Programming Guide to the Oak Sensor Family

Application Note

Revision history

Date	Doc. Rev.	Changes
01-Jul-2007	Rev. 1.0	Initial Release

Content

1. Definitions

Data Transfer direction:

In the context of USB transfer directions are always regarded from the Host (PC) point of view:

IN Device à Host

OUT Host à Device

Endpoint

An Endpoint can be looked at as a communication channel. Oak Sensors have two Endpoints: An Interrupt IN endpoint for real time data and a Control Endpoint (often referred as Endpoint 0 or EP0) for configuration data which is not time critical.

Reports

USB information is transferred in packets with a fixed structure, called "Report". In the case of HID devices there are two kinds of reports used:

Interrupt report	These reports have a fixed reserved bandwidth, and therefore a guaranteed latency of no more than 1ms. All Oak Sensors use interrupt reports to transfer the measured sensor values.
Feature report	These reports are transferred through a special control communication pipe. They have no guaranteed bandwidth thus they also have not real-time behavior. All Oak Sensors use Feature reports to send/receive configuration parameters.

SI Units

The International System of Units (abbreviated SI from the French Le Système international d'unités) is the modern form of the metric system. It is the world's most widely used system of units, both in everyday commerce and in science.

It describes all units derived from the seven base units meter, kilogram, second, ampere, kelvin, candela and mole. For angular dimensions, the radian is used to avoid confusion, since angular figures are dimensionless by definition.

2. Software Functionality

In general the software can be divided into the following tasks

1. Find one or multiple appropriate Oak sensors.
   This is usually done during the initialization of the application.
2. Detect insertion / removal of any sensor at runtime (Hot Plugging)
3. Retrieve information about the device and data channels (e.g. data format)
4. Read captured sensor values
5. Read / write sensor configuration parameters

Depending on the application and hardware setup, some of these tasks may be omitted. If for example the sensor will never be removed from the host computer, hot plug functionality does not have to be implemented.

2.1. Detecting Already Connected Devices

When starting the application, it usually scans all USB devices to detect if one or more suitable devices are already connected.

In Windows operating systems, the result of this operation is a DevicePath to each suitable device. A DevicePath is a string representing one specific instance of a USB device.

If you always use the same host computer with the same Oak sensor, you could treat the DevicePath to be constant. However, we strongly discourage this practice, since it prohibits for example to exchange an Oak sensor by another, identical one.

2.2. Hot Plugging

All USB devices allow Hot Plugging, which is connecting / disconnecting the device while the application is running.

If your application needs to support this feature, it needs to be informed, whenever a USB device is attached or removed from the bus.

In Windows operating systems, the result of this operation is a DevicePath to each suitable device. A DevicePath is a string representing one specific instance of a USB device.

Although the result of this process is the same as described above in section 2.1, Windows offers a completely different API to accomplish these two tasks.

2.3. Retrieve Information About the Device and Data Channels

Almost all information about the sensor values can be retrieved through predefined Windows API calls. This includes the complete device ID (Vendor ID; Product ID; Revision; Serial Number), name of the device and each single channel, data format of each data channel (bit width, signed/unsigned, scaling factor, and physical SI unit).

Toradex has made some proprietary extensions to these standards:

- The channel name always ends with the pretty name of the SI unit, enclosed in square brackets.
  For example the unit for pressure is named [Pa], instead of [$kg \cdot m^{-1} \cdot s^{-2}$]
- Each Oak Device offers a user-programmable name for the whole device, as well as for each single channel.

2.4. Read Captured Sensor Values

Retrieving sensor values means that the application receives IN reports. (Remember, IN means Device à Host). There are two ways to do that:

a) The application requests the IN report just in time through the control endpoint.
Since the control endpoint has no real time capabilities, there is no guaranteed latency.

b) The application uses the buffered interrupt IN report pipe.
This is the preferred method, as it comprises several advantages: As Interrupt reports have a guaranteed latency of only 1ms, it can be used for real time operations. Buffering assures that no data gets lost or is read twice. Report buffers are duplicated for each application requesting the report, hence it is possible to run multiple applications using exactly the same data received from the USB device.

For Windows operating systems, method a) is only supported by Windows XP and later – there is a special API function for this.
For method b) Windows offers a streaming interface. This means an application handles USB sensor data like a file; first it opens the device (using the DevicePath), then data can be read using a ReadFile command, and finally the application should close the device.

The data read can be easily transformed into real physical values, using the information described in section 2.3 above.

2.5. Sensor Configuration

Sensor configuration is written and read by sending and receiving Feature Reports. Windows has a special API call for this operation.

Feature Reports are part of the USB standard, but there is no standard about how to establish a bidirectional communication. Thus Toradex had to define a proprietary protocol.

In contrast to reading sensor values through Interrupt reports, Feature report communication is not multithreading-save, i.e. only one application is allowed to exchange Feature reports with a specific sensor at a time.

2.5.1 General report format

All Oak sensors use Feature reports of 1+32 Bytes length. The first byte is not really part of the report, but it is the report number. It is always 0 for all Oak sensors.

Feature Out reports

Byte	Name	Description
0	Rpt#	Report number, always 0 for all Oak sensors
1	GnS	0 = Set configuration parameter 1 = Get configuration parameter
2	Tgt	Location of the parameter 0 = RAM (volatile parameter) 1 = Flash (nonvolatile parameter) 2 = CPU 3 = Sensor 4 = other
3	Size	Number of Data Bytes
4	Index LSB	Index to the selected configuration parameter e.g. parameter number, or sensor register address
5	Index MSB	
6-32	Data	Parameter value. The exact meaning depends on the parameter. Details can be found in the sensor datasheet.

Feature In reports

Byte	Name	Description
0	Rpt#	Report number, always 0 for all Oak sensors
1	Status	0xFF = report contains valid data, Oak device is ready to receive a new command all other values = report contains invalid data. This happens typically when the feature In report is requested, before the Oak device had enough time to prepare the data.
2-32	Data	Parameter value. The exact meaning depends on the parameter. Details can be found in the sensor datasheet.

2.5.2 General Algorithm to write a Parameter

To avoid timing problems we recommend to use the following algorithm to write configuration parameters:

Repeat
    Read Feature In Report
    (optional delay)
Until Byte 1 of the report is 0xFF → this is, the Oak device is ready to receive a new command through a feature out report
Write Feature Out report → The actual transfer of the command
Repeat
    Read Feature In Report
    (optional delay)
Until Byte 1 of the report is 0xFF → for safety, wait until the Oak device has finished processing this command.

2.5.3 General Algorithm to read a Parameter

Repeat
    Read Feature In Report
    (optional delay)
Until Byte 1 of the report is 0xFF → this is, the Oak device is ready to receive a new command through a feature out report
Write Feature Out report → transfer the command to read a parameter
Repeat
    Read Feature In Report
    (optional delay)
Until Byte 1 of the report is 0xFF → wait until the Oak device has finished processing this command. Now the report contains the requested parameter

3. What Information is Stored Inside the Oak Device

3.1. USB / HID Standardized Information

3.1.1 Product Information

VID (Vendor ID)	this is 0x1b67 for all Toradex products
PID (Product ID)	16 bit value which is constant for a particular product
REV (Revision ID)	16 bit value indicating the revision status of the hardware
SN (Serial number)	This number is unique within the series of a particular product
	The combination of the four parameters above leads to an identification of a USB product that is unique worldwide.
Device Name	A friendly name that describes the functionality of the USB device. Unfortunately Windows XP displays this information only during the first time connection of the device. The device manager ignores this information and displays only a general "USB Human Interface Device".
Current consumption	Maximum current consumption of the USB device (0..500mA)

3.1.2 Data Channel Information

For each data channel the USB device provides information about the data stored. Oak sensor devices always provide data in SI units. So temperatures are reported in kelvin, distances in meters, acceleration in m/s^2, etc.
Values are reported as integers, with an associated decimal exponent, so the format is $i \times 10^e$.

Usage	What the data is used for. For common devices like keyboards, mice and game pads, the USB standard defines fixed codes for the most common functions. For sensor applications, only a few codes are defined. So the usage of in Oak sensor devices is often declared to be "vendor-defined".
Channel Size	Number of bits. At the time of writing this document, all Oak sensor channels are 16 bits in width.
Value Range	Minimum and maximum valid value. This can be less than the full range that could be represented with the available number of bits. It also defines if the channel data is signed (minimum less than zero) or unsigned.
Unit	This is a USB-standardized form to define how the channel's physical unit is derived from the basic SI units.
Unit Exponent	A decimal exponent that is needed to bring the reported integer sensor value into the correct range (refer to the introducing text at the beginning of this section).
Channel Name	A friendly name that describes the content of one channel.

3.2. Toradex Extensions

3.2.1 User Device Name

The user can define a name of up to 20 characters in length for the device. The name can be stored in RAM or nonvolatile in the Oak device's flash memory.

3.2.2 User Channel Name

The user can define a name of up to 20 characters in length for each channel. The name can be stored in RAM or nonvolatile in the Oak device's flash memory.

3.2.3 Friendly Unit String

Toradex has defined a special convention on the naming of each channel (see "Channel Name" in section 3.1.2):

The channel name always ends with a friendly user string, enclosed in square brackets. For example "Pressure [Pa]": your application can retrieve the common unit Pascal (Pa), instead of the bare SI base units [$kg \cdot m^{-1} \cdot s^{-2}$]. Dimensionless units are represented as "[1]".

4. The Oak Helper Library for User Applications

To keep the user from the burden of learning the Windows API functions to communicate with Human Interface Devices, Toradex has developed a library that greatly simplifies that task.

The library comes as a DLL, along with the according linker (.lib) file.

It was written in C/C++, and at the time of writing this document, only a C header file is available. However it should be quite a simple task to translate that interface to any other language.

The library was compiled in two different versions:

- "oaka.dll" has all strings in the interface defined to be standard null-terminated C strings (ASCII).
- "oakw.dll" has all strings in the interface defined to be standard null-terminated Unicode strings.

Please take care to choose the correct library, depending wheter you have chosen to compile your application in ASCII or Unicode mode. The header file (.h) is the same for both revisions of the library.

For documentation please refer to the header file, and to the provided source code examples that make use of this library.

5. References

Windows Driver Development Kit (DDK)
http://www.microsoft.com/whdc/devtools/ddk/default.mspx

You do not have to write a driver to access the Oak sensors from your own application. However the DDK contains an example how to handle communication with USB HID devices (which is what Oak sensors are).