



# ENG-0034 EEPROM Format

Revision 18  
21-Apr-2026

# Changelog

Rev	Date	By	Reason
1	2017-25-1	J. Traud	Formatting, updated FID protocol to include coefficients on device.
2	2017-18-7	J. Traud	Added Excitation to page 1. Added bad pixel allotment to page 5.
3	2018-05-29	J. Traud	Updated spec to include two sets of GAIN and OFFSET values as well as calibration information for output laser power (UNRELEASED)
4	internal	M. Zieg R. Dickerson	<ul style="list-style-type: none"> <li>● Added floating-point excitation wavelength</li> <li>● changed revisioning from per-page to entire EEPROM</li> <li>● changed ints to uint where negatives were invalid</li> <li>● moved min/maxIntegrationTimeMS to fit 24-bit range</li> </ul>
5	internal	M. Zieg	added productConfiguration
6	2018-18-09	M. Zieg	added Raman intensity calibration
7	internal	T. Stohrer	added Average FWHM ( <i>nm or cm<sup>-1</sup> per Excitation</i> )
8	2020-03-23	M. Zieg	added page 6/7 subformat
9	internal	M. Zieg	Added FeatureMask
10	Feb 25, 2021	M. Zieg	Added laserWarmupSec, FeatureMask.gen15, and FeatureMask.cutOffFilterInstalled
11	Apr 14, 2021	M. Zieg	Added subformat 3
12	Jun 28, 2021	M. Zieg	Added HardwareEvenOddCorrection
13	Nov 4, 2021	M. Zieg E. Dort	Updated subformat 3, added subformat 4
14	Mar 21, 2022	M. Zieg	Added sigLaserTEC, hasInterlockFeedback
15	Nov 30, 2022	M. Zieg	added laserWatchdogSec, lightSourceType, hasShutter
16	4-Nov-2024	M. Zieg	<ul style="list-style-type: none"> <li>● added <ul style="list-style-type: none"> <li>○ startupLaserTecSetpointRaw</li> <li>○ FeatureMask.disableBlePower</li> <li>○ FeatureMask.disableLaserArmedIndication</li> <li>○ powerWatchdogTimeoutSec</li> <li>○ detectorTimeoutSec</li> <li>○ horizontalBinningMode (w/modes 0-5)</li> </ul> </li> <li>● clarified <ul style="list-style-type: none"> <li>○ FPGA_COMPILATION_OPTIONS</li> <li>○ Subformat 4 field locations</li> </ul> </li> </ul>
17	3-Mar-2025	M. Zieg	<ul style="list-style-type: none"> <li>● removed <ul style="list-style-type: none"> <li>○ baud rate</li> <li>○ linearity coeffs</li> <li>○ lifetime metrics</li> <li>○ min/max laser temperature</li> </ul> </li> <li>● changed</li> </ul>

			<ul style="list-style-type: none"> <li>○ moved scansToAverage from subformat 4 to page 3</li> <li>● added                             <ul style="list-style-type: none"> <li>○ subformat 5 (Multi-Wavelength Raman)</li> </ul> </li> </ul>
18		M. Zieg	<ul style="list-style-type: none"> <li>● added                             <ul style="list-style-type: none"> <li>○ smlAttenuatorDAC (range 10-40 decimal)</li> <li>○ FeatureMask.laserTimeoutAfterAcqCnt</li> <li>○ FeatureMask.isOEM</li> <li>○ allocate 6 bytes for assemblyRev</li> <li>○ maxLaserTempDegC (sint8)</li> <li>○ XS Page 8</li> </ul> </li> <li>● changed                             <ul style="list-style-type: none"> <li>○ expand Laser Watchdog to FX2</li> </ul> </li> </ul>

*Note that the Revision of the ENG-0034 document corresponds to the “format” field of the EEPROM page structure, as indicated in the last byte on the first page (page 0, byte 63).*

# Contents

<b>1. General Description</b>	<b>5</b>
1.1. USB PID	5
1.2. Other sources of information	5
1.3. Software Driver Libraries	5
<b>2. Field Definitions</b>	<b>6</b>
2.1. Feature Mask	6
2.1.1. FeatureMaskXS	7
2.2. Light Source Type	7
2.3. Horizontal Binning Method	7
2.3.1. BIN_2X2	8
2.3.2. CORRECT_SSC	8
2.3.3. CORRECT_SSC_BIN_2X2	8
2.3.4. BIN_4X2	8
2.3.5. BIN_4X2_INTERP	9
2.3.6. BIN_4X2_AVG	9
<b>3. EEPROM Page Structure</b>	<b>9</b>
<b>3.1. XS EEPROM Pages</b>	<b>13</b>
Proposed Fields	13
<b>4. Custom EEPROM Structure</b>	<b>14</b>
4.1. Subformat 1: NIST SRM Raman Intensity Calibration	14
4.2. Subformat 2: Advanced Wavelength Calibration Spline	15
4.3. Subformat 3: Untethered Configuration	18
4.4. Subformat 4: Detector Regions	19
4.5. Subformat 5: Multi-Wavelength Raman	19
<b>Proposed Changes</b>	<b>19</b>

# 1. General Description

This document details the method for identifying the model, serial number, configuration settings and features of a Wasatch Photonics spectrometer, primarily through parsing its internal EEPROM.

In particular, this describes models which utilize the Wasatch **Feature Identification Device (FID)** protocol, essentially a standard of using the onboard EEPROM to “self-describe” the features and confirmation available within the spectrometer.

This document does not describe spectrometers designed for the OCT market such as the Cobra series.

## 1.1. USB PID

The simplest way to tell what type of spectrometer you’re connected to is by checking its USB VID (Vendor ID) and PID (Product ID) codes as reported by the USB bus. Following are valid / supported VID and PID combinations for Wasatch Photonics USB spectrometers.

VID	PID	USB Descriptor
0x24aa	0x1000	WP spectrometer with FX2 $\mu$ Controller and Hamamatsu silicon detector
0x24aa	0x2000	WP spectrometer with FX2 $\mu$ Controller and Hamamatsu InGaAs detector
0x24aa	0x4000	WP spectrometer with ARM $\mu$ Controller

## 1.2. Other sources of information

Other information about the spectrometer can be determined by communicating with it via USB opcodes and reading the responses of various USB commands described in Wasatch Photonics Engineering document ENG-0001, “USB FID API.” This document describes the EEPROM contents and structure; that document describes all the various USB commands supported by FID spectrometers, including those required to read and write the EEPROM.

In particular, there are USB commands provided to retrieve the spectrometer’s microcontroller firmware version, its FPGA firmware version, FPGA compilation options and other key attributes from which the supported feature set can be derived.

## 1.3. Software Driver Libraries

Although this document provides a technical reference to the EEPROM contents and structure, most application developers are not required (or advised) to read and parse the EEPROM manually. All of Wasatch’s “application-level drivers” (control libraries) have pre-built functions to parse the EEPROM contents and make them easily accessible as clearly labeled object attributes.

Examples:

- C# / .NET: [WasatchNET.EEPROM](#)
- Python: [wasatch.EEPROM](#)
- C/C++: [WasatchVCpp::EEPROM](#)
- Xamarin: [EnlightenMobile.Models.EEPROM](#)

## 2. Field Definitions

The following EEPROM fields represent bitmasks or enumerations requiring additional format specification beyond the raw field table.

### 2.1. Feature Mask

“FeatureMask” is a big-endian uint16 EEPROM field on page 0 which provides compact storage of certain rare features and settings which software should be aware of. The current field structure is:

Bit	Mask	Name	Description	Initial Rev
0	0x0001	invertXAxis	Spectra should be horizontally inverted (is read-out red-to-blue)	9
1	0x0002	bin2x2	2D detectors should attempt to bin (average) four-pixel squares (2 across by 2 tall), e.g. to smooth-out alternating colors from Bayer filters	9
2	0x0004	gen15	Spectrometer includes “Gen 1.5” electronics including the OEM Accessory Connector	10
3	0x0008	cutOffFilterInstalled	Spectrometer has a cut-off filter installed, presumably of a wavelength indicated by the configured Horizontal ROI pixel	10
4	0x0010	hardwareEvenOdd-Correction	InGaAs even-odd correction is performed in the spectrometer’s FPGA, and does not need to be applied in software (drivers etc)	12
5	0x0020	sigLaserTEC	SiG Laser has a TEC	14
6	0x0040	hasInterlockFeedback	Supports canLaserFire and isLaserFiring (via laser driver / interlock board)	14
7	0x0080	hasShutter	Spectrometer has a built-in shutter	15
8	0x0100	disableBLEPower	BLE power should be <i>disabled</i> at boot	16
9	0x0200	disableLaserArmed-Indication	If set, <i>do not</i> indicate "laser ARMED" via the "laser firing" LED on XS	16
10	0x0400	interlockExcluded	if set, indicates laser does not have an interlock	17
11	0x0800	laserTimeoutMissedFrameCount	If set, interpret laserWatchdogSec field to indicate the number of <i>unread consecutive spectra</i> which the detector may collect (in	18

Bit	Mask	Name	Description	Initial Rev
			free-running mode), before the laser automatically shuts off. This is used to detect "hung" or non-responsive software which is no longer actively reading the output of the spectrometer. This is only practicably usable over USB.	
12	0x1000	isOEM	unit is not a Wasatch standard product, and is allowed to exhibit unique OEM-specific behavior	18
13	0x2000		<i>reserved</i>	
14	0x4000		<i>reserved</i>	
15	0x8000		<i>reserved</i>	

### 2.1.1. FeatureMaskXS

In addition to the standard 16-bit FeatureMask field listed above, the following 4-byte FeatureMaskXS field is also available on XS spectrometers.

Bit	Mask	Name	Description	Initial Rev
0	0x0000 0001	bleDoorSensor	Subscribe to notifications on supported doorway sensor(s)	18

## 2.2. Light Source Type

Value	Definition
0	<i>undefined</i>
1	Class 3B Single-Mode Laser (~100mW max)
2	Class 3B Multi-Mode Laser (~450mW)
...	<i>reserved</i>
254	No laser or internal light source
255	<i>undefined</i>

## 2.3. Horizontal Binning Method

This value indicates the type of horizontal binning to be applied in software if the bin2x2 FeatureMask flag is enabled.

At this time, all horizontal binning is applied in software (e.g. Wasatch.PY or WasatchNET), not in firmware. For XS spectrometers with IMX385 detectors, 1952 pixels will be read from the spectrometer to the host over USB, even if that number is then reduced in software.

Value	Definition
0	BIN_2X2
1	CORRECT_SSC
2	CORRECT_SSC_BIN_2X2
3	BIN_4X2
4	BIN_4X2_INTERP
5	BIN_4X2_AVG
...	<i>reserved</i>

### 2.3.1. BIN\_2X2

This is historically the standard horizontal averaging applied on Sony IMX detectors known as "2x2 binning". A new "smoothed" spectrum is constructed by averaging together the raw intensity values of adjacent pixels. The original pixels 0 and 1 are averaged to make the new pixel 0, the original pixels 1 and 2 are averaged to make the new pixel 1, etc. The final averaged pixel is repeated (averaged pixel 1950 is copied to 1951) to maintain a 1952-pixel spectrum.

This process essentially left-shifts the entire spectrum by ½ pixel, and therefore requires updated wavelength and Raman Intensity Calibrations if changed.

### 2.3.2. CORRECT\_SSC

This algorithm uses interpolated lookup tables generated from Sony's published color sensitivity graphs for the Red, Green and Blue channels of the Bayer filter. Vertically binned column sums are re-weighted by wavelength to correct for the appropriate red, green and blue components within that column at that wavelength, to correct for the drift between red and blue response. Green signal is likewise corrected to provide an intensity normalization across the spectrum.

### 2.3.3. CORRECT\_SSC\_BIN\_2X2

SSC correction is applied as above, and then a Sliding Average applied after to smooth remaining irregularities.

### 2.3.4. BIN\_4X2

Pairs of (even, odd) original pixels are grouped as A (0 and 1), B (2 and 3), C (4 and 5), etc, for  $1952/2 = 976$  pairs. When vertically binned, each group will have 25% red, 25% blue, and 50% green.

A new spectrum will be constructed by averaging together adjacent groups, such that the new pixel 0 will contain  $(A+B)/2$ , pixel 1 will contain  $(B+C)/2$ , etc. Each averaged pixel will therefore contain 4 columns of source pixels at the same 25% / 25% / 50% ratio.

On an IMX385, the total number of averaged pairs (smoothed pixels) will be 975 ( $1952/2 - 1$ ).

### 2.3.5. BIN\_4X2\_INTERP

Same as BIN\_4X2, but then interpolate the reduced count of smoothed pixels back to the full detector pixel count using the configured wavelength calibration. This is valid when the original wavecal was performed using BIN\_2X2, as per-pixel wavelength values were already captured at intra-pixel boundaries (based on the average of adjacent pixels).

This provides a fully 4-pixel smoothed spectrum with the original number of pixels which matches the standard production wavelength calibration.

### 2.3.6. BIN\_4X2\_AVG

This is similar to other BIN\_4X2\_\* algorithms, in that "up to" 4 physical pixel columns can be averaged together into a single output pixel, but it is unique and different from any other procedure:

1. For every pair of adjacent (even, odd) physical pixels (e.g. (0, 1), (2, 3) etc, but not (1, 2) which starts with an odd index), average the TWO physical values together into one output pixel. That is, output pixel 0 is the average of physical pixels 0 and 1, output pixel 2 is the average of physical pixels 2 and 3, etc.
2. For every pair of adjacent (odd, even) physical pixels (e.g. (1, 2), (3, 4) etc, but not (2, 3) which starts with an even index), average the FOUR surrounding pixels together into one output pixel. That is, output pixel 1 is the average of physical pixels (0, 1, 2, 3), and output pixel 3 is the average of physical pixels (2, 3, 4, 5).

Using this algorithm, the following statements will be true:

- Every output pixel will contain an equal number of red and blue contributors.
- Every output pixel's "contributors" will begin with an even-numbered (0-indexed) physical pixel.
- Every output pixel's "contributors" will start on the same column color, with regard to the Bayer filter.
- The number of output pixels will match the number of physical input pixels (e.g. 1952 for an IMX385).
- The number of "contributing physical pixels" to output pixels will alternate across the output spectrum, with even-numbered output pixels being the average of TWO physical pixels, and odd-numbered output pixels being the average of FOUR physical pixels.

## 3. EEPROM Page Structure

The EEPROM for different models may contain 256 or 512 "pages" of 64 bytes each. On some models, part of the EEPROM is used for firmware code storage and is not available for data storage. On all models, at least 8 pages (512 bytes) are consistently available for use in standard spectrometer configuration. The EEPROM itself is physically either a Microchip [24LC128](#) (FX2 architecture, 16KB) or [AT24C256C-XHL-T](#) (ARM architecture, 32kB).

EEPROM pages are read and written as raw buffers by the firmware. Firmware on FX2-based models does not attempt to read or parse individual fields within the pages, therefore the internal format and structure of EEPROM pages can change and evolve over time without recompiling the firmware. This is not the case with ARM-based XS spectrometers, which do dynamically read and parse their EEPROMs at boot, typically to apply "startup" settings and default values to internal parameters.

The last byte of the first page (page 0, byte 63) is a "format" revision number describing the first 6 pages EEPROM (pages 0-5). The last byte of the 6<sup>th</sup> page (page 5, byte 63) is a "subformat" revision number for the following EEPROM pages (pages 6-7 and beyond).

Table 3 EEPROM Page Overview

Page	General Function
0	Device identification and features
1	Device calibration
2	Detector configuration
3	Lifetime usage statistics
4	Customer data
5	Bad pixel configuration
6	<i>Custom</i>
7	<i>Custom</i>

The following datatypes are referenced in the EEPROM field definitions:

- char[] — an ASCII string of the given maximum length. If a value less than the maximum is written to the field, at least one trailing null ('\0') should be used as a C-style string terminator. If the full field length is used, no terminating null is required.
- bool — although physically stored as a uint8 (unsigned char), field is logically a Boolean and only values of 0 and 1 are guaranteed supported.
- float32 — these are 4-byte IEEE 754 Float
- uint32 — unsigned long int
- uint16 — unsigned short int
- uint8 — unsigned octet; these values may be scalars or may be internally treated as enums (see relevant command documentation)

Table 4 EEPROM Page Format

Page	Size	Offset	Description	Format
0	64	0-15	Model name	char[16]
		16-31	Serial number	char[16]
		32-35	<i>Unused</i>	uint32
		36	Cooling available	bool
		37	Battery available	bool
		38	Laser available	bool
		39-40	Feature Mask (bitmask)	uint16
		41-42	Slit size in um	uint16

		43-44	Startup Integration Time (ms)	uint16
		45-46	X series: Startup Temperature in °C	int16
		47	Startup Triggering Mode	uint8
		48-51	Gain (for InGaAs: Even Pixel Gain) <sup>1</sup>	float32
		52-53	Offset (for InGaAs: Even Pixel Offset) <sup>2</sup>	int16
		54-57	Odd Pixel Gain (InGaAs systems only) <sup>2</sup>	float32
		58-59	Odd Pixel Offset (InGaAs systems only) <sup>2</sup>	int16
		60-61	XS series: Startup Laser TEC Setpoint (raw)	uint12
		62	<i>Unused</i>	
		63	EEPROM format revision = 17	byte

Page	Size	Offset	Description	Format
1	64	0-3	Wavelength calibration Coeff <sub>0</sub>	float32
		4-7	Wavelength calibration Coeff <sub>1</sub>	float32
		8-11	Wavelength calibration Coeff <sub>2</sub>	float32
		12-15	Wavelength calibration Coeff <sub>3</sub>	float32
		16-19	°C → DAC TEC Coeff <sub>0</sub>	float32
		20-23	°C → DAC TEC Coeff <sub>1</sub>	float32
		24-27	°C → DAC TEC Coeff <sub>2</sub>	float32
		28-29	T <sub>max</sub> (max TEC setpoint in °C)	int16
		30-31	T <sub>min</sub> (min TEC setpoint in °C)	int16
		32-35	ADC → °C Detector Temperature Coeff <sub>0</sub>	float32
		36-39	ADC → °C Detector Temperature Coeff <sub>1</sub>	float32
		40-43	ADC → °C Detector Temperature Coeff <sub>2</sub>	float32
		44-45	Thermistor Resistance at 298K	int16
		46-47	Thermistor Beta Value	int16
		48-59	Calibration Date	char[12]
		60-62	Calibrated By	char[3]
		63	<i>Unused</i>	

<sup>1</sup> InGaAs products use two registers for gain and two for offset. This allows for the even and odd pixels to be adjusted independently. All other products use the singular gain and offset register found on bytes 48 through 53

Page	Size	Offset	Description	Format
2	64	0-15	Detector Name	char[16]
		16-17	Active Pixels Horizontal	uint16
		18	Laser Warmup Time (seconds)	uint8
		19-20	Active Pixels Vertical	uint16
		21-24	Wavelength Calibration Coeff <sub>4</sub>	float32
		25-26	Actual Horizontal Pixels	uint16
		27-28	ROI Horizontal Start	uint16
		29-30	ROI Horizontal End	uint16
		31-32	ROI Vertical Region 1 Start	uint16
		33-34	ROI Vertical Region 1 End	uint16
		35-36	ROI Vertical Region 2 Start	uint16
		37-38	ROI Vertical Region 2 End	uint16
		39-40	ROI Vertical Region 3 Start	uint16
		41-42	ROI Vertical Region 3 End	uint16
		43-63	<i>Unused</i>	uint8[21]

Page	Size	Offset	Description	Format
3	64	0-10	<i>Unused</i>	uint8[11]
		11	maxLaserTempDegC	int8
		12-15	Laser Power mW → percent Coefficient 0	float32
		16-19	Laser Power mW → percent Coefficient 1	float32
		20-23	Laser Power mW → percent Coefficient 2	float32
		24-27	Laser Power mW → percent Coefficient 3	float32
		28-31	Maximum Laser Power (mW)	float32
		32-35	Minimum Laser Power (mW)	float32
		36-39	Excitation Wavelength (nm)	float32
		40-43	Min Integration Time (ms, 24-bit)	uint32
		44-47	Max Integration Time (ms, 24-bit)	uint32
		48-51	Average FWHM (nm or cm <sup>-1</sup> per Excitation)	float32
		52-53	Laser Watchdog Timer (sec on XS, count on FX2) (0x0000 or 0xffff will disable)	uint16
		54	Light Source Type (enum)	uint8
		55-56	Power Watchdog Timeout (sec)	uint16
		57-58	Detector Timeout (sec)	uint16
		59	Horizontal Binning Method (enum)	uint8
		60	Startup Scans to Average	uint8
		61	SML Attenuator DAC (10-40 dec)	uint8
		62-63	<i>Unused</i>	uint16

Page	Size	Offset	Description	Format
4	64	0-63	User Text String	char[64]

Page	Size	Offset	Description	Format
5	64	0-1	Bad Pixel 1 <sup>2</sup>	int16
		2-3	Bad Pixel 2	int16
		4-5	Bad Pixel 3	int16
		6-7	Bad Pixel 4	int16
		8-9	Bad Pixel 5	int16
		10-11	Bad Pixel 6	int16
		12-13	Bad Pixel 7	int16
		14-15	Bad Pixel 8	int16
		16-17	Bad Pixel 9	int16
		18-19	Bad Pixel 10	int16
		20-21	Bad Pixel 11	int16
		22-23	Bad Pixel 12	int16
		24-25	Bad Pixel 13	int16
		26-27	Bad Pixel 14	int16
		28-29	Bad Pixel 15	int16
		30-45	productConfiguration	char[16]
		46-51	Assembly Revision	uint8[6]
		52-62	<i>Unused</i>	uint8[11]
		63	Page 6 and 7 subformat	uint8

### 3.1. XS EEPROM Pages

All Wasatch Photonics spectrometers support the 8 pages (indexed 0-7) defined above. In addition, XS spectrometers support the following additional page definitions.

Page	Size	Offset	Description	Format
8	64	0-15	Laser Password	char[16]
		16-19	FeatureMaskXS	uint32
		20-63	<i>Undefined</i>	byte[44]

#### Proposed Fields

The following fields have been proposed but not yet approved or implemented:

- maxBatteryTempToFireLaser (Celsius, default 40)

<sup>2</sup> A value of -1 in any "Bad Pixel" field represents "N/A"; otherwise, the value indicates the (zero-based) pixel index that should be rejected (typically by averaging adjacent pixels in software)

## 4. Custom EEPROM Structure

The last two EEPROM pages can be configured to hold a variety of different fields and structures, depending on the sub-format code in the last byte of page 5:

Subformat	Page 6 and 7 Contents
0	User Data
1	NIST SRM Raman Intensity Calibration
2	Advanced Wavelength Calibration Spline
3	Untethered Configuration
4	Detector Regions ( <i>deprecated</i> )
5	Multi-Wavelength Raman
5-255	<i>Undefined</i>

### 4.1. Subformat 1: NIST SRM Raman Intensity Calibration

This format is used for Raman spectrometers for which a NIST-standard SRM Raman Intensity Calibration has been provided.

The first byte of the Raman Intensity Calibration section indicates the meaning of the following 7 floating-point fields.

- A value of 0 indicates no calibration is available.
- A value of 1-7 indicates a polynomial calibration of order  $n$ . That is, a value of 7 would indicate that the calibration is provided as a 7<sup>th</sup>-order polynomial, and that the next 8 floats represent coefficients 0 ( $x^0$ ) through 7 ( $x^7$ ).

Page	Size	Offset	Description	Format
6	64	0	Raman Intensity Calibration order	uint8
		1-4	Coeff 0	float32
		5-8	Coeff 1	float32
		9-12	Coeff 2	float32
		13-16	Coeff 3	float32
		17-20	Coeff 4	float32
		21-24	Coeff 5	float32
		25-28	Coeff 6	float32
		29-32	Coeff 7	float32
		33-63	<i>Unused</i>	

Page	Size	Offset	Description	Format
7	64	0-63	<i>Unused</i>	

In order to normalize precision, the polynomial has been curve-fit against log10 of the actual Raman intensity scaling factors. Therefore, to generate and apply the scaling factors for each

pixel, you would do something like the following in application code (example taken from Wasatch.PY's [wasatch.SpectrometerSettings.update\\_raman\\_intensity\\_factors](#)):

```
##
# @param eeprom (Input) populated wasatch.EEPROM object
# @returns Raman intensity correction factors (one per pixel)
#         as 1D numpy array
def expand_raman_intensity_factors(eeprom):
    if 1 <= eeprom.raman_intensity_calibration_order <= 7:
        coeffs = eeprom.raman_intensity_coeffs
        if coeffs is not None:
            try:
                factors = []
                for pixel in range(self.pixels()):
                    log10_factor = 0.0
                    for i in range(len(coeffs)):
                        x_to_i = math.pow(pixel, i)
                        scaled = coeffs[i] * x_to_i
                        log10_factor += scaled

                    expanded = math.pow(10, log10_factor)
                    factors.append(expanded)
                return numpy.array(factors, dtype=numpy.float64)
            except:
                print("error generating intensity factors")
    return None

##
# @param spectrum (Input) uncorrected Raman spectrum as 1D
#                 numpy array
# @param factors (Input) Raman intensity correction factors
#                 one per pixel) as 1D numpy array
# @returns corrected Raman spectrum as 1D numpy array
def apply_raman_intensity_factors(spectrum, factors):
    return spectrum * factors
```

## 4.2. Subformat 2: Advanced Wavelength Calibration Spline

This format is available for customers who wish to calibrate their spectrometer's wavelength x-axis using a cubic spline fit.

A typical n-point spline would include:

- n (the number of points in the spline)
- n floating-point wavelengths
- n floating-point y values
- n floating-point y2 values
- first and last valid wavelength (optional but recommended)

So a 12-point spline would need to store  $n$  itself (value 12, 1 byte),  $3n$  floats for the spline points (144 bytes), plus 2 min/max floats (8 bytes), for 153 bytes. That doesn't fit into two 64-byte pages, so for this feature we're rolling-in page 4 as well (normally reserved for User Data).

The spline wavecal would be expanded using the *splint()* function found in section 3.3 of Numerical Recipes in C (1986) (reproduced here for posterity), where:

- *xa* = the array of wavelengths used to generate the spline (e.g., array of the 12 Wavelength, values)
- *ya* = array of y-values
- *y2a* = array of  $y_2$  ( $y''$ ) values
- *n* = size of the arrays (e.g. 12)
- *x* = the wavelength for which you want the corresponding fractional pixel generated

(Note that spline wavecal is designed to generate pixel from wavelength, the opposite of our standard wavelength calibration generation.)

```
float splint(float *xa, float *ya, float *y2a, int n, float x) {
    int klo = 0;
    int khi = n - 1;

    while (khi - klo > 1) {
        int k = (khi + klo) >> 1;
        if (xa[k] > x)
            khi = k;
        else
            klo = k;
    }

    float h = xa[khi] - xa[klo];
    if (h == 0.0)
        throw("Bad XA input");

    float a = (xa[khi] - x) / h;
    float b = (x - xa[klo]) / h;
    return a * ya[klo]
        + b * ya[khi]
        + ((a*a*a-a) * y2a[klo] + (b*b*b-b)*y2a[khi]) * (h*h)/6.0;
}
```

Page	Size	Offset	Description	Format
6	64	0	Spline Point Count (0 - 14)	uint8
		1-3	<i>Unused</i>	
		4-7	Wavelength <sub>0</sub>	float32
		8-11	Y <sub>0</sub>	float32
		12-15	Y <sub>2</sub> <sub>0</sub>	float32
		16-19	Wavelength <sub>1</sub>	float32
		20-23	Y <sub>1</sub>	float32
		24-27	Y <sub>2</sub> <sub>1</sub>	float32
		28-31	Wavelength <sub>2</sub>	float32
		32-35	Y <sub>2</sub>	float32
		36-39	Y <sub>2</sub> <sub>2</sub>	float32
		40-43	Wavelength <sub>3</sub>	float32
		44-47	Y <sub>3</sub>	float32
		48-51	Y <sub>2</sub> <sub>3</sub>	float32
		52-55	Wavelength <sub>4</sub>	float32

Page	Size	Offset	Description	Format
		56-59	Y <sub>4</sub>	float32
		60-63	Y2 <sub>4</sub>	float32

Page	Size	Offset	Description	Format
7	64	0-3	Wavelength <sub>5</sub>	float32
		4-7	Y <sub>5</sub>	float32
		8-11	Y2 <sub>5</sub>	float32
		12-15	Wavelength <sub>6</sub>	float32
		16-19	Y <sub>6</sub>	float32
		20-23	Y2 <sub>6</sub>	float32
		24-27	Wavelength <sub>7</sub>	float32
		28-31	Y <sub>7</sub>	float32
		32-35	Y2 <sub>7</sub>	float32
		36-39	Wavelength <sub>8</sub>	float32
		40-43	Y <sub>8</sub>	float32
		44-47	Y2 <sub>8</sub>	float32
		48-51	Wavelength <sub>9</sub>	float32
		52-55	Y <sub>9</sub>	float32
		56-59	Y2 <sub>9</sub>	float32
		60-63	<i>Unused</i>	

Page	Size	Offset	Description	Format
4	64	0-3	Wavelength <sub>10</sub>	float32
		4-7	Y <sub>10</sub>	float32
		8-11	Y2 <sub>10</sub>	float32
		12-15	Wavelength <sub>11</sub>	float32
		16-19	Y <sub>11</sub>	float32
		20-23	Y2 <sub>11</sub>	float32
		24-27	Wavelength <sub>12</sub>	float32
		28-31	Y <sub>12</sub>	float32
		32-35	Y2 <sub>12</sub>	float32
		36-39	Wavelength <sub>13</sub>	float32
		40-43	Y <sub>13</sub>	float32
		44-47	Y2 <sub>13</sub>	float32
		48-55	<i>Unused</i>	
		56-59	Wavelength Minimum	float32
		60-63	Wavelength Maximum	float32

### 4.3. Subformat 3: Untethered Configuration

This format is used for spectrometers that can operate without active USB or BLE control.

Page 6 processing is exactly the same as subformat 1 (NIST SRM Raman Intensity Calibration).

Page	Size	Offset	Description	Format
7	64	0	libraryType	uint8
		1-2	libraryID	uint16
		3	<i>Unused</i>	uint8
		4	minRampPixels	uint8
		5-6	minPeakHeight	uint16
		7	matchThreshold	uint8
		8	libraryCount	uint8
		9	throwAwayCount	uint8
		10-63	<i>Unused</i>	

Pages 8-9 hold 16 byte character arrays that store loaded library names.

Page	Size	Offset	Description	Format
8	64	0-15	Library 1 Name	char[16]
		16-31	Library 2 Name	char[16]
		32-47	Library 3 Name	char[16]
		48-63	Library 4 Name	char[16]

Page	Size	Offset	Description	Format
9	64	0-15	Library 5 Name	char[16]
		16-31	Library 6 Name	char[16]
		32-47	Library 7 Name	char[16]
		48-63	Library 8 Name	char[16]

Pages 10-498 (61 pages (3904 bytes) per library entry x 8 entries, starting on page 10) are reserved for library data.

## 4.4. Subformat 4: Detector Regions

*Deprecated, re-define when appropriate.*

## 4.5. Subformat 5: Multi-Wavelength Raman

This subformat is for units with optics supporting multiple excitation wavelengths, and requiring storage for the 2nd excitation, 2nd wavecal, 2nd SRM calibration, 2nd vertical ROI, 2nd horizontal ROI etc.

Page 6 is necessarily identical to Subformat 1, to capture the "default" wavelength's Raman Intensity Calibration. All the overloadable calibrations are then stored on the single page 7, which can theoretically be duplicated in the future if designs permit more than two wavelengths.

Page	Size	Offset	Description	Format
7	4	0	2nd Excitation Wavelength (nm)	float32
	4	4	2nd Wavelength Calibration Coeff 0	float32
	4	8	2nd Wavelength Calibration Coeff 1	float32
	4	12	2nd Wavelength Calibration Coeff 2	float32
	4	16	2nd Wavelength Calibration Coeff 3	float32
	4	20	2nd Wavelength Calibration Coeff 4	float32
	2	22	2nd Horizontal ROI Start Pixel	uint16
	2	24	2nd Horizontal ROI End Pixel	uint16
	4	28	2nd average FWHM (nm or $\text{cm}^{-1}$ )	float32
	4	32	2nd Raman Intensity Calibration Coeff 0	float32
	4	36	2nd Raman Intensity Calibration Coeff 1	float32
	4	40	2nd Raman Intensity Calibration Coeff 2	float32
	4	44	2nd Raman Intensity Calibration Coeff 3	float32
	4	48	2nd Raman Intensity Calibration Coeff 4	float32
	4	52	2nd Raman Intensity Calibration Coeff 5	float32
	1	56	2nd Horizontal Binning Mode	uint8
	6	57	<i>reserved</i>	uint8[6]

## Proposed Changes

- Add OEM Accessory Connector boot configuration to EEPROM
- Add Etalon Calibration to EEPROM for XS (1952px x float32 = 7808 bytes = 122 pages)