

API for ESI-USB Sensor

Introduction

The application programming interface or API for the ESI-USB pressure sensor is a Windows DLL (dynamic link library) written in C++. The library provides a set of functions that may be used to integrate the use of the sensors into client applications. This library can be called from any language or programming environment that can make use of DLLs. This includes C#, C++, VB.NET and LabVIEW.

Error Codes

The following error codes can be returned by the library functions.

Value	Name	Description
0	OK	The operation succeeded.
-1	FAIL	The operation failed.
-2	INVALID_INDEX	The operation failed due to an invalid sensor index.
-3	INVALID_PARAMETER	The operation failed because one of the parameters is out of range or otherwise invalid.
-4	INVALID_STATE	The operation failed because the sensor is in an invalid state. This will occur, for example, if a reading is attempted from a sensor that is not in use.

Library Functions

This section lists the functions that will be implemented by the library. The client code will control the sensor by calling these functions in the appropriate order.

Set-Up and Initialisation

Name FindSensors
Parameters int* SensorCount
Returns A status code showing success or error.
Comments This is called to find connected sensors. This must be the first call made to the library by the client application. The function first finds which serial ports are present in the system. For each port it attempts to open and read data back from a pressure sensor. The value of the SensorCount parameter is set to the number of sensors that were detected. The details of the sensors can then be retrieved. Note that only sensors that are not already being used by another application will be detected. When this function returns the library is not using any of the sensors. UseSensor must be called before any readings are taken.

Name FindSensorsEx
Parameters int SensorType, int* SensorCount
Returns A status code showing success or error.
Comments This is the same as FindSensors except for the SensorType parameter. This may be set to zero to only find slow sensors or to one to find fast sensors.

Name GetSensorInfo
Parameters int SensorIndex, int* PortNumber, char* SerialNumber, int SerialNumberLength
Returns A status code showing success or error.
Comments This function returns the serial port number and serial number of a particular sensor. The SensorIndex is a zero-based index (if

there are two sensors detected then valid indices are zero and one) specifying which of the sensors found by FindSensors is intended. The serial port number is returned in the PortNumber parameter. SerialNumber is a character array and SerialNumberLength is the number of characters in the array. This should be at least eight characters long for the current length of serial numbers.

Name GetSensorInfoEx
Parameters int SensorIndex, int* PortNumber, char* SerialNumber, int SerialNumberLength, int* IsFast
Returns A status code showing success or error.
Comments This is the same as GetSensorInfo except for the addition of the IsFast parameter. This is set to one to indicate the sensor is a fast sensor and zero if it is not.

Name UseSensor
Parameters int SensorIndex
Returns A status code showing success or error.
Comments This function tells the library that the specified sensor is to be used. When this is called the library opens communications with the sensor over the corresponding serial port. An error will be returned if the sensor is already in use by the library.

Name IsSensorUsed
Parameters int SensorIndex, int* Used
Returns A status code showing success or error.
Comments This function reads back whether a particular sensor has been selected to be used with the library. The Used parameter is set to one if the sensor is in use and zero if it is not.

Name ReleaseSensor
Parameters int SensorIndex
Returns A status code showing success or error.
Comments This function releases a specified sensor. The communications with the sensor are closed and it could then be used by another application if required. An error will be returned if the sensor is not in use by the library.

Information

Name GetAPIVersion
Parameters char* Version, int Length
Returns A status code showing success or error.
Comments This function returns the version of the API in the Version parameter. The Version parameter is an array of characters. The Length parameter contains the array length. This should be long enough to hold the version information.

Name GetPressureRange
Parameters int SensorIndex, float* Range
Returns A status code showing success or error.
Comments The SensorIndex specifies the sensor in question. The value of the Range parameter is set to the full range of the sensor in bar.

Name GetPressureUnits
Parameters int Units, char* UnitString, int UnitStringLength
Returns A status code showing success or error.

Comments This function is given a pressure units code and returns a string containing the units in text form in the UnitString parameter. The UnitStringLength parameter is the number of characters in the UnitString array. The units codes are as below:

0	bar
1	mbar
2	psi
3	MPa
4	Pa
5	mm H ₂ O
6	mm Hg
7	atm
8	km cm ²
9	kPa

Name GetTemperatureUnits
Parameters int Units, char* UnitString, int UnitStringLength
Returns A status code showing success or error.
Comments This function is given a temperature units code and returns a string containing the units in text form in the UnitString parameter. The UnitStringLength parameter is the number of characters in the UnitString array. The units codes are as below:

0	Celsius
1	Kelvin
2	Fahrenheit

Name GetManufactureDate
Parameters int SensorIndex, long long* date
Returns A status code showing success or error.
Comments This function gets the date of manufacture for the specified sensor. The returned value is the number of 100-nanosecond intervals that have elapsed since January 1, 0001 at 00:00:00:000 in the Gregorian calendar.

Name GetCalibrationDate
Parameters int SensorIndex, long long* date
Returns A status code showing success or error.
Comments This function gets the date of calibration for the specified sensor. For details on the returned value see GetManufactureDate.

Name GetManufactureLabVIEWDate
Parameters int SensorIndex, long* date
Returns A status code showing success or error.
Comments This is the same as GetManufactureDate except the time is returned as the number of seconds since the start of 1904 for compatibility with LabVIEW.

Name GetCalibrationLabVIEWDate
Parameters int SensorIndex, long* date
Returns A status code showing success or error.
Comments This function gets the date of calibration for the specified sensor. For details on the returned value see GetManufactureLabVIEWDate.

Operation

Name Read
Parameters int SensorIndex, int Units, bool Absolute, double Temperature,

Returns	float* Pressure
Comments	A status code showing success or error. This function reads and returns the pressure in the selected engineering units. The SensorIndex specifies the sensor to be read. The Units parameter selects the choice of units from the list given above for the GetPressureUnits function. The Absolute parameter selects whether to return an absolute or a gauge reading. If Absolute is zero then a gauge value is returned, otherwise an absolute value is returned. The Temperature parameter is the temperature in Celsius. The measured pressure in the specified units is returned in the Pressure parameter. The library must have first called UseSensor for the specified sensor before a reading can be taken.
Name	ReadTemperature
Parameters	int SensorIndex, int Units, float* Temperature
Returns	A status code showing success or error.
Comments	This function reads and returns the temperature in the selected engineering units. The SensorIndex specifies the sensor to be read. The Units parameter selects the choice of units from the list given above for the GetTemperatureUnits function. The temperature is returned in the final parameter. The library must have first called UseSensor for the specified sensor before a reading can be taken.
Name	SetGaugeDifferential
Parameters	int SensorIndex, double Differential
Returns	A status code showing success or error.
Comments	This sets the difference between absolute and gauge pressures in bar. The SensorIndex specifies the sensor in question. The value is used in the Read function when giving absolute values.
Name	StartMonitoring
Parameters	double PressureInterval, double TemperatureInterval
Returns	A status code showing success or error.
Comments	Fast sensors can be set to continually read and store pressure and temperature values. Calling this function starts the monitoring process for all sensors currently being used by the library. The pressure interval is in seconds. It must be set in the range 0.001 to 0.2 seconds. The temperature interval is also in seconds. This should be in the range 0 to 25.5 seconds. A value of zero indicates that temperature isn't to be read during the monitoring process. Once the monitoring process is underway, GetMonitorData is called to get the readings and StopMonitoring is called to stop the acquisition.
Name	StopMonitoring
Parameters	None
Returns	None.
Comments	Stops a monitoring session previously started with StartMonitoring.
Name	GetMonitorData
Parameters	int SensorIndex, float* Time, float* Pressure, float* Temperature, int* Length, int Units, int Absolute, int Units2
Returns	A status code showing success or error.
Comments	During the monitoring process the library reads back and stores

the pressure and temperature values. This function is called to retrieve the values for a particular sensor. The calling code would need to periodically read back the data for each sensor. The SensorIndex parameter sets which sensor's data is required. The Time, Pressure and Temperature parameters are arrays which will be filled with the sensor data. The time is the time in seconds since measurement was begun. The Length parameter contains the length of the time, pressure and temperature arrays. On return it is set to the number of readings that were returned. If the input arrays are longer than the number of readings available then all data will be returned. If the arrays are shorter than the number of readings available then the arrays will be filled and there will still be data stored within the library. If this function is called when there are no readings available the length will be set to zero. Units sets the required pressure units. Units2 sets the required temperature units. Absolute sets whether absolute or gauge readings are required.

Name GetActualRate
Parameters double RequestedRate, double* ActualRate
Returns A status code showing success or error.
Comments The sensor can only run at certain discrete sample rates. When called this function sets the ActualRate to the rate in Hertz that will be achieved for the given RequestedRate. RequestedRate is also in Hertz.

Name ZeroSensor
Parameters Int SensorIndex, int Count, double Interval
Returns A status code showing success or error.
Comments This function sets the pressure input of the specified sensor to zero. A number of readings are taken and their average is then used to adjust the calibration coefficients stored on the sensor. SensorIndex selects the sensor. Count sets the number of measurements to take and Interval is the time in seconds between measurements. Suggested values are to set Count to 10 and Interval to 0.2. The current pressure reading must be within 10% of the sensor's full scale reading. If it is outside this range then INVALID_STATE will be returned.

Clean-Up

Name CleanUp
Parameters None
Returns A code indicating error or success.
Comments This function must be called when the application code has finished using the library. This closes communications and cleans up all system resources.