Optical Switch FOD5508 USB Communication Protocol

10 July 2015

Version 2 Revision 0

1 Summary

This document defines the communication protocol between USB Host and Optical Switch FOD5508 Device with USB port.

2 Contents

3 List of Figures

4 List of Tables

5 USB Information

Optical Switch **Device** has Lifodas Vendor ID VID=0x273e and assigned Product ID PID=0x0007. Communication Version is 0x0100 and means “01.00”.

Optical Switch Device has Endpoint0 for Control Transfers and one Interrupt IN Endpoint (Endpoint1 with address 0x01) for HID asynchronous events.

The functionality of USB devices is defined by class codes, communicated to the USB host to affect the loading of suitable software driver modules for each connected device.

Optical Switch Device has a **HID** class code.

The various operating systems and, in particular, the Windows operating systems, provide drivers for HID-class devices, which can be used for communication with these devices. Using these drivers is advantageous, because this way the need to write drivers for communication with those devices is eliminated.

A transfer represents several transactions that create a data set with a particular structure and meaning for the device. With HID-class devices, a transfer is called *report*. The report data have a certain structure, which is specified in the report descriptors

A report descriptor contains articles that describe the size and structure of the data reported. Such a descriptor provides information about the data reported by the device for each of its functions and about the data intended for the device’s functions. Examples of information are the size of data returned, whether the data are absolute or relative, and the minimum and maximum values of the individual data items. Also, a report descriptor indicates the nature of the data reported

There are three types of reports: input, output, and feature. An input report is sent by a device and contains data intended for applications ran on the host computer. An output report is sent by the host computer to a device and contains application data intended for controls or displays. A feature report contains data intended for a device or data representing the status of a device. Unlike the data in the input or output reports, the data in a feature report are intended for use by the device configuration tools and not by the applications.

Some devices may have multiple report structures. To differentiate these structures, a report identifier is used, which is a one-byte value preceding each report.

6 HID-Class Specific USB Requests

HID-class devices must implement the *Get_Descriptor* and *Set_Descriptor* standard USB requests. In addition to these requests, HID-class devices may implement some requests that are specific to this class. These requests initiate transactions that allow the host computer to determine the capabilities and state of a device and to set the state of output and feature articles. The main HID-class specific requests are *Get_Report* and *Set_Report*. Implementation of the *Get_Report* request is mandatory. Table 1 The *Get_Report* and *Set_Report* requests specific to HID-class devices presents the Get_Report and Set_Report requests.

Table 1 The *Get_Report* and *Set_Report* requests specific to HID-class devices

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1010 0001	GET_REPORT (0x01)	Report type and Report ID	Interface	Report Length	Report
0010 0001	SET_REPORT (0x09)	Report type and Report ID	Interface	Report Length	Report

The *Get_Report* request allows the host computer to receive a report from a device via the default control pipe. The *wValue* field must contain the report type in the high byte and the report identifier (report ID) in the low byte. The report type specifies an input report (0x01), an output report (0x02), or a feature report (0x03). If report identifiers are not used, the low byte of the *wValue* field must be set to 0.

The *Get_Report* request is useful at device initialization time for determining the state of its features. The request is not intended for polling the device state at regular intervals. For repeated input reports, an interrupt In pipe should be used. Optionally, for output reports an interrupt Out pipe may be used (Not used for Optical Switch Device).

The *Set_Report* report allows the host computer to send a report to a device for setting the state of some controls. The meaning of the request fields is the same as for the *Get_Report* request.

7 Interfacing with HID-Class Devices

A HID-class device communicates with the driver of this class either via the default control pipe or via an interrupt pipe. The device uses the default control pipe for the following operations:

- Receiving USB requests sent by the host computer and sending the answer to these requests;
- Sending data when the device is polled by the HID-class driver via the *Get_Report* request;
- Receiving data from the host computer.

The HID-class driver uses an interrupt pipe for the following operations:

- Receiving data asynchronously from a device (data that have not been requested explicitly);
- Sending data to a device with low latency.

Using an interrupt Out pipe is optional. If a device initializes an interrupt endpoint to the output direction, then the output reports are sent by the host computer to the device via this pipe. If an interrupt endpoint with the output direction is not available, then the output reports are sent by the host computer via the control endpoint, using *Set_Report* requests.

8 HID Report Descriptor Used for Optical Switch FOD5508

```
const uint8_t Hid_ReportDescriptor[HID_SIZ_REPORT_DESC] ={

0x06, 0xFF, 0x00,       /* USAGE_PAGE (Vendor Page: 0xFF00)          */
0x09, 0x01,             /* USAGE (Optical Switch)                    */
0xa1, 0x01,             /* COLLECTION (Application)                  */

/* Optical Channel */

0x85, 0x01,             /*      REPORT_ID (1)                        */
0x09, 0x01,             /*      USAGE (Optical Channel)              */
0x15, 0x00,             /*      LOGICAL_MINIMUM (0)                  */
0x26, 0xff, 0x00,       /*      LOGICAL_MAXIMUM (255)                */
0x75, 0x08,             /*      REPORT_SIZE (8)                      */
0x95, 0x01,             /*      REPORT_COUNT (1)                     */
0x91, 0x82,             /*      OUTPUT (Data,Var,Abs,Vol)            */

0x85, 0x01,             /*      REPORT_ID (1)                        */
0x09, 0x01,             /*      USAGE (Optical Channel)              */
0x81, 0x82,             /*      INPUT (Data,Var,Abs,Vol)             */

/* Number of Optical Channels */

0x85, 0x02,             /*      REPORT_ID (2)                        */
0x09, 0x02,             /*      USAGE (Number of Optical Channels)   */
0x15, 0x00,             /*      LOGICAL_MINIMUM (0)                  */
0x26, 0xff, 0x00,       /*      LOGICAL_MAXIMUM (255)                */
0x75, 0x08,             /*      REPORT_SIZE (8)                      */
0x95, 0x01,             /*      REPORT_COUNT (1)                     */
0x81, 0x82,             /*      INPUT (Data,Var,Abs,Vol)             */

/* String Descriptors */

0x85, 0x03,             /*      REPORT_ID (3)                        */
0x09, 0x03,             /*      USAGE (String Descriptors)           */
0x15, 0x00,             /*      LOGICAL_MINIMUM (0)                  */
0x26, 0xff, 0x00,       /*      LOGICAL_MAXIMUM (255)                */
0x75, 0x08,             /*      REPORT_SIZE (8)                      */
0x95, 0x01,             /*      REPORT_COUNT (1)                     */
0x91, 0x82,             /*      OUTPUT (Data,Var,Abs,Vol)            */

0x85, 0x03,             /*      REPORT_ID (3)                        */
0x09, 0x01,             /*      USAGE (Optical Channel)              */
0x81, 0x82,             /*      INPUT (Data,Var,Abs,Vol)             */

/* String Descriptors Control */

0x85, 0x04,             /*      REPORT_ID (4)                        */
0x09, 0x04,             /*      USAGE (String Descriptors Control)   */
0x15, 0x00,             /*      LOGICAL_MINIMUM (0)                  */
0x26, 0xff, 0x00,       /*      LOGICAL_MAXIMUM (255)                */
0x75, 0x08,             /*      REPORT_SIZE (8)                      */
0x95, 0x01,             /*      REPORT_COUNT (1)                     */
```

```
0x91, 0x82,             /*      OUTPUT (Data,Var,Abs,Vol)       */
//0xb1, 0x82,   /*      FEATURE (Data,Var,Abs,Vol)              */

0x85, 0x04,             /*      REPORT_ID (4)                   */
0x09, 0x01,             /*      USAGE (Optical Channel)         */
0x81, 0x82,             /*      INPUT (Data,Var,Abs,Vol)        */


/* Device Control */

0x85, 0x05,             /*      REPORT_ID (5)                   */
0x09, 0x05,             /*      USAGE (Device Control)          */
0x15, 0x00,             /*      LOGICAL_MINIMUM (0)             */
0x26, 0xff, 0x00,       /*      LOGICAL_MAXIMUM (255)           */
0x75, 0x08,             /*      REPORT_SIZE (8)                 */
0x95, 0x01,             /*      REPORT_COUNT (1)                */
0x91, 0x82,             /*      OUTPUT (Data,Var,Abs,Vol)       */


0xc0                    /* END_COLLECTION                       */
};
```

9 Communication

The communication between Optical Switch and Host is implemented via HID reports. Table 2 Optical Switch HID reports. presents Optical Switch HID reports.

Table 2 Optical Switch HID reports.

Report ID	Report Type	Report Size	Report Description
1	Input	1 byte	Optical Switch Control: Returns selected Optical Channel. Channels are from 0 to N-1. To read this report the control pipe (*GET_REPORT*) and Interrupt pipe can be used. This report will be send to the Host asynchronously via Interrupt Pipe.
1	Output	1 byte	Optical Switch Control: Set Optical Channel. Channels are from 0 to N-1. To write Optical Channel use Control Pipe (*SET_REPORT*). To ensure that data is written poll the device by reading this report on control pipe while it will not be busy (while read data will not be equal busy flag 0xFF).
2	Input	1 byte	Number of Optical Channels: Returns maximum supported Optical Channels. To read this report use control pipe (*GET_REPORT*).
3	Input	1 byte	String Descriptor. To read this report use control pipe (*GET_REPORT*).
3	Output	1 byte	String Descriptor. To write this report use Control Pipe (*SET_REPORT*).
4	Input	1 byte	String Descriptor Control. To read this report use control pipe (*GET_REPORT*).
4	Output	1 byte	String Descriptor Control. To write this report use Control Pipe (*SET_REPORT*). To ensure that data is written poll the

			device by reading this report on control pipe while it will not be busy (while read data will not be equal busy flag 0xFF).
5	Output	1 byte	Device Control. To write this report use Control Pipe (*SET_REPORT*).

9.1 Optical Switch Control

Optical Channel can be changed by Host command or by user pushing device buttons. Any change Optical channel change will be reported by *Optical Switch Control Report* (ReportID=0x01) on Interrupt pipe. Also this report can be read using Control Pipe (GET_REPORT).

To set Optical channel send optical channel number using Control Pipe (SET_Report ReportID=0x01). To ensure that channel is changed poll the device by reading *Optical Switch Control Report* (ReportID=0x01) on interrupt pipe while data will be received or on control pipe while it will not be busy (while read data will not be equal busy flag 0xff).

Number of Optical Channels can be read *Number of Optical Channels Report* (ReportID=0x02) on control pipe.

9.2 String Descriptors

The device has three String Descriptors:

1. Product Name String Descriptor. Maximal string length is 16 8bits characters plus ending zero. String pointer is **0x01**;
2. Serial Number String Descriptor. Maximal string length is 16 8bits characters plus ending zero. String pointer is **0x02**;
3. Firmware Version String Descriptor. Maximal string length is 16 8bits characters plus ending zero. String pointer is **0x03**.

To read or write string descriptor first string descriptor pointer must be set. To set string descriptor pointer write *String Descriptor Control Report (ReportID=0x04)* to Control Pipe. To ensure that string pointer is changed poll the device by reading *String Descriptor Control Report (ReportID=0x04)* on control pipe while it will not be busy (while read data will not be equal busy flag 0xFF).

When the string pointer is changed string characters can be read by using *String Descriptor Report (ReportID=0x03)* on control pipe while ending zero will not be read.

When the string pointer is changed string characters and ending zero can be written by using *String Descriptor Report (ReportID=0x03)* on control pipe. After that write **Key0=0xFD** and **Key1=0xFE** to the device by writing *String Descriptor Control Report (ReportID=0x04)* on control pipe to save string descriptor to non-volatile memory. After each written *String Descriptor Control Report (ReportID=0x04)* polling must be used to ensure that command is completed. After writing Key1 polling must be issued after some delay (about 100ms), because writing to non-volatile memory stalls the device.

9.3 Device control

Some device control commands can be issued by writing *Device Control Report (ReportID=0x05)* on control pipe. After writing to this report polling is not used.

Available commands are:

1. Device Power Off. Report data must be **0xa0**;
2. Device Software Reset. Report data must be **0xa1**;
3. Enter Device Firmware Upgrade (DFU) mode. Report data must be **0xa2**;
4. Lock keyboard (hardware keys). Removing USB cable unlocks keyboard. Report data must be **0xa3**;
5. Unlock keyboard (hardware keys). Removing USB cable unlocks keyboard. Report data must be **0xa4**.

10 Revision History

Table 3 Revision History

Version/ Revision	Date	Notes
V2/Rev0	10 July 2015	Commands to lock/unlock keyboard added. (Chapter 9.3)