The os7500 Fabry-Perot Accelerometer

Overview and Guidelines for Use

1. Introduction and Theory of Operation

The os7500 accelerometer from Micron Optics is a special type of Fabry-Perot (F-P) sensor, consisting of an optical cavity that is coupled to a spring-mass system that deflects under an applied acceleration. Each os7500 has a special narrow wavelength range that it responds to, and several such sensors can be chained together on a single optical channel to work with swept-wavelength interrogators. The os7500 also employs mechanical filtering to help reduce the effect of aliasing that is common in passive optical devices.

A F-P interferometer consists of two partially reflective mirrors that bound an air cavity. Due to wave interference effects, the intensity of light that is reflected or transmitted through the cavity depends upon the length of the cavity and the wavelength of light. The governing equations for an ideal system are:

$$T = \frac{1}{1 - F\sin^2\frac{\varphi}{2}}$$
$$R = 1 - T$$
$$F = \frac{4R_0}{\left(1 - R_0\right)^2} \qquad (1)$$
$$\varphi = \frac{4\pi d}{\lambda}$$

Where T and R are the fraction of transmitted and reflected intensity, respectively; R_0 is the reflectance of the mirrors; λ is the wavelength of light and d is the cavity length. In practice, we can come very close to realizing the ideal response, but small deviations will still be present. Sources of such deviations include the finite spot size of the interrogating light, surface imperfections in the mirrors, and the finite thickness of the dielectric coating stacks required to make the mirrors. A theoretical optical reflectance response is shown in Figure 2.

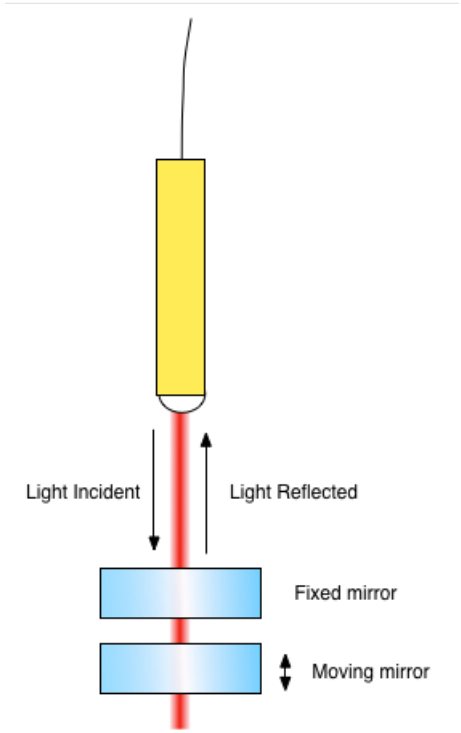

Figure 1. The basic design of a F-P sensor employs an optical input from a fiber collimator that is used to illuminate an optical cavity formed between two mirrors. One of these mirrors is movable, and can be excited with an external acceleration.

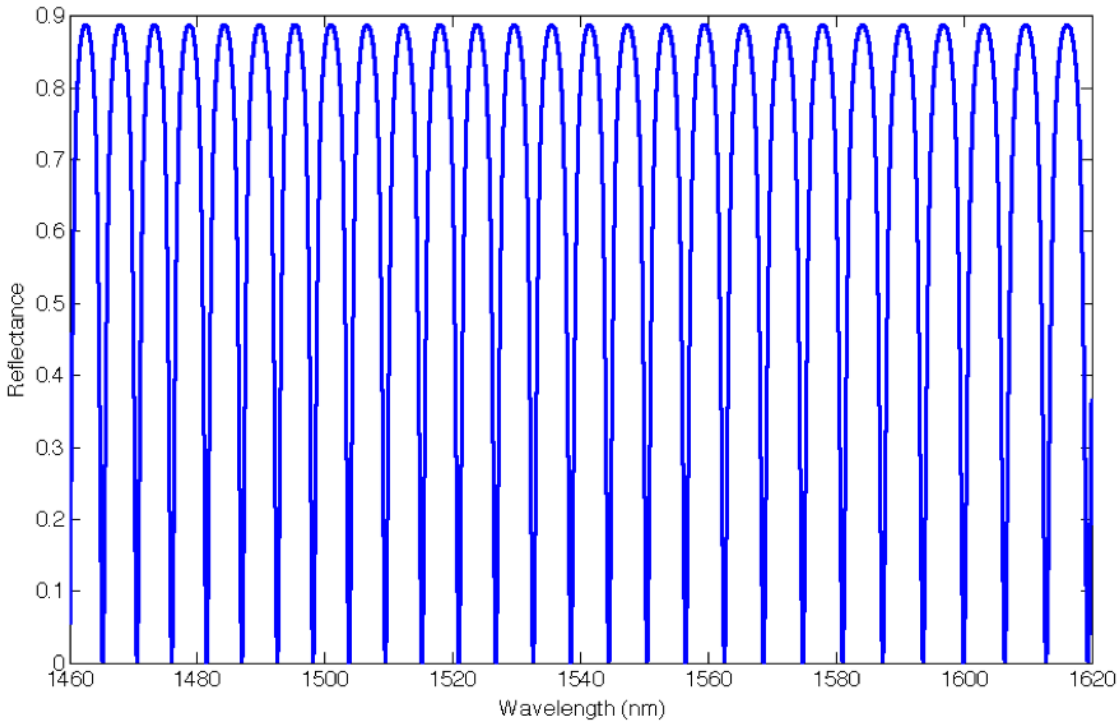

Figure 2. This is a theoretical response of a Fabry-Perot interferometer like the ones used in the in the os7500 sensors.

There are two reasons why this type of sensor can be advantageous in comparison to FBG based sensing:

- The optical fiber is no longer mechanically coupled to the sensing element. This means the sensing element can be optimized without consideration of the mechanics associated with the fiber. It also means that the strain limits in the fiber do not determine the dynamic range of the sensor.
- The cavity length can be optimized according to the application to give the desired free spectral range and sensitivity.

The os7500 devices comprise a special design that enables each sensor to only respond to a 20nm wavelength window. Wavelengths outside of a sensor's window are ignored and passed through from one fiber to the next in either direction. The optical response from the sensor is sent back out on the input fiber, while all wavelengths outside of the sensor's window are passed to the second fiber with minimal loss. With this patent pending design, it is possible to string several Fabry-Perot sensors on a single interrogator channel. Figure 3 shows a spectrum from a channel on a Si255 Hyperion instrument, where two sensors are connected in series.

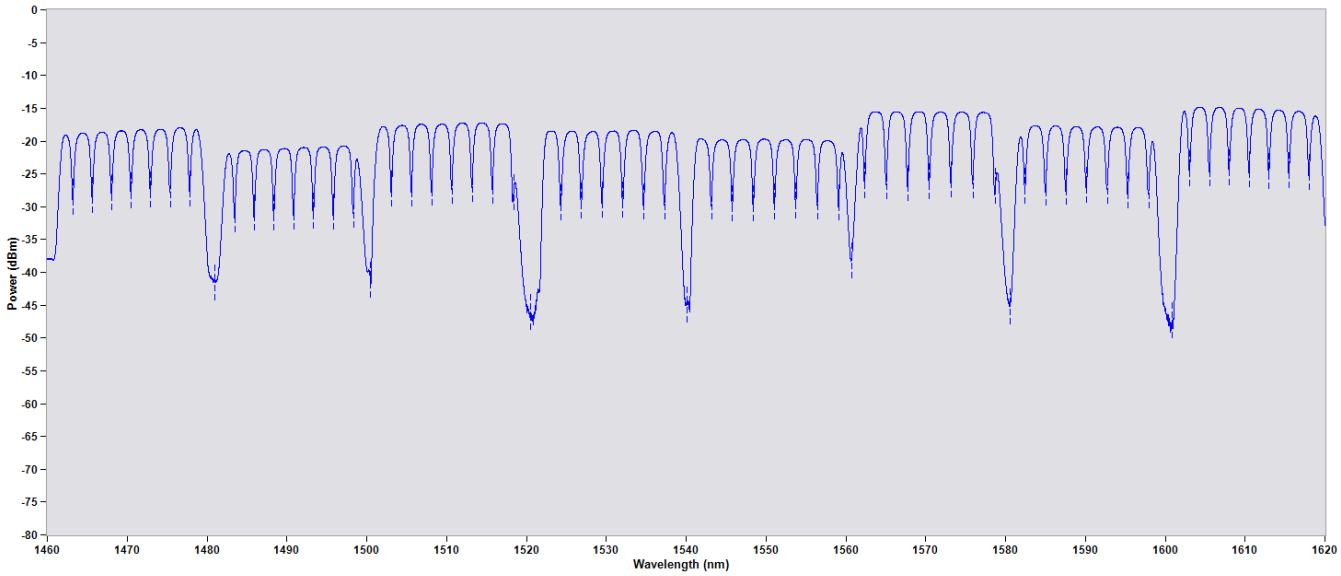

Figure 3. This is a full spectrum scan of a 160 nm SI255 Hyperion system from Micron Optics. There are 8 os7500 sensors attached together in series on this channel, with each sensor having a response within a 20 nm wavelength window.

When combined with a swept-wavelength interrogator from Micron Optics, it is possible to get an accurate dynamic measure of the cavity length of the accelerometer. The interrogator does this by detecting the minima, or valleys, in the reflected wavelength response. Changes in the wavelengths where the minima occur are caused by changes in the cavity length induced by applied external accelerations. An example algorithm optimized for deriving the acceleration time series based upon the sequence of observed minima is available from Micron Optics.

Software or firmware tracks the individual valleys within each sensor's window. Within a given scan, we are able to measure multiple valleys. Each such valley measurement is its own measurement of the cavity length, as the valley wavelengths are related to the cavity length by the relation

$$\lambda_n = \frac{2d}{n} \qquad (2)$$

Here, n is called the "order" of the valley. Our algorithm is based upon determining an appropriate value for n based upon the measured valley positions, and then use that to derive the value of the cavity length, d. In actual practice, there are deviations from the ideal, which means n may not necessarily be an integer. However, adjacent valleys are always integer 1 steps away from each other. So essentially, we seek to find an n value for the first valley observed, and then the subsequent valleys (ordered with increasing wavelength) are n-1, n-2, etc. This value of n is not determined every sample, but is instead a fixed quantity that can be learned by the software very rapidly. Once it is learned, it does not need to be updated so long as the software is able to continue to track which orders are within the window.

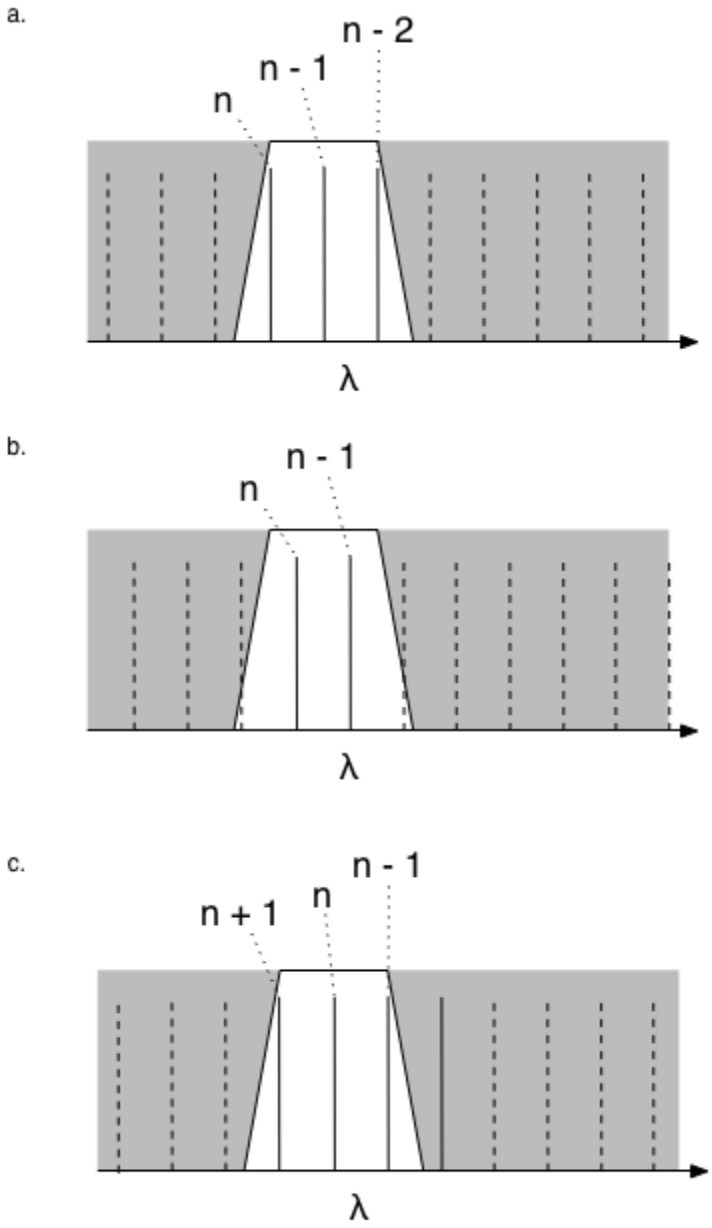

Figure 4. This figure illustrates how different optical orders can come and go within the wavelength window of a given sensor. In one sample, we have three orders within the window. In the next sample, one of the orders has moved beyond the upper filter band limit, and in the final sample a new order has entered from the lower end of the filter band. By tracking which orders are within the filter band, we can thus extend the dynamic range of the sensor.

Over time, it is possible for the accelerometer to encounter disturbances that result in changes to which orders are present within the window. It is possible for the software to detect these changes dynamically, which means it is possible to measure very large deviations. The only limitation on this is that the change from one sample to the next must be less than ½ of the wavelength separation of adjacent peaks, also know as the free spectral range (FSR). This limitation actually determines the peak operating range, but it also enables a very wide range of operation as the frequency is reduced. A sensor that is designed for only +/- 2.5 g at 300 Hz can accommodate up to 10 g at 50 Hz.
We can express this by noting that the above limitation is actually a limitation on the maximum velocity of the moving mirror, relative to the fixed mirror.

In terms of frequency of vibration, ω, we can relate the velocity to the applied external acceleration by including the sensitivity response function G, which relates changes in the cavity length, x, to applied accelerations, a

$$v(\omega) = \omega x = \omega \bullet a(\omega) \bullet G(\omega) \qquad (3)$$

And so we can then combine these to show that the maximum acceleration actually depends inversely on the frequency.

$$a_{\max}(\omega) = \frac{v_{\max}}{\omega G} = \frac{\lambda f_s}{4\omega S} \qquad (4)$$

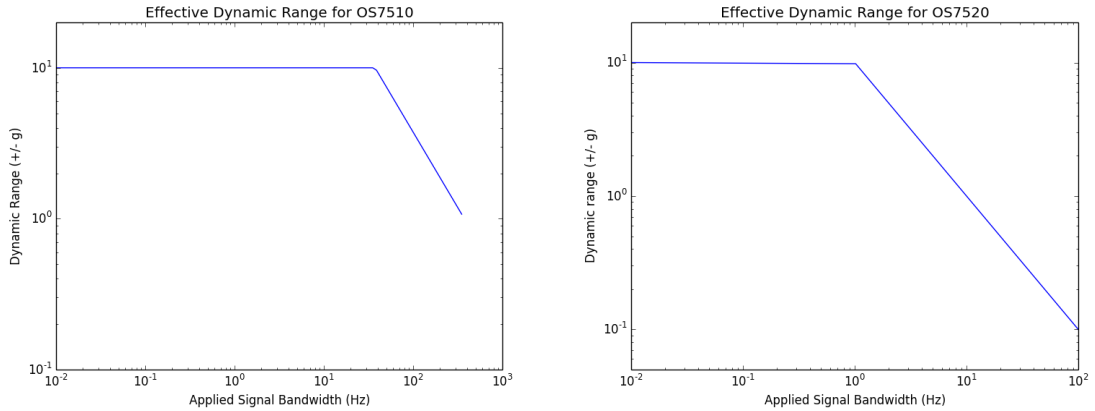

Figure 5. The effective dynamic range is actually a function of the applied signal bandwidth. For instance, if we have a pure sine wave acceleration at 10 Hz on the os7520, the peak to peak amplitude that we can measure will be 1 g, while at 100 Hz the maximum peak to peak amplitude is only 0.1 g. These figures are for reference only, please refer to the datasheet or the sensor test report for the specific sensor for exact specification of the peak amplitude.

The os7500 sensors have a response function that can be well approximated by a damped oscillator:

$$X(s) = \frac{\frac{1}{\omega_0^{\,2}}}{1 + \frac{1}{\omega_0 Q}s + \frac{1}{\omega_0^{\,2}}s^2} \qquad (5)$$

In this equation ω_0 is the fundamental resonant frequency of the accelerometer proof mass, typically about 2000 Hz for the os7510 and 350 Hz for os7520. The Q for the os7510 is approximately 10, while for the os7520 it is below 1.

The os7500 accelerometers are unlike any devices currently on the market. They can provide excellent quantitative vibration data, but they do have significant limitations when compared with off the shelf non-optical accelerometers. The most important distinction is that the device is completely passive. There are no active electronic components, and thus there is no feedback mechanism inside the sensor. The proof mass moves freely when an acceleration is applied. The second distinction is that the quantity that is changing in the accelerometer is the shape of the optical response curve. It is not possible to simply read out a single voltage or current and determine the applied acceleration. Instead, the sensor must be interrogated with an optical signal, and this signal must be analyzed in real time in order to produce a quantity that is proportional to the applied acceleration.

2. Sensing Algorithm

The optical readout involves the application of an algorithm. This algorithm takes the position of the defining spectral characteristics and produces an estimate of the cavity length. MOI supplies software which implements one such algorithm, which uses the position of the spectral valleys as detected by a si255 Hyperion instrument.

Any individual valley position will provide dynamic information about the cavity length, per equation (2). Multiple valleys yield multiple measures of the cavity length variation, and when the multiple measurements are averaged, we get the

benefit of reduced noise when compared with detecting just a single peak. In order to do this, however, we need to first have an estimate of the *n* order value, which means we need an estimate of the absolute cavity length. Our algorithm is able to calculate this, but the goal is not to produce an exact value of the cavity length, but instead to produce an estimate that minimizes the error in the dynamic sensitivity across all of the different valleys detected.

For instance, suppose we have a sensor that has *m* number of peaks within its wavelength band. We know that the actual cavity length can be determined from (2), but only if the order number *n* is known. Each individual valley yields its own estimate of the cavity length:

$$\hat{d}_i = \frac{\hat{n}_i \lambda_i}{2}, i = 1 \ldots m \qquad (6)$$

The order estimates are also constrained, as the order changes by exactly one from one valley to the next. The initial order is not constrained to be an integer, as there can be an arbitrary phase associated with the mirror coatings and other geometric factors. With this constraint, the cavity length estimates become:

$$\hat{d}_i = \frac{(\hat{n}_1 - i + 1)\lambda_i}{2} \qquad (7)$$

The average of these different estimates is then taken as the output from the algorithm.

This algorithm runs on the Hyperion Instrument, and the sensors interface on the Hyperion can be used for streaming acceleration data directly, as described below.

Note: As of Firmware Version 12.14, to be released in September 2017, improvements to the implementation of this algorithm will be released. These improvements are designed to be more robust to applied accelerations that exceed the range of the sensors, and also improves the accuracy of the initial order estimation. Earlier versions have an issue wherein they deliver values as NaN when the sensor has lost track of the correct order, and must iterate in order to return to a good state. This will no longer happen after the new release.

3. Hyperion Sensors API

The os7500 sensors are the first to use a new interface that is available on the Hyperion platform for streaming sensor data in engineering units. This data is streamed over a TCP port that does not interfere with the simultaneous streaming of peaks or spectra, and so it can be used simultaneously with other applications and with ENLIGHT. Currently, ENLIGHT does not use this sensors API, and so we do not have the ability to stream acceleration data with ENLIGHT, but this is a feature that is planned to be in ENLIGHT in late 2017.

MOI also makes available a Labview application that uses the sensors API to acquire and display data from os7500 sensors, which will be described below.

Add Sensor

Synopsis

Add a new sensor configuration to the instrument. The instrument will immediately begin calculating derived values for the sensor that can be read back on the sensor streaming port.

Usage

```
#AddSensor <name> <model> <channel> <distance> <wavelength_band> <calibration factor>
```

Description

Add a new sensor. The first field is the name, the second is the model (ie os7510), the third is the sensor channel number, the fourth is distance (in meters) and any additional parameters are sensor specific. For os7510 or

os7520 sensors, the additional parameters are the wavelength band and the calibration factor (in units of nm/g). Both of these are reported on the data sheet that is provided with each sensor.

Examples

Command:

```
#AddSensor os7510-1490-1 os7510 1 1 1490 62
```

Response:

```
A sensor with ID {f37f6882-209c-4de9-9e74-c87b2a98eb53} was successfully created and added to the instrument.
```

Response Content:

```
None
```

See Also

Python API method: `Hyperion.add_sensor`

LabVIEW API: `h.SENSORS.AddSensor.vi`

Known Issues

None.

Get Sensor Names

Synopsis

Get the list of sensors currently configured on the instrument.

Usage

```
#GetSensorNames
```

Description

Get the names of the sensors. The order of the names corresponds to the order of the data values on the streaming sensor data port. The names are space delimited in both the message and the content fields of the response.

Examples

Command:

```
#GetSensorNames
```

Response:

```
os7510-1490-1 os7510-1530-2
```

Response Content:

```
'os7510-1490-1 os7510-1530-2'
```

See Also

Python API method: `Hyperion.get_sensor_names`

LabVIEW API: `h.SENSORS. GetSensorNames.vi`

Known Issues

None.

Export Sensors

Synopsis

Outputs the detailed configuration information for each sensor.

Usage

`#ExportSensors`

Description

Export the sensors. The message contains readable descriptions of each sensor. The content contains this information in binary form to simplify parsing for custom applications. The binary output format is described below.

Parameter Name	Data Type	Byte Length	Description
headerVersion	U16	2	A unique identifier for the preset.
numberOfSensors	U16	2	The number of bytes in the string name that follows
Repeat Below pattern numberOfSensors times			
sensorBaseVersion	U16	2	Version number for the general sensors configuration.
id	[u8]	16	A unique identifier for the sensor. This is a 128 bit number, most easily parsed as an array of 16 bytes.
nameLength	U16	2	The length of the following name field, in bytes.
Name	string	nameLength	The assigned name for the sensor
modelLength	u16	2	The length of the following model field, in bytes
model	string	modelLength	The model name of the sensor. This must be a specific model name that is implemented on the hyperion (currently only `os7510` or `os7520` are acceptable)
channel	u16	2	Channel number on the instrument that the sensor is connected to.
distance	double	8	The distance, in m of fiber, from the instrument to the sensor.
Parameters below are specific to the sensor model (showing parameters for os7510 and os7520)			
sensorModelVersion	U16	2	Version number for the specific sensor model configuration
wavelengthBand	double	8	The center of the sensor wavelength band, in nanometers. os7510 and os7520 have bands that are spaced 20 nm apart, starting at 1470.0 nm and ending at 1610.0 nm.
calFactor	double	8	The calibration factor, expressed as nanometers of change in cavity length per applied acceleration in g.
reserved_1	double	8	Reserved values are used by MOI to tune the algorithm. These should not be altered unless directed by MOI.
reserved_2	double	8	
reserved_3	double	8	

Examples

Command:

```
#ExportSensors
```

Response:

```
[{'calFactor': 62.0,
  'channel': 2,
  'distance': 0.0,
  'id': [<16 bytes>],
  'model': 'os7510',
  'name': 'os7510-1530-2',
  'rcGain': 0.5,
  'rcThresholdHigh': 0.8,
  'rcThresholdLow': 1e-05,
  'version': 2,
  'wavelengthBand': 1530.0}]
```

Response Content:

```
See description
```

See Also

Python API method: `Hyperion.export_sensors`

LabVIEW API: `h.SENSORS.ExportSensors.vi`

Known Issues

None.

Save Sensors

Synopsis

Save the currently configured sensors in nonvolatile memory.

Usage

```
#SaveSensors
```

Description

Save the defined sensors. The sensor configurations will then persist after reboot.

Examples

Command:

```
#SaveSensors
```

Response:

```
'The sensors have been successfully saved.'
```

Response Content:

```
None
```

See Also

Python API method: `Hyperion.save_sensors`

LabVIEW API: `h.SENSORS.SaveSensors.vi`

Known Issues

None.

Remove Sensor

Synopsis

Remove a currently defined sensor by name.

Usage

```
#RemoveSensor <sensor name>
```

Description

The specified sensor is removed from the list of configured sensors, and data will no longer be output for this sensor.

Examples

Command:

```
#RemoveSensor 'os7510-1530-2'
```

Response:

```
"Sensor 'os7510-1530-2' was successfully removed."
```

Response Content:

```
None
```

See Also

Python API method: `Hyperion.remove_sensors (accepts a single name or a list of names)`

LabVIEW API: `h.SENSORS.RemoveSensor.vi`

Known Issues

This command did not function properly on firmware Version 12.12.0. This is fixed in all of the firmware

Get Sensor Value By Name

Synopsis

Get the most recent derived value for the sensor.

Usage

```
#GetSensorValueByName <sensor name>
```

Description

The value returned is the most recent sample, in the engineering units for the specified sensor.

Examples

Command:

```
#GetSensorValueByName 'os7510-1530-2'
```

Response:

```
'The current is 0.00273631538279475.'
```

Response Content:

```
<double precision number>
```

See Also

None.

Known Issues

None.

4. OS7500 Sample Data Acqusition Application

MOI provides a windows executable that can be used for demonstrating the performance of the os7500 sensors, and for saving data in a very basic text format. We will be offering additional software tools by early 2018, including the ability to acquire sensor data directly in ENLIGHT.

The x55 HYPERION LabVIEW os7500 FP Accelerometer Data Acquistion application can be downloaded from the MOI support page. Source code for this application is available upon request.

When the application is started, it opens on the setup tab, as shown below.

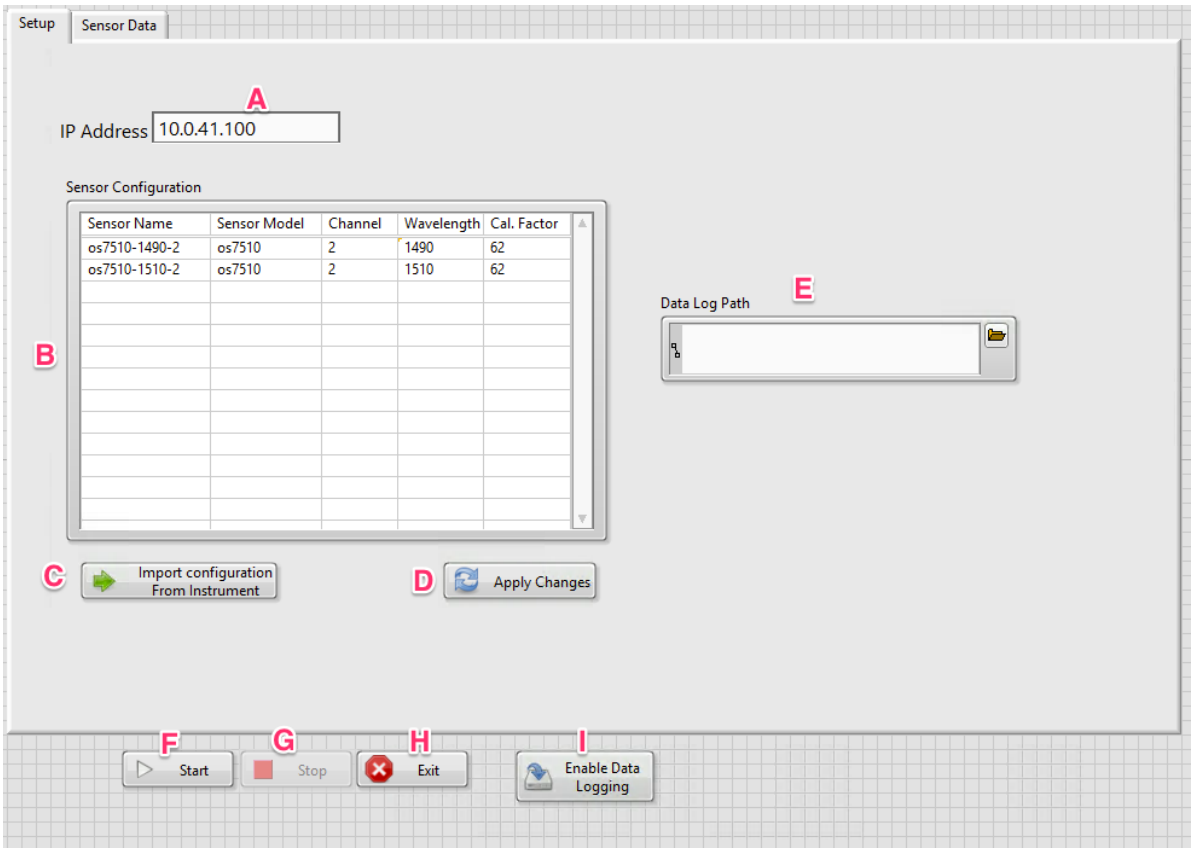

Figure 6.

The labeled components are described below.

A. **IP Address.** The instrument ip address must be entered correctly before any buttons are pressed (except for exit, which can be pressed at any time to exit the application).

B. **Sensor Configuration.** The list of defined sensors. If you already have sensors defined on the instrument, this table can be populated by pressing the button labeled "Import Configuration From Instrument". All five of the values shown must be entered for every sensor. Any changes made to this list will only be applied to the instrument after pressing "Apply Changes".

C. **Import Configuration From Instrument.** This populates the table with the list of existing sensors that have been defined. If no changes are needed, the "start" button can be pressed to begin acquisition.

D. **Apply Changes.** This applies any changes made to the table to the instrument. When this button is pressed, all existing sensors are deleted, and then the entire table is reloaded onto the instrument.

E. **Data Log Path.** Path to a file where data will be saved, if data logging has been enabled by pressing "Enable Data Logging"

F. **Start.** This begins acquiring streaming data and plotting it on the graph on the "Sensor Data" tab. This will also save data if "Enable Data Logging" has been pressed.

G. **Stop.** Stop acquisition without executing the program.

H. **Exit.** Exit the program.

I. **Enable Data Logging.** Enable logging of sensor data to a text file.

The Sensor Data tab is shown below.

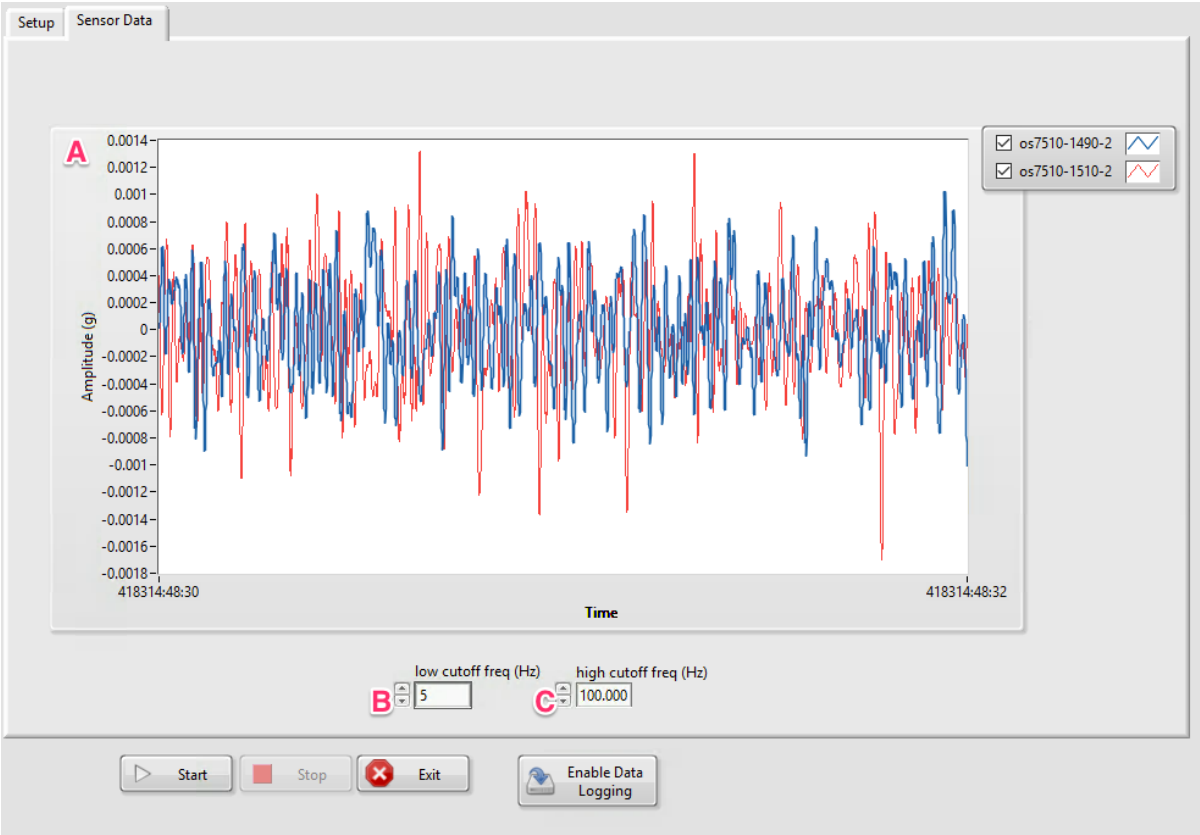

Figure 7.

The labeled components are described below.

A. Waveform Chart. This is a running strip chart of acquired data. All of the configured sensors are shown on this plot. Visibility of different sensors can be controlled with the check boxes on the legend. The data displayed is passed through a bandpass filter specified by the high and low cutoff frequencies. Most users find this filter to be useful for viewing data. Data that is saved to disk does NOT have this filter applied.

B. Low Cutoff Frequency (Hz). The low frequency cutoff for the bandpass filter.

C. High Cutoff Frequency (Hz). The high frequency cutoff for the bandpass filter.

5. Installation Considerations

While the os7500 is a robust sensor, it is not resistant to high g shocks (>500 g). When mounting to hard surfaces, it is very important to not drop the part or allow it to impact the surface. With minimal handling precautions, these parts should operate stably for many years.

The os7500 should be firmly fastened to the surface for which the acceleration measurement is needed. This can be achieved by using either a clamping mechanism, the #10-32 threaded screw hole, an adhesive, or some combination thereof. Prior to installation, make certain that the part is oriented appropriately for the acceleration to be measured. The part sensitive axis and sensor plane are shown in Figure 8. If clamping is used, it is advisable to distribute the clamping force across the width of the part, so that excessive stress is not being applied to a small area on the lid. A 10-32 UNF threaded hole is also available for mounting. For permanent installations, we recommend combining this with an adhesive that is appropriate for adhering the stainless steel case to the surface, such as DP-4XL from 3M. A three axis mounting

block, shown in Figure 9, is also available as an accessory. The accelerometers are mounted to the block using 10-32 screws. For permanent installations, detents in the block can be used to improve the effectiveness of the adhesive.

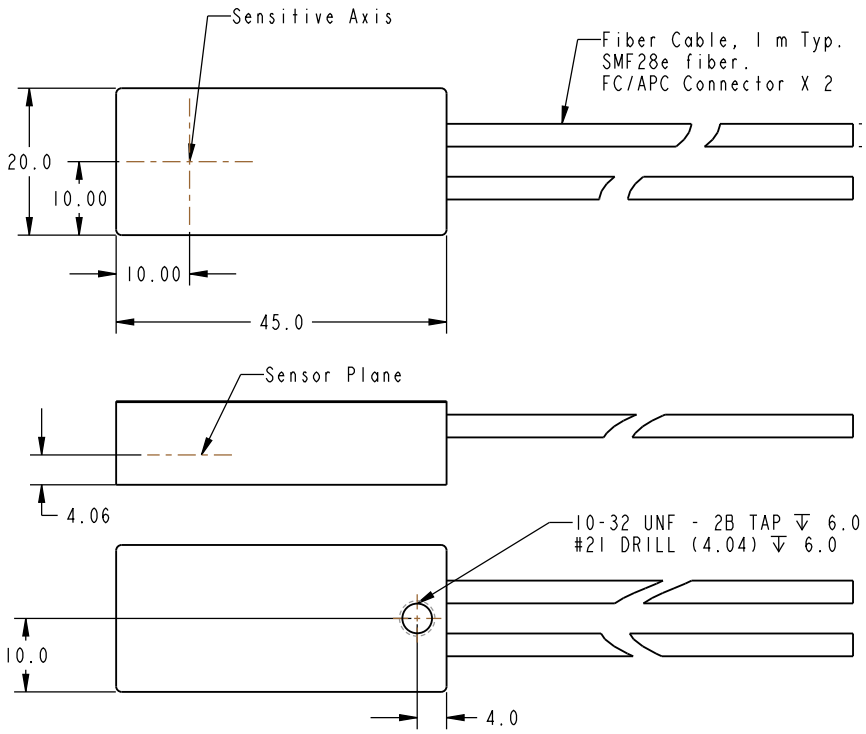

Figure 8. This illustrates the position of the sensor axis and sensor plane, relative to the outer dimensions of the sensor.

Figure 9. A three axis mounting block is available as an accessory.

6. General Usage Guidelines

There are a few important points to make about the intended areas of use for the os7500 series sensors. These should generally be considered as “vibration” sensors. They are not ideal for applications where DC precision is required, due to

the lack of an absolute DC reference combined with a relatively high thermal dependence. The sensors will provide reliable dynamic measurements at frequencies above 1 Hz in all situations. If the sensors are buried or placed in an otherwise temperature invariant environment, then the low frequency operation will certainly improve.

The sensors are not ideal for environments where the sensors need to remain continuously operational during periods of acceleration that exceed the operating range. The passive design combined with the measurement algorithm result in a sensor that cannot simply be driven into saturation when certain limits are hit. When applied accelerations exceed the specified peak ranges, the output will be erroneous and unpredictable.

7. Conclusion

The os7500 sensor platform enables all-optical high resolution acceleration measurement in a package that can be readily multiplexed with other sensors. This passive sensor differs from conventional accelerometers in a number of critical ways that have been presented above. Micron Optics is committed to helping customers realize the full value of this new sensor.