Feature Identification Device (FID) EEPROM Specification

For Wasatch Photonics Spectrometers

Revision 1.12

Jun 28, 2021

Revision Log

Revision	Date	By	Reason
1.1	2017-25-1	J. Traud	Formatting, updated FID protocol to include coefficients on device.
1.2	2017-18-7	J. Traud	Added Excitation to page 1. Added bad pixel allotment to page 5.
1.3	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)
1.4	internal	M. Zieg R. Dickerson	• Added floating-point excitation wavelength • changed revisioning from per-page to entire EEPROM • changed ints to uint where negatives were invalid • moved min/maxIntegrationTimeMS to fit 24-bit range
1.5	internal	M. Zieg	added productConfiguration
1.6	2018-18-09	M. Zieg	added Raman intensity calibration
1.7	internal	T. Stohrer	added Average FWHM *(nm or cm^{-1} per Excitation)*
1.8	2020-03-23	M. Zieg	added page 6/7 subformat
1.9	internal	M. Zieg	Added FeatureMask
1.10	Feb 25, 2021	M. Zieg	Added laserWarmupSec, FeatureMask.gen15, and FeatureMask.cutOffFilterInstalled
1.11	Apr 14, 2021	M. Zieg	Added subformat 3
1.12	Jun 28, 2021	M. Zieg	Added HardwareEvenOddCorrection

Contents

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.

Table 1 FID VID and PID

VID	PID	USB Descriptor
0x24aa	0x1000	WP spectrometer with FX2 µController and Hamamatsu silicon detector
0x24aa	0x2000	WP spectrometer with FX2 µController and Hamamatsu InGaAs detector
0x24aa	0x4000	WP spectrometer with ARM µController

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. EEPROM Bitmasks

The following EEPROM fields represent bitmasks 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 EEPROM 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	hardwareEvenOddCorrection	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		*reserved*	
6	0x0040		*reserved*	
7	0x0080		*reserved*	
8	0x0100		*reserved*	
9	0x0200		*reserved*	
10	0x0400		*reserved*	
11	0x0800		*reserved*	
12	0x1000		*reserved*	
13	0x2000		*reserved*	
14	0x4000		*reserved*	
15	0x8000		*reserved*	

3. FPGA Compilation Options

The FPGA can be compiled to enable or disable different features. The FPGA Compilation Options register can be read to indicate which features are enabled. Below is the format for the register.

Table 2 FPGA Compilation Options Register

Bit	Description	Possible Values
[2-0]	Integration time resolution	0: 1 ms 1: 10 ms 2: Switchable between 1 us and 1 ms. *3-7: UNDEFINED*
[5-3]	Data header or tag	0: No header or tag 1: Ocean Optics header and tag 2: Wasatch tag *3-7: UNDEFINED*
[6]	Cf Select (Only available on InGaAs)	0: High-Gain mode not available 1: High-Gain mode available
[8-7]	Laser	0: No laser 1: Internal laser 2: External laser *3: UNDEFINED*
[11-9]	Laser control	0: Laser modulation 1: Laser transition points 2: Laser ramping *3-7: UNDEFINED*
[12]	Area scan feature	0: Not available 1: Available
[13]	Actual integration time feature	0: Not available 1: Available

[14]	Horizontal binning feature	0: Not available 1: Available
[15]	Undefined	Available for new features

4. Model Information

The EEPROM contains 512 pages of 64 bytes each, of which the first 8 pages (512 bytes) have been allocated for use in released spectrometer designs. The EEPROM is physically a Microchip AT24C256C-XHL-T (datasheet) with 32kB (32,768 bytes) capacity.

EEPROM pages are read and written as raw buffers by the firmware. The firmware 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.

The last byte of the first page (page 0, byte 63) is a format revision number for the first 6 pages EEPROM (pages 0-5). The last byte of the 6th page (page 5, byte 63) is a format 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	*Custom*
7	*Custom*

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
- byte — these values 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	Baud rate	uint32
		36	Cooling available	bool
		37	Battery available	bool
		38	Laser available	bool
		39-40	Feature Mask	uint16
		41-42	Slit size in um	uint16
		43-44	Startup Integration Time in ms	uint16
		45-46	Startup Temperature in °C	int16
		47	Startup Triggering Mode	byte
		48-51	Gain (for InGaAs: Even Pixel Gain)[1]	float32
		52-53	Offset (for InGaAs: Even Pixel Offset)[2]	int16
		54-57	Odd Pixel Gain (InGaAs systems only)[2]	float32
		58-59	Odd Pixel Offset (InGaAs systems only)[2]	int16
		60-62	*Unused*	
		63	EEPROM format revision = 10	byte

Page	Size	Offset	Description	Format
1	64	0-3	Wavelength calibration $Coeff_0$	float32
		4-7	Wavelength calibration $Coeff_1$	float32
		8-11	Wavelength calibration $Coeff_2$	float32
		12-15	Wavelength calibration $Coeff_3$	float32
		16-19	°C → DAC TEC $Coeff_0$	float32
		20-23	°C → DAC TEC $Coeff_1$	float32
		24-27	°C → DAC TEC $Coeff_2$	float32
		28-29	T_{max} (max TEC setpoint in °C)	int16
		30-31	T_{min} (min TEC setpoint in °C)	int16
		32-35	ADC → °C Detector Temperature $Coeff_0$	float32
		36-39	ADC → °C Detector Temperature $Coeff_1$	float32
		40-43	ADC → °C Detector Temperature $Coeff_2$	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	*Unused*	

[1] 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	laserWarmupSec	uint8
		19-20	Active Pixels Vertical	uint16
		21-24	*Wavelength Calibration Coeff$_4$*	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
		43-46	*Reserved: Linearity Coeff$_0$*[2]	float32
		47-50	*Reserved: Linearity Coeff$_1$*	float32
		51-54	*Reserved: Linearity Coeff$_2$*	float32
		55-58	*Reserved: Linearity Coeff$_3$*	float32
		59-62	*Reserved: Linearity Coeff$_4$*	float32
		63	*Unused*	

Page	Size	Offset	Description	Format
3	64	0-3	*Reserved: Device lifetime operation (minutes)*	uint32
		4-7	*Reserved: Laser lifetime operation (minutes)*	uint32
		8-9	*Reserved: Max laser temperature (°C)*	int16
		10-11	*Reserved: Min laser temperature (°C)*	int16
		12-15	Laser Power perc → mW Coefficient 0[3]	float32
		16-19	Laser Power perc → mW Coefficient 1	float32
		20-23	Laser Power perc → mW Coefficient 2	float32
		24-27	Laser Power perc → mW Coefficient 3	float32
		28-31	Maximum Laser Power (mW)	float32
		32-35	Minimum Laser Power (mW)	float32
		36-39	Excitation Wavelength (nm)[4]	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^{-1} per Excitation)*	float32
		52-63	*Unused*	

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

[2] Linearity Coeffs 0-3 have been used to provide laser power calibration in photodiode-equipped systems

[3] Not all spectrometers receive laser power calibration or min/max thresholds

[4] Floating-point version of excitation wavelength (nm) in version 4+

Page	Size	Offset	Description	Format
5	64	0-1	Bad Pixel 1[5]	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]
		46-62	*Unused*	
		63	Page 6 and 7 subformat	byte

5. 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
4-255	*Undefined*

5.1. 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^{th}-order polynomial, and that the next 8 floats represent coefficients 0 (x^0) through 7 (x^7).

[5] 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)

Page	Size	Offset	Description	Format
6	64	0	Raman Intensity Calibration format	byte
		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	*Unused*	

Page	Size	Offset	Description	Format
7	64	0-63	*Unused*	

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_ramman_intensity_factors(spectrum, factors):
    return spectrum * factors
```

5.2. 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), 3*n* 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 $Wavelength_i$ values)
- ya = array of y-values
- y2a = array of y2 (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)	byte
		1-3	*Unused*	
		4-7	$Wavelength_0$	float32
		8-11	Y_0	float32
		12-15	$Y2_0$	float32
		16-19	$Wavelength_1$	float32
		20-23	Y_1	float32
		24-27	$Y2_1$	float32
		28-31	$Wavelength_2$	float32
		32-35	Y_2	float32
		36-39	$Y2_2$	float32
		40-43	$Wavelength_3$	float32
		44-47	Y_3	float32
		48-51	$Y2_3$	float32
		52-55	$Wavelength_4$	float32
		56-59	Y_4	float32
		60-63	$Y2_4$	float32

Page	Size	Offset	Description	Format
7	64	0-3	$Wavelength_5$	float32
		4-7	Y_5	float32
		8-11	$Y2_5$	float32
		12-15	$Wavelength_6$	float32
		16-19	Y_6	float32
		20-23	$Y2_6$	float32
		24-27	$Wavelength_7$	float32
		28-31	Y_7	float32
		32-35	$Y2_7$	float32
		36-39	$Wavelength_8$	float32
		40-43	Y_8	float32
		44-47	$Y2_8$	float32
		48-51	$Wavelength_9$	float32
		52-55	Y_9	float32
		56-59	$Y2_9$	float32
		60-63	*Unused*	

Page	Size	Offset	Description	Format
4	64	0-3	$Wavelength_{10}$	float32
		4-7	Y_{10}	float32
		8-11	$Y2_{10}$	float32
		12-15	$Wavelength_{11}$	float32
		16-19	Y_{11}	float32
		20-23	$Y2_{11}$	float32
		24-27	$Wavelength_{12}$	float32
		28-31	Y_{12}	float32
		32-35	$Y2_{12}$	float32
		36-39	$Wavelength_{13}$	float32
		40-43	Y_{13}	float32
		44-47	$Y2_{13}$	float32
		48-55	*Unused*	
		56-59	Wavelength Minimum	float32
		60-63	Wavelength Maximum	float32

5.3. Subformat 3: Untethered Configuration

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

Page 6 and 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	scansToAverage	uint8
		4	minRampPixels	uint8
		5-6	minPeakHeight	uint16
		7	matchThreshold	uint8
		8-63	*Unused*	

Pages 8-9 are undefined in this subformat.

Pages 10-73 (64 pages, 4096 bytes total) are reserved for spectral data.