average current
Spectrometer:	
Design	Asymmetric crossed Czerny-Turner
Focal length	28 mm
Input fiber connector	SMA 905 -- See Warning on next page
Grating	600g/mm
Entrance Slit	10, 25, 100, or 200 µm slits (In the absence of a slit, the fiber acts as the entrance slit)
Detector	ELIS1024, 1024 pixel linear CMOS, 7.8 x 125µm pixels
Hot Pixels[1]	Typically 0 – 5 ; 20 maximum
Detector Sensitivity	6.74V/lux-sec typical (555nm)
Range	
VIS	350 – 800 nm
NIR	650 – 1100 nm
UV	190 – 650 nm
Pixel Well Depth	800k e-
Average linearity	< +/- 1% from 15-95% full scale (2500 - 14000 counts net)
Corrected linearity	< +/- 0.5% from 15-95% full scale (2500 - 14000 counts net)
Uncorrected linearity	+/-5% from 5-95% full scale (2500 - 14000 counts net)
Spectroscopic:	
Integration Time	10 µs – 10 s
Dynamic Range	5 x 10^9 (system, 10s max. integration), ~4600 single acquisition
Signal-to-Noise Ratio	>1500:1 (maximum signal)
Readout Noise	≤3 counts rms
Dark Current	~150 counts/sec at 60°C; ~50 counts/sec at 35°C
Fixed Pattern Noise (Normal Mode)[2]	± 3 counts
Resolution (FWHM)	
Typical	1nm (10µm slit), 1.5nm (25µm slit), 6nm (100µm slit), 12nm (200µm slit)
Maximum	2.0nm (10µm slit), 2.5nm (25µm slit), 8nm (100µm slit), 16nm (200µm slit)
A/D Resolution	14 bits
Stray Light	Maximum 0.25% nm below cutoff filter: UV GG420, VIS OG590, NIR RG850
Wavelength Temperature Stability	0.06 pixel/° C
Spectrometer Channels	One
Baseline Drift	± 3 counts[3]
Unit-to-unit repeatability	± 25%

[1] Hot pixels at 1 ms integration time are defined as those that are barely outside the range 1500 ± 3 counts; at 2 s: <1300 counts

[2] Fixed pattern noise (Raw Mode) is ~100 counts up to 300ms integration time

3 Baseline Drift auto-corrected for temperature, in Normal mode operation (other than dark current contribution, which should only be significant for integration time >300 ms)

Specifications	Criteria
Environmental Conditions: Temperature Humidity	 -30° to +70° C Storage & 0° to +50° C Operation; 10° C change/hour ramp 0% - 90% noncondensing
Interfaces: USB RS-232	 USB 2.0, 12 Mbps 3-wire RS-232 (Tx, Rx, ground), scan rate of ~5 scans per second at 460K Baud; Communications is N81 with support for hardware (RTS/CTS) handshaking (with firmware version 0043 and later) and no support for software flow control; Default baud rate 9600; +/- 5V

WARNING

Use only precision connectors that meet IES specification standard 60874 when connecting a fiber to the STS. Ferrule lengths that are out of specification can destroy the STS.

Note

For typical integration times and normal ambient temperatures, it should only be necessary to perform a single dark scan after startup at a given integration time.

Mechanical Diagram

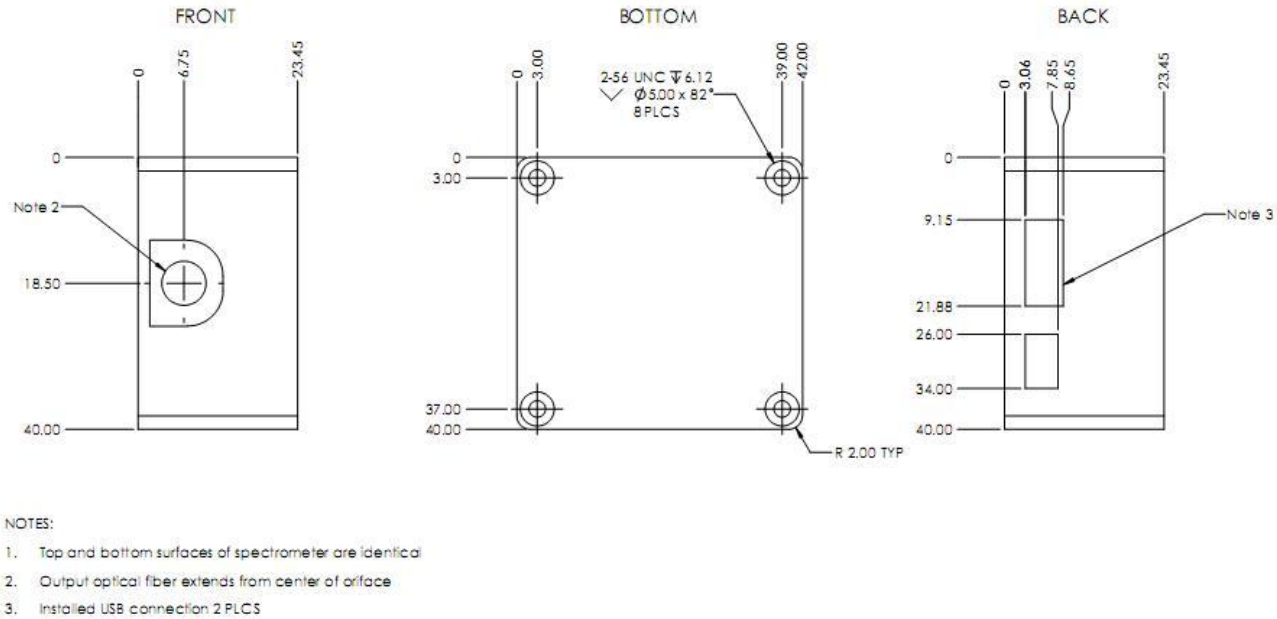
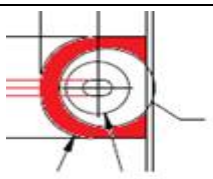


Figure 1: STS Dimensions

Fiber tolerance: the position of the fiber is limited by the movement of the Z-Sleeve within the boundary of the Housing recessed area (see red area). The Z-Sleeve diameter is 9.0mm. The Housing recess width is 12.7mm. Therefore, it is possible for the fiber position to vary +/- 1.8mm.



Recess in top of Housing

Electrical Pinout

19-Pin Mini HDMI Connector

Listed below is the pin description for the 19-pin Mini HDMI Accessory Connector. The connector is model MHDMR-19-02-F-TH-L-TR from Samtec. Connector information and samples are available from **www.Samtec.com**. Use of an off-the-shelf mHDMI cable is not acceptable as some of the pins may be shorted internally.

Pin#	Signal Name	Description
1	GND	Ground
2	Rx	RS-232 Data In
3	Tx	RS-232 Data Out
4	CTS	Clear to Send*
5	ExtTrgIn	(3.3v Logic)
6	Lamp_Enable	(3.3v Logic)
7	RTS	Request To Send*
8	Continuous Strobe	(3.3v Logic)
9	Single Strobe	(3.3v Logic)
10	Ext_Reset	External Reset
11	N/A	N/A
12	N/A	N/A
13	N/A	N/A
14	N/A	N/A
15	GPIO-4	(3.3v Logic)
16	GPIO-3	(3.3v Logic)
17	GPIO-2	(3.3v Logic)
18	5V In	External 5V power
19	GPIO-1	(3.3v Logic)
*Only available on boards at revision F or later.		

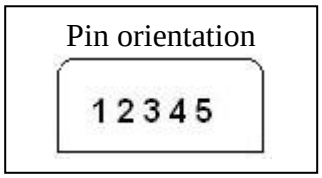
Pin orientation

Function	Input/Output	Description
RS-232 Tx	Output	RS-232 Transmit signal – for communication with PC connect to DB9 pin 2. ± 5 V min swing ± 5.7 V typ, 3 kΩ load to ground.
RS-232 Rx	Input	RS-232 Receive signal – for communication with PC connect to DB9 pin 3. -30 v min +30 v max. Max. voltage for low 0.6, min. voltage for high 2.4. Input resistance 3-7 kΩ
Lamp Enable	Output	A CMOS signal that is driven Active HIGH when the Lamp Enable command is sent to the STS followed by spectrum request
Continuous Strobe	Output	CMOS output signal used to pulse a strobe at configured frequency
Ground	Input/Output	Ground
Single Strobe	Output	CMOS output pulse used as a strobe signal, which has a programmable delay relative to the beginning of the spectrometer integration period. Also has programmable width.
External Reset	Input	Hold to low for 1 ms to reset. After reset or after startup, wait for 1s before sending commands to the STS (the STS performs a number of measurements at startup to correct for baseline). As long as the communications interface is up when the message is sent, the device will eventually respond even if it is still booting, but it is best to wait 1s before attempting communication. The pin is internally weakly pulled up by a 10K resistor to 3.3V.
ExtTrigIn	Input	The CMOS input trigger signal. In External Hardware Trigger mode this is a rising edge trigger input.
RS-232 CTS	Output	RS-232 Control logic signal - used to enable or suspend host transmission to the STS. Positive bus voltage means "OKAY FOR HOST TO SEND". Negative bus voltage indicates that the host should suspend transmission within 4 to 8 characters. When using Revision F CCAs, the STS will maintain a positive voltage at the CTS output when this interface is idle.
RS-232 RTS	Input	RS-232 Control logic signal - used to enable STS transmission to the host. Positive bus voltage means "OKAY FOR STS TO SEND". Negative bus voltage indicates that the STS should suspend transmission. The STS will respond to a change from a positive to a negative bus voltage within 2 characters. When using Revision F CCAs, it is recommended that the host maintain a Positive bus voltage when this interface is idle.
GPIO(1-4)	Input/Output	Four 3.3V General Purpose Software Programmable Digital I/Os

5-Pin Micro USB Connector

Listed below is the pin description for the Micro USB connector.

Pin#	Signal Name	Description
1	V_{USB}	USB power connection (+5V)
2	D-	Data Line -
3	D+	Data Line +
4	N/C	No connection
5	GND	Ground



DB15-to-DB9F Serial Port Cable Connector Pinouts

The pin description shown below is for the 1-foot 15-pin to 9-pin adapter cable (Part Number 030-40000-000) that connects to the JAZ-CBL-DB15 accessory cable (which attaches to the MHDMI connector) and enables RS232 communication. See JAZ-ADP-GPIO Adapter and JAZ-CBL-DB15 Accessory Cable Instructions for more information on the JAZ-CBL-DB15.

Note

The ADP-MHDMI-RS232 cable kit is available from Ocean Optics to allow the STS to interface to an accessory device's RS-232 connection. However, this cable kit does not support flow control. A custom cable must be created to support this feature.

DB-15 Pin#	DB-9F Pin#
10	5
6	3
7	2

Reprogramming Mode

Reprogramming mode is a mode where the device is ready to accept a new application firmware. The STS indicates that it is in reprogramming mode by the status LED flashing in a pattern of two short blinks followed by a pause. There are two ways the device can enter reprogramming mode:

- If the device fails the Start-up Test function.
- By sending the reprogramming mode command over USB or RS232.

Start-up Test Function

As of firmware revision 0034, the STS will verify its firmware on startup. If the firmware is intact, the device will operate normally. If it appears to be damaged or corrupted, then the device will enter reprogramming mode. This start-up test protects the STS from being rendered unusable due to failed attempts at reprogramming.

Upgrading Firmware

To change the application firmware in the STS the device must first be put into reprogramming mode. Once in reprogramming mode an OBP file can simply be written to the device via USB or RS232. The OBP file is a self-contained encrypted message that will reprogram the STS. RS232 data must be sent with 8N1 at 9600 baud. It is highly recommended to use hardware flow control for this procedure. If hardware flow control is not available then a schedule of pauses must be followed. The schedule is as follows. After the first 128 bytes wait 3 seconds, then after each 64 byte block wait 10 ms till file write is complete.

Reliability Prediction

The reliability calculations performed for the STS Spectrometer excludes failures related to known design flaws and product overuse (stress). To date, 38,000 unit-hours of STS operation in typical use environments have been accumulated with no “lifetime-related failures.” Based on this data (and assuming a worse-case scenario of one failure occurring today), the MTBF is at least 31562 hours, at a confidence interval of 70%. The predicted MTBF continually increases proportionally with failure-free usage.

Pixel Binning

In pixel binning, the pixels are actually physically summed so that 2x binning cuts the number of pixels in half to 512 and 4x down to 256 and 8x to 128. Resolution is not maintained while increasing sensitivity; resolution is lost in the trade off.

Because it is full speed and not high speed, 200+ scans per second is difficult. There are tight timing requirements on the USB host in order to get higher than this rate. The STS will do it, but the host has to be trying to get the data very quickly. On most machines, getting speeds above 200 requires a USB hub to be between the STS and the computer since the hub attempts to read the data back from the device more often than the machines native full speed USB chips do. The transaction translator of a high speed hub can Increase the STS throughput by reducing some of the inactive time periods where the STS is waiting to send its data.

At 115KBaud, a 10 ms – 20 ms delay between commands is appropriate. Assuming 128 bytes for message send and receive, a 14 ms turnaround is expected. The only exception is the Baud Rate command; you need to wait at least 500 ms after sending that command before closing and reopening the port at the new rate. At 115KBaud, you can receive approximately 5 spectra per second.

Binning Mode	USB/ No Hub	USB High-speed Hub	RS232 460K	RS232 115K
0 (1024)	70	80	14	5
1 (512)	120		28	10
2 (256)	160		40	15
3 (128)	250	450	70	25

Timing Signals

Timing Constraints

The following table defines the timing limits of firmware-controlled aspects of the STS:

Parameter	Value	Notes
Minimum integration time	10µs	Integration period error is approximately 1% at 10µs, and less than 0.1% for integration times ≥ 1ms
Minimum cycle time	13.3ms	Cycle time includes integration time, readout, and transmission via USB. This time may only be guaranteed at the minimum integration time.
Single strobe resolution	1µs	
Single strobe minimum delay	5µs	
Single strobe transition maximum delay	335.5ms	
Continuous strobe minimum period	50µs	
Continuous strobe resolution	1µs	
Continuous strobe maximum period	5s	
External trigger response With no delay With delay	 200ns latency,+/- 100ns < 1µs latency, +/- 100ns	Minimum delay for Trigger Delay mode is 5µs Maximum delay for Trigger Delay mode is 335.5ms Trigger delay resolution is 1µs
USB bandwidth	USB 2.0 Full Speed (12mbit/s)	

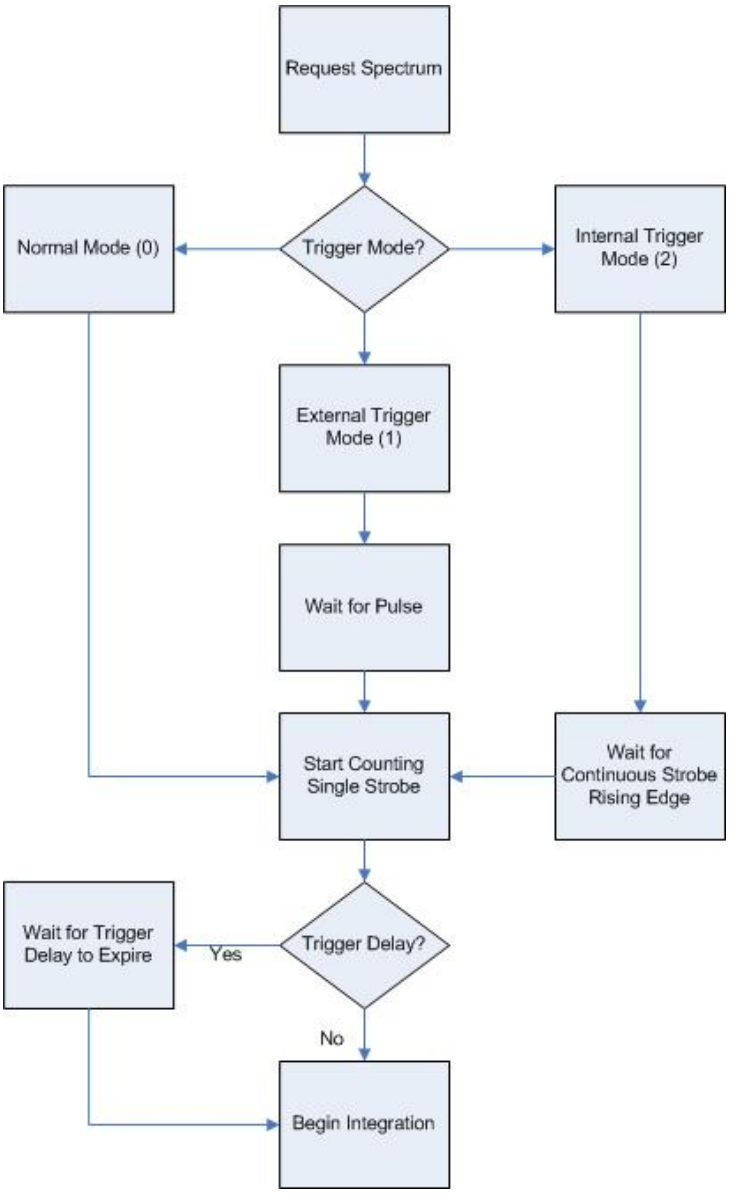
Strobe Signals

The STS provides three timing signals through an external header to trigger external devices. These signals are single strobe, continuous strobe, and strobe/lamp enable. A typical device that may need to be triggered is a pulsed Xenon lamp such as the PX-2, which can be driven with a single or continuous strobe. Alternately, the strobe/lamp enable pin could control an LED or other constant light source.

The single strobe and trigger delay events should be kept no less than 3µs apart. Otherwise, the single strobe edges or the start of delayed integration may be slightly later than expected.

On the STS, the microprocessor is responsible for generating all of the timing and to some extent will be forced to change output signals sequentially if multiple events are due at once.

The following figure shows the relationship among the triggering modes and strobe signals.



Single Strobe

The Single Strobe (SS) signal is a pulse that occurs after a programmable delay. This pulse has a user-defined delay and width. This pulse allows for synchronization of external devices to the spectrometer's integration period. The Strobe delay can range from 5µs to 335.5ms. If the trigger delay is zero, the Single Strobe is based on the beginning of the integration period. Otherwise, the single strobe and trigger delay timers begin counting from the same moment, which can be internally or externally triggered.

Continuous Strobe

The Continuous Strobe signal is software-configurable. The STS has a special trigger mode (Internal, Mode 2) that treats the continuous strobe like a trigger so that the spectrometer is automatically synchronized to the pulse. The integration period starts 4.24 microseconds before the rising edge of the continuous strobe.

Synchronizing Strobe Events

If the application requires more than one pulse per integration period, the user needs to insure the continuous strobe and integration period are synchronized. The integration time must be set so that an equal number of strobe events occurs during any given integration period. The Internal Trigger (Mode 2) should be used in this case.

Triggering Modes

The STS supports three triggering modes, which are set with the Trigger Mode command. Detailed information of each triggering mode follows.

Normal (Mode 0)

In this mode, the STS will begin integrating as soon as possible after receiving a request for a spectrum from a remote host.

External Hardware Trigger (Mode 1)

In this mode, the STS can be configured to begin the integration period with no delay (as soon as possible after receiving an electrical signal on the external trigger pin with 200ns +/- 100ns latency). Alternately, the STS can be configured to begin integrating at a fixed delay after receiving an electrical signal on the external trigger pin. This delay will be accomplished through a timer (minimum delay 5µs +/- 100ns).

Internal Trigger (Mode 2)

In this mode, the STS will be triggered at the beginning of the next rising edge of the continuous strobe signal, at which point a trigger delay may begin if configured (otherwise the integration would begin).

Digital Inputs & Outputs

General Purpose Inputs/Outputs (GPIO)

The STS has 4 user programmable 3.3V digital Input/Output pins, which can be accessed at the 19-pin MHDMI accessory connector. Through software, the state of these I/O pins can be defined and used for multi-purpose applications such as communications buses, sending digital values to an LCD/LED display, or even implementing complex feedback systems.

Recommended Operating Voltages:

VIH High-level input voltage: 2.0 (min) to 5.0 V
VIL Low-level input voltage: 0 to 1.3 V

Absolute Maximum Operating Voltages:
VIN -0.3 to 5.5 V

Typical Output Voltage: 3.3V

High-level source current:
VOH = 2.4 V min
IOH = 8-mA

Low-level sink current:
VOL = 0.4 V max
IOL = 8-mA

Communication and Interface

USB 2.0

The primary data interface between the STS 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 ~ 75 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 commands) while waiting for the spectrum to be returned (see *USB 2.0* for the USB command 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 STS to be a standalone device, which can output data to other logic devices/controllers such as a PLC or microcontroller.

RS-232 operation requires separate power (5-Volt power source), either through the MHDMI connector (Pin 1 ground, Pin 18 5V+) or the USB port (using the standard micro USB cable).

STS USB/RS-232 Port Interface Communications and Control Information

Overview

The STS is a microcontroller-based Miniature Fiber Optic Spectrometer that can communicate via the Universal Serial Bus (USB) or RS-232. This section contains the necessary command information for controlling the STS 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 STS. Only experienced USB programmers should attempt to interface to the STS via these methods.

Note

After start-up or reset, wait 1 second before sending commands to the STS. The STS performs measurements at start-up to correct for baseline.

USB Information

Ocean Optics Vendor ID number is 0x2457 and the Product ID is 0x4000.

Protocol Design

The binary command protocol for the STS 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 commands rather than EEPROM slots.

STS Command Protocol

There are two types of messages in this protocol:

- "commands" that do not return any information
- "queries" that cause the device to return information

When developing a device driver that will communicate with the STS, 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 "command", and left at 0 for every "query", then all communications with the STS 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 an STS 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. This must not set the "ACK requested" bit because to do so would generate a spurious ACK in addition to the expected response.

By using these two functions to encapsulate all transfers to the STS, 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	0xC1C0	Chosen for the following reasons: • High bits set, so not likely to occur in ADC data less than 16 bits wide or message type fields • When concatenated with the tail of a previous message, creates a distinct sequence Note: Do not reverse this byte order.
2	Protocol Version	2	0x0000 – 0xFFFF	Initially set to 0x1000. The host should only send messages known to be supported in the reported version of the protocol. Subsets of this protocol may be specified for other devices using a version less than 0x1000. The device may reject messages with a specified protocol it does not recognize.

Offset (Bytes)	Field	Size (Bytes)	Valid Values	Notes
4	Flags	2	0x0000 – 0xFFFF	Bits in this flag are assigned as follows: • Bit 0: response to earlier request (message type is set equal to request type). Set by device. • Bit 1: acknowledgment (ACK) if previous message included request for ACK. Set by device. • Bit 2: acknowledgment (ACK) requested. Set by Host. • Bit 3: 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. Set by device. Error Number field contains reason for NACK. • Bit 4: 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. Set by device. • Bit 5: The message protocol used by the caller is deprecated. If set, an older version of the protocol has been detected (version less than 0.1100). Set by the device.
6	Error Number	2	0x0000 – 0xFFFF	Only set by the 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. 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 (Commands have come from too many bus interfaces) • 11: Out of memory. Failed to allocate enough space to complete request. • 12: Command is valid, but desired information does not exist. • 13: Int Device Error. May be unrecoverable. • 100: Could not decrypt properly

Offset (Bytes)	Field	Size (Bytes)	Valid Values	Notes
				• 101: Firmware layout invalid • 102: Data packet was wrong size (not 64 bytes) • 103: hardware revision not compatible with firmware • 104: Existing flash map not compatible with firmware • 255: Operation/Response Deferred. Operation will take some time to complete. Do not ACK or NACK yet.
8	Message Type	4	0x00000000 – 0xFFFFFFFF	Each message type represents a command. See *Message Types*.
12	Regarding	4	0x00000000 – 0xFFFFFFFF	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	Valid types: • 0: no checksum (must still provide a block of 16 bytes after the payload, but they can be zero). • 1: MD5 (fully fills the 16 byte checksum block)
23	Immediate Data Length	1	0x00 – 0x10	Total number of bytes used in the Immediate Data field (see below).
24	Immediate Data	16		Provides an alternative to specifying a payload so commands 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	Includes the payload, if any, plus the checksum and footer. This is included for 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. STS 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;  /* = 0x1000 */
    unsigned short flags;
    unsigned short errno;
```

```
    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. Note that 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

The STS may populate the Immediate Data field or the Payload as appropriate for the length of the data being returned, regardless of the message type. Host programs decoding this protocol should always be capable of checking both areas in any response.

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) or if the checksum does not require the full 16 bytes. The unused parts of the block are for padding to ensure the message length is consistent. This protocol does not support checksums longer than 16 bytes, e.g. SHA, but the intent of the checksum is to detect bit errors, not to prevent tampering or to provide cryptographic assurance. 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 a salt value).

Footer

After the checksum block, a 4-byte footer is provided. The footer has a constant value of 0xC5C4C3C2. Do not reverse the order of the footer.

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 commands. 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 commands for the subcategory. 255 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

The subcategories and commands 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). All data will be carried over Endpoint 1or Endpoint 2 IN and OUT unless otherwise stated.

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	0x10	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 command	Checksum	0xC5	0xC4	0xC3	0xC2
		Footer			

The following is an example of how the Get And Send Corrected Spectrum Immediately message type (0x001 010 00) 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	0x10	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
0x00	0x10	0x10	0x00	*x*	*x*	*x*	*x*
Message type (0x00101000)				Regarding (user-specified)			

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

Byte 24 … Byte 39
unused

Byte 40	Byte 41	Byte 42	Byte 43	No payload	Byte 44…Byte 59	Byte 60	Byte 61	Byte 62	Byte 63
0x14	0x00	0x00	0x00		Checksum	0xC5	0xC4	0xC3	0xC2
Bytes remaining						Footer			

Response

Header

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0xC1	0xC0	0x00	0x10	0x01	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
0x00	0x10	0x10	0x00	*x*	*x*	*x*	*x*
Message type (0x00101000)				Regarding (user-specified)			

Byte 16	Byte 17	Byte 18	Byte 19	Byte 20	Byte 21	Byte 22	Byte 23
						0x00	0
Reserved						Checksum type	Immediate length
Byte 24 ... Byte 39							
unused							

Byte 40	Byte 41	Byte 42	Byte 43	Payload (2048 bytes of spectral data)	Byte 2092...Byte 2107	Byte 2108	Byte 2109	Byte 2110	Byte 2111
0x14	0x08	0x00	0x00		Checksum	0xC5	0xC4	0xC3	0xC2
Bytes remaining						Footer			

General Device Commands

Message Type	Purpose	Input Data	Output Data	Notes
0x000 000 00	Reset	N/A	N/A	Forces a reset of the device. Wait 1 second before reopening the port.
0x000 000 01	Reset Defaults			Clears certain persisted values including default baud rate and pixel binning mode. Once they're cleared, the device will reset and should come up in the factory default mode. Does not erase serial number, bench, alias, calibration, or user strings.
0x000 000 80	Get hardware revision	N/A	Unsigned byte	This value is sensed from the hardware itself. Request has no payload. Reply is a single byte.
0x000 000 90	Get 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. Request has no payload. Reply is a 2-byte integer (LSB first) of the revision.
0x000 001 00	Get serial number	N/A	String	

Message Type	Purpose	Input Data	Output Data	Notes
0x000 001 01	Get serial number length	N/A	Unsigned byte	Output data 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
0x000 003 00	Get number of available user strings	N/A	Unsigned byte	User-defined strings for storing small amounts of arbitrary data
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.
0x000 008 00	Get RS-232 baud rate	N/A	4-byte unsigned integer, LSB first	Factory default is 9600 baud
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	Input is baud rate as an integer (300 – 460,800). If a change baud rate is issued while another transfer is being sent out, the current transfer completes at the current baud rate before changing to the new baud rate.
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.
0x000 010 10	Configures Status LED			Byte 0 is reserved and must always be 0x00 Byte 1 is the pattern to drive the LED: 1 = LED will blink in an S-O-S pattern at a high priority (this will not override a POST or hard fault indication, but will override all others)

Message Type	Purpose	Input Data	Output Data	Notes
				2 = LED will fade in and out at a low priority (anything but the solid-on idle pattern will override this) If this byte is anything other than 1 or 2 then the LED will revert to its normal operation.
0x000 FFF 00	Put device in reprogramming mode	N/A	N/A	Causes the STS to accept an OBP file provided by Ocean Optics.

Spectrometer Commands

Message Type	Purpose	Input Data	Output Data	Notes
0x001 010 00	Get and send corrected spectrum immediately	N/A	Pixel values (integers)	This returns the intensity of every pixel on the detector as LSB, MSB as soon as it is available. There is no payload in the request. The reply has 2048 bytes of payload. The pixel intensities are corrected for temperature drift and fixed-pattern noise.
0x001 011 00	Get and send raw spectrum immediately	N/A	Pixel values (integers)	This returns the raw ADC output for every pixel on the detector as LSB, MSB as soon as it is available. There is no payload in the request. The reply has 2048 bytes of payload.
0x001 020 00	Get partial spectrum mode			Returns a specification for partial spectrum retrieval (see below). If no specification has been set since the device was started, this will return a NACK indicating "no value available".
0x001 020 10	Set partial spectrum mode			Input is as follows (partial pixel mode formats): Mode 1 (interval spacing): 0x01 00 SS SS Mode 2 (band): 0x02 00 SS SS II II CC CC Mode 3 (selected) :): 0x03 00 XX XX YY YY ZZ ZZ ... Where: SS SS is the starting pixel index (16 bits, LSB first), II II is the increment (positive or negative, but not zero, also 16-bits LSB first), and CC CC is the total number of pixels (including the starting pixel, cannot be larger than the total pixel count; 16 bits LSB first). XX XX is the index of the first pixel to grab, YY YY is the index of the second pixel to grab, ZZ ZZ is the third, and so on, for up to ten pixels. Each pixel index is 16 bits, LSB first. For instance, a payload of 0x03 00 05 00 01 00 09 00 02 00 would get pixels (5, 1, 9, 2) in that order.

Message Type	Purpose	Input Data	Output Data	Notes
0x001 020 80	Get and send partial corrected spectrum			Acquires a baseline-corrected spectrum and returns just the pixels specified by the last partial spectrum specification. If no specification has been set since the device was started, this will return a NACK with a "device not ready" error. Only the specified pixels will be in the response, so the caller must have the specification available to decipher the results. The pixels will be packed into the immediate data portion of the header if they will fit; otherwise, they will be appended after the bytes_remaining field as an extended payload. Averaging and boxcar will be applied if necessary before the values are returned. Partial spectrum modes: 0x0001: Acquire every Nth pixel, starting with pixel 0. The only parameter for this mode is the spacing, which is how much to increment the pixel index by after each pixel. 0x0002: Acquire a band of pixels. Three parameters are expected: starting pixel index (inclusive), increment (can be positive or negative, but not 0), count (includes a starting pixel, cannot be larger than the total count). The count is a maximum; fewer pixels will be returned if there are not enough pixels in the current binning mode to satisfy this parameter. 0x0003: Acquire pixels by index. This allows up to 10 pixel indices (identical to USB2000+) to be given in any order. If any index is invalid for the active resolution, the pixel value for that index will be 0xFFFF. This is to prevent the specification from becoming invalidated by changing resolutions. All parameters (including the mode) are 16-bit values (LSB first), and they are all unsigned except for the increment for mode 2.* The returned values should include any and all averaging and boxcar compensation as specified separately, and the baseline will be compensated as usual.

*Examples:

Specification mode 1, spacing of 4 (pixels 0, 4, 8, 12, 16, ...): 0x01 00 04 00

Specification mode 2, starting at pixel 100, every pixel for 10 pixels (100, 101, 102, ..., 109): 0x02 00 64 00 01 00 0A 00

Specification mode 3, for pixels 5, 8, 500, and 375: 0x03 00 05 00 08 00 F4 01 77 01

Message Type	Purpose	Input Data	Output Data	Notes
0x001 100 10	Set integration time (µs)	Unsigned 32-bit integer	N/A	Input is 4 bytes for time in µs. Order is LSB, …, MSB No reply. The minimum is 10.
0x001 101 10	Set trigger mode	Unsigned byte	N/A	Input has 1 byte in the request for trigger mode. Valid trigger modes: Mode 0 (default): integration begins as soon as possible after request Mode 1: integration or trigger delay begins with a rising trigger edge Mode 2: integration is internally triggered to synchronize with continuous strobe No reply.
0x001 101 20	Simulate trigger pulse	N/A	N/A	Causes the STS to react exactly as though an electrical rising edge signal was applied to the external trigger pin of the device. This is intended for cases where the device is put into an external trigger mode and an acquisition has been started, but no signal is forthcoming. By sending this command over a different communications interface (e.g. RS232 or the second USB endpoint) it is possible to make the device finish its acquisition normally.
0x001 102 80	Get pixel binning factor		Single byte	Returns a single byte indicating the binning mode.
0x001 102 81	Get maximum binning factor		Single byte	Returns a single byte representing the largest binning factor that may be used (3). The minimum is assumed to be zero.
0x001 102 85	Get default binning factor		Single byte	Returns the startup binning factor as a single byte (0 by factory default, may be overridden as an acquisition parameter while device is powered up, and default may also be reset; (see below).
0x001 102 90	Set pixel binning factor			Takes a single byte as the binning factor to use for this bus until the device is reset.
0x001 102 95	Set default binning factor	Zero bytes or single byte		Can take either zero bytes (clearing back to factory default) or one byte (indicating new default mode) as arguments

Message Type	Purpose	Input Data	Output Data	Notes
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. Input is 1 byte: 0 = off, 1= on No reply.
0x001 105 10	Set trigger delay (µs)	Unsigned 32-bit integer	N/A	Input is 4 bytes for time in µs (minimum 5 µs). Order is LSB, …, MSB No reply.
0x001 200 00	Get scans to average	Unsigned 16-bit integer		Gets the number of scans (1-5000) to average together before returning the spectrum.
0x001 200 10	Set scans to average	Unsigned 16-bit integer, LSB first		Sets the number of scans to average (1-5000). Argument is an unsigned 16-bit integer, LSB first. The spectrum response will still be 16 bits per pixel. The average will round to the nearest integer, with any exact half being rounded up.
0x001 210 00	Get boxcar width	Unsigned byte		Gets the boxcar width being applied to all spectra. Valid range is 0-15.
0x001 210 10	Set boxcar width	Unsigned byte		Sets the boxcar width to apply to all spectra. Valid range is 0-15. Boxcar smoothing will apply to every pixel, even if there are not a balanced number of neighbors on both sides.
0x001 801 00	Get wavelength coefficient count	N/A	Unsigned byte	Request has no input. Reply has 1-byte payload with number of coefficients.
0x001 801 01	Get wavelength coefficient	Unsigned byte	IEEE single-precision	Request has 1-byte input data for coefficient index starting with wavelength intercept at index 0. Reply has 4-byte float (LSB first).
0x001 801 11	Set wavelength coefficient	Unsigned byte, IEEE single-precision	N/A	Input is the order of the coefficient to set (indexing starts with wavelength intercept at index 0), followed by an IEEE single-precision float. No reply.
0x001 811 00	Get nonlinearity coefficient count	N/A	Unsigned byte	Reply has 1-byte output data with number of coefficients.

Message Type	Purpose	Input Data	Output Data	Notes
0x001 811 01	Get nonlinearity coefficient	Unsigned byte	IEEE single precision	Request has 1-byte input for coefficient index to retrieve. Reply has 4-byte float (LSB first).
0x001 811 11	Set nonlinearity coefficient	Unsigned byte, IEEE single precision	N/A	Input is the order of the coefficient to set, followed by an IEEE single-precision float. No reply.
0x001 820 01	Get irradiance calibration	N/A	Up to 4096 bytes	Request has no payload. Reply has up to 4096 bytes (whatever has been stored previously), intended for 1024 x 4-byte floats. If nothing has been stored, the reply will have NACK bit set in flags.
0x001 820 02	Get irradiance calibration count	N/A	Unsigned 32-bit integer	Request has no payload. Reply is a 4-byte integer, LSB first, indicating total number of 4-byte floats to be returned in payload of reply to Get Irradiance Calibration including 0.
0x001 820 03	Get irradiance calibration collection area	N/A	IEEE single precision	Request has no payload. Reply: If a collection area has been set, it's returned as a 4-byte float (LSB first). If not defined, it returns a NACK.
0x001 820 11	Set irradiance calibration	Up to 4096 bytes	N/A	Request has up to 4096 bytes in payload. Sending a zero-length buffer will delete any irradiance calibration from STS. No reply.
0x001 820 13	Set irradiance calibration collection area	IEEE single precision	N/A	Request has a 4-byte float. Otherwise, no payload (to erase value completely). Sending a zero-length buffer will delete any collection area previously stored. No reply.
0x001 831 00	Get number of stray light coefficients	N/A	Unsigned byte	
0x001 831 01	Get stray light coefficient	Unsigned byte	IEEE single precision	Input is the order of the coefficient to retrieve
0x001 831 11	Set stray light coefficient	Unsigned byte, IEEE single precision	N/A	Input is the order of the coefficient to set, followed by an IEEE single-precision float

Message Type	Purpose	Input Data	Output Data	Notes
0x001 860 00	Spectrometer get hot pixel indices	N/A	Up to 116 bytes	Request has no data. Reply has up to 58 x 2-byte integers (1 integer per pixel index). If nothing has been stored, the reply will have NACK bit set in flags.
0x001 860 10	Spectrometer set hot pixel indices	Up to 116 bytes	N/A	Request has up to 58 x 2-byte integers for pixel indices. No reply.
0x001 B00 00	Get bench ID	N/A	String	Request has no input data. Reply has up to a 32-byte ASCII string in output (for bench ID).
0x001 B01 00	Get bench serial number	N/A	String	Request has no input data. Reply has an ASCII string in output (for serial number).
0x001 B02 00	Get slit width microns	N/A	Unsigned 16-bit integer	Request has no input data. Reply is a 2-byte integer (LSB first) for the micron value.
0x001 B03 00	Get fiber diameter microns	N/A	Unsigned 16-bit integer	Request has no input data. Reply is a 2-byte integer (LSB first) for the micron value.
0x001 B04 00	Get grating	N/A	String	Request has no input data. Reply has an ASCII string in output (for grating).
0x001 B05 00	Get filter	N/A	String	Request has no input data. Reply has an ASCII string in output (for filter).
0x001 B06 00	Get coating	N/A	String	Request has no input data. Reply has an ASCII string in output (for coating).

GPIO Commands

Bit 0 =GPIO-1, Bit 1 = GPIO-2, Bit 2 = GPIO-3, Bit 3 = GPIO-4

Message Type	Purpose	Input Data	Output Data	Notes
0x002 000 00	Get number of GPIO pins	N/A	Unsigned byte	
0x002 001 00	Get output enable vector	N/A	32-bit integer	Bits are set to 1 for pins configured as outputs.
0x002 001 10	Set output enable vector	2 unsigned 32-bit integers	N/A	First 32 bits is vector, second 32 bits is mask. Bits set to 1 correspond to pins being set as outputs. Only the bits in the vector whose corresponding bits in the mask are 1 will be changed.
0x002 003 00	Get value vector	N/A	32-bit integer	Can be used to read inputs or configuration of outputs.
0x002 003 10	Set value vector	2 unsigned 32-bit integers	N/A	First 32 bits is vector, second 32 bits is mask. Bits set to 1 correspond to pins being driven to a logic high. Only bits configured as outputs will be changed. Only the bits in the vector whose corresponding bits in the mask are 1 will be changed.

Strobe Commands

Message Type	Purpose	Input Data	Output Data	Notes
0x003 000 10	Set single-strobe pulse delay (µs)	unsigned 32-bit integer	N/A	Minimum is 5 µs.
0x003 000 11	Set single-strobe pulse width (µs)	unsigned 32-bit integer	N/A	Minimum is 1 µs.
0x003 000 12	Set single-strobe enable	unsigned byte	N/A	0 = disabled 1 = enabled
0x003 100 10	Set continuous strobe period (µs)	unsigned 32-bit integer	N/A	Minimum is 50 µs.
0x003 100 11	Set continuous strobe enable	Unsigned byte	N/A	0 = disabled 1 = enabled

Temperature Commands

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 STS contains three memory locations for the temperature sensor as follows:

0 = Detector Board Thermistor

1 = Reserved/Internal Use

2 = Microcontroller Sensor Temperature

Message Type	Purpose	Input Data	Output Data	Notes
0x004 000 00	Get temperature sensor count	N/A	byte	Request has no input data. Reply has 1 byte with the number of temperature sensors available.
0x004 000 01	Read temperature sensor	Unsigned byte	IEEE single-precision floating point	Provides the temperature in °C (if calibrated) or raw counts for the sensor (if uncalibrated). Input is 1 byte for the index of the sensor to read. Reply: output is a 4-byte float (LSB first) of temperature in °C.
0x004 000 02	Read all temperature sensors	N/A	Array of IEEE single-precision floating points	Each float is the temperature in °C (if calibrated) or raw counts for the sensor (if uncalibrated). The length of the array will be equal to the number of sensors available. Request has no input data. Reply has a 4-byte floating point (LSB first) for each sensor in the output. Units are in °C.

Appendix A

Calibrating the Wavelength of the STS and Nonlinearity Correction

Wavelength Calibration

This appendix describes how to calibrate the wavelength of your spectrometer. Though each spectrometer is calibrated before it leaves Ocean Optics, the wavelength for all spectrometers will drift slightly as a function of time and environmental conditions. Ocean Optics recommends periodically recalibrating the STS.

You are going to be solving the following equation, which shows that the relationship between pixel number and wavelength is a third-order polynomial:

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

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)

You will be calculating the value for I and the three Cs.

Calibrating the Spectrometer

Preparing for Calibration

To recalibrate the wavelength of your spectrometer, you need the following components:

- A light source capable of producing spectral lines

Note

Ocean Optics' HG-1 Mercury-Argon lamp is ideal for recalibration. If you do not have an HG-1, you need a light source that produces several (at least 4-6) spectral lines in the wavelength region of your spectrometer.

- An STS spectrometer
- A spreadsheet program (Excel or Quattro Pro, for example) or a calculator that performs third-order linear regressions

---

Note

If you are using Microsoft Excel, choose **Tools | Add-Ins** and check **AnalysisToolPak** and **AnalysisToolPak-VBA**.

Calibrating the Wavelength of the Spectrometer

► *Procedure*

Perform the steps below to calibrate the wavelength of the spectrometer:

1. Place SpectraSuite into Scope mode and take a spectrum of your light source. Adjust the integration time (or the A/D conversion frequency) until there are several peaks on the screen that are not off-scale.
2. Move the cursor to one of the peaks and position the cursor so that it is at the point of maximum intensity.
3. Record the pixel number that is displayed in the status bar or legend (located beneath the graph). Repeat this step for all of the peaks in your spectrum.
4. Use the spreadsheet program or calculator to create a table like the one shown in the following figure. In the first column, place the exact or true wavelength of the spectral lines that you used.

   In the second column of this worksheet, place the observed pixel number. In the third column, calculate the pixel number squared, and in the fourth column, calculate the pixel number cubed.

Independent Variable

Dependent Variables

Values Computed from the Regression Output

True Wavelength (nm)	Pixel #	Pixel # ^{2}	Pixel # ^{3}	Predicted Wavelength	Difference
253.65	175	30625	5359375	253.56	0.09
296.73	296	87616	25934336	296.72	0.01
302.15	312	97344	30371328	302.40	-0.25
313.16	342	116964	40001688	313.02	0.13
334.15	402	161604	64964808	334.19	-0.05
365.02	490	240100	117649000	365.05	-0.04
404.66	604	364816	220348864	404.67	-0.01
407.78	613	375769	230346397	407.78	0.00
435.84	694	481636	334255384	435.65	0.19
546.07	1022	1044484	1067462648	546.13	-0.06
576.96	1116	1245456	1389928896	577.05	-0.09
579.07	1122	1258884	1412467848	579.01	0.06
696.54	1491	2223081	3314613771	696.70	-0.15
706.72	1523	2319529	3532642667	706.62	0.10
727.29	1590	2528100	4019679000	727.24	0.06
738.40	1627	2647129	4306878883	738.53	-0.13
751.47	1669	2785561	4649101309	751.27	0.19

5. Use the spreadsheet or calculator to calculate the wavelength calibration coefficients. In the spreadsheet program, find the functions to perform linear regressions.
   - If using Quattro Pro, look under **Tools | Advanced Math**
   - If using Excel, look under **Analysis ToolPak**
6. Select the true wavelength as the dependent variable (Y). Select the pixel number, pixel number squared, and the pixel number cubed as the independent variables (X). After executing the regression, you will obtain an output similar to the one shown below. Numbers of importance are noted.

Regression Statistics

Multiple R	0.999999831
R Square	0.999999663
Adjusted R Square	0.999999607
Standard Error	0.125540214
Observations	22

R Squared

Intercept

	Coefficients	Standard Error
Intercept	190.473993	0.369047536
X Variable 1	0.36263983	0.001684745
X Variable 2	-1.174416E-05	8.35279E-07
X Variable 3	-2.523787E-09	2.656608E-10

First coefficient

Second coefficient

Third coefficient

7. Record the Intercept, as well as the First, Second, and Third Coefficients. Additionally, look at the value for R squared. It should be very close to 1. If not, you have most likely assigned one of your wavelengths incorrectly.

   Keep these values at hand.

Saving the New Calibration Coefficients: USB Mode

Use SpectraSuite to save calibration coefficients to the STS spectrometer.

► *Procedure*

To save wavelength calibration coefficients using SpectraSuite, perform the following steps:

1. In SpectraSuite, select Spectrometer Features.
2. Click on the Wavelength tab.
3. Enter the correct values for the intercept and the coefficients.
4. Click the Save to Spectrometer button to write the values to the STS spectrometer.

The new wavelength calibration coefficients are now loaded onto the EEPROM memory chip on the STS.

Nonlinearity Correction

A linear device is one in which the output is proportional to the input. For example, if the input is doubled then the output is doubled. In the case of a spectrometer, the output is counts and the input is light (photons). We expect doubling the amount of light should get double the counts at each pixel (assuming a zero baseline). It may not be easy to double the amount of light by adjusting the light source but it is easy to double the amount of light collected by doubling the integration time. So, for a constant amount of light, the signal (in counts) should vary proportionally with the integration time. The detector and the A/D converter in a real spectrometer will have some degree of nonlinearity, and the correction compensates for that.

Nonlinearity correction uses either a 4th order or a 7th order polynomial (a 4th order polynomial can be thought of as a 7th with C5, C6 and C7 set to zero). Many Ocean Optics devices store the correction coefficients in the device, including the STS.

The correction for nonlinearity is performed as follows.

Where:

P: Pixel number starting at 0.

Sp: Scope-mode intensity (uncorrected counts) of a sample at pixel *p*

Dp: Scope mode intensity (uncorrected counts) of pixel *p* for a stored dark spectrum c0, c1, ..., c7: Non-linearity correction coefficients (7th order shown here; stop at c4 for 4th order).

Fp: correction factor for pixel *p*

Rp: Corrected scope-mode intensity of sample at pixel *p*

x: Temporary variable representing the sample minus baseline

For every pixel P in the spectrum:

$x = (Sp - Dp)$

$Fp = c0 + c1x + c2x2 + c3x3 + c4x4 + c5x5 + c6x6 + c7x7$

$Rp = x / Fp$

The resulting spectrum, *R*, is the nonlinearity-corrected spectrum. Typically, the dark spectrum is added back in (thus, the end result is (*Rp*+*Dp*)) after doing all such corrections so that further spectral math can subtract the dark again without resulting in a negative baseline.

The process for computing these coefficients is somewhat involved. Ocean Optics provides software that can do this calculation (OOINLCorrect). The experiment for deriving these coefficients requires a stable light source that can saturate the spectrometer across a significant portion of its spectral range (e.g., LS-1). The software changes the integration time steadily and determines the variation in intensity versus time and generates the polynomial coefficients that will make this linear.