Stettbacher Signal Processing
Margrit Rainer Strasse 12a
CH-8050 Zürich

Phone: +41 43 299 57 23
Fax: +41 43 299 57 25
E-Mail: cam@stettbacher.ch

STETTBACHER SIGNAL PROCESSING

O-3000 Camera Series

Low-Level Programming Interface

Version 1.20
2014-07-17

Abstract: This document describes the configuration of the USB endpoints used for the communication with O-3000 cameras.

Contents

1 Introduction

The preferred way to communicate with the O-3000 series camera is to use the camera driver provided by Stettbacher Signal Processing. This driver is based on the Open-Source libusbx project and hides the complexity of the USB protocol inside a multiplatform driver.

For embedded projects however, where libusbx is not desired/feasible, it is mandatory to interact directly with the camera. This document describes how to communicate with the camera, i.e. how to send commands and receive status or error/warning messages. Additionally, it is explained, how to receive and interpret video data.

Key is to understand the concept of USB endpoints and the different transfer types.

2 A Crash Course in USB Terminology

The USB 2.0 specification (see [1]) describes the USB standard in an exhaustive way. Interfacing to the O-3000 camera series does not require to fully understand the USB 2.0 standard, but some basic knowledge is essential. In this chapter, some of the terms are detailled.

Connecting an USB device, like the O-3000 camera, to an USB host, typically a PC, will establish a communication channel between both. This communication channel provides a set of independent sub-channels, called pipes. The host-side end of the pipe are buffers, the device-side end consists of so-called endpoints. Each endpoint is uniquely addressed by the USB device's address and the endpoint number. Endpoint 0 is reserved for device control; the other endpoints (up to 15 for high-speed devices) are implemented application specific. An endpoint offers an IN channel (device to host) and an OUT channel (host to device), although not both of them have to be implemented. Endpoints support one of following transfers:

- control transfers
- isochronous transfers
- interrupt transfers
- bulk transfers

The specific transfer characteristics are described in detail in [1]. For the O-3000 camera programming, it is sufficient to know the bulk transfer.

3 USB Endpoint Configuration

The O-3000 series camera utilises a number of USB endpoints as described in the table below. Endpoints support bidirectional communication, although not every endpoint has to use both directions. Directions are always referenced from the host's point of view. Only endpoints 0 to 2 are used.

Endpoint	Direction	Transfer mode	Description
0	IN	Control	Reserved
	OUT	Control	Reserved
1	IN	Bulk	Video data sent from camera to host
	OUT	Bulk	Messages sent from host to camera
2	IN	Bulk	Messages sent from camera to host
	OUT	-	not used

Table 1: USB endpoint configuration.

4 Message Channel

4.1 Message Format

Messages are formatted in a XML-like syntax, detailled in [3].

4.2 Outbound Channel (Host to Device)

The purpose of the outbound channel is to send commands or configuration data to the camera. Configuration data is sent in form of parameters to the command *set*. To retrieve status information or configuration data, the command *get* can be sent. Communication over USB is regarded robust enough not to require handshaking. Therefore, the device will not send any acknowledge messages. In case any invalid configuration data is sent, the device tries a guess. E.g. if the frame width is not a multiple of 32 as required by the specification, the camera will round and apply the value accordingly. Dependent on the command/parameter, this auto-correction takes place silently, or a warning message is sent. In either case, to verify which values has been applied, read back the value with the complimentary *get* command.

4.3 Inbound Channel (Device to Host)

The inbound channel allows the device to send messages to the host. Such messages are either responses to the command *get*, or warnings or errors. Latter can be related to a *get* command, or not.

4.4 Message Timing

The inbound and outbound message channel are not synchronised. This means, sending a *get* message to retrieve a value will eventually result in a response being sent back, but the delay between these actions is not defined. To properly treat this situation, it is recommended to implement the receiving part in an asynchronous way, e.g. in a separate thread.

5 Video Channel

5.1 Continuous (Streaming) Mode

In streaming mode, the camera will continuously send image frames. These image frames are split into USB transfers of equal lenght. As the video frame size is dependent on the resolution, the start of the video frame (SOF) is usually at an arbitrary position within the USB transfer. To facilitate synchronisation, each video frame is prefixed with an image header, consisting of a preamble and information about the frame dimensions, etc. The format of the header is described in detail in [2]. Once a preamble is detected within an USB transfer, it is feasible to extract the frame size and predict the position of the next SOF. As the USB data rate is rather high, it is absolutely critical to handle the incoming transfers efficiently. Otherwise, the camera will lose synchronisation and start to drop frames. Although the camera will eventually recover, the situation is to be avoided.

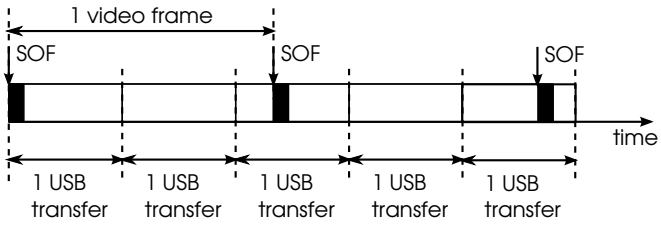

Figure 1: Streaming mode timing diagram.

5.2 Single Frame (Snapshot) Mode

Contrary to streaming mode, in snapshot mode only a single frame is sent by the camera, either triggered by the snapshot command, or by an hardware event. In this mode, the start of frame coincides with the start of the first USB transfer. All but the last USB transfers will have equal lenght. The last USB transfer will contain the exact amount of data needed to complete the frame. The next snapshot frame will start with another USB transfer.

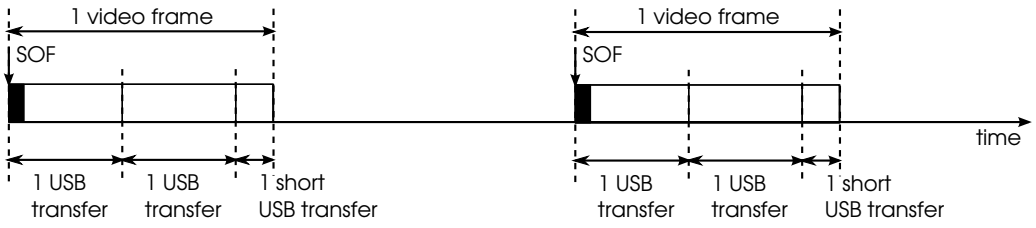

Figure 2: Snapshot mode timing diagram.

5.3 Image Data Format

The image data format is described in [2].

6 Literature and References

[1] USB Working Group (2000). "Universal Serial Bus Specification", Compaq Computer Corporation et al., version 2.0, 2000-04-27.

[2] Stettbacher Signal Processing (2013). “O-3000 Camera Series Image Frame Format Description”, Stettbacher Signal Processing, version 1.0, 2013-02-12

[3] Stettbacher Signal Processing (2013). “O-3000 Camera Series XML Command User Specification”, Stettbacher Signal Processing, version 1.0, 2013-02-12