INTRODUCTION

Below is a list of the main commands and comments used in the DrvKit 2.1 software to communicate between the computer and the TAG Optics Driving Kit 2.1. This information is provided on an AS IS basis by TAG Optics.

This software was designed to use Python 32bit 2.7.3. External libraries required are wxPython 2.9.4.0, PyUSB 1.0.0a2, numpy 1.6.2, and matplotlib 1.2.0.

CODES AND COMMENTS

tag.py TagControl class:

Multiple commands are used to communicate with the DrvKit2.1. The main commands that actively control the operation of the device are as follow: “SendInfo”, “SendCommand”, “SendCommandValue”, “SendCommandPulseValue” return an integer value of the command you ask (see below for further details).

- Connect(): Open a USB connection to the device and clear out the buffer. Sets correct configuration values for communication with the device.
- Close(): Closes the connection and destroys the USB device object in case things don't go as planned. This prevents any further communication.
- SetDefault(): Tries to return the device to the settings which is goes to when you first connect the device, regardless of if the program is running.
- SendInfo(cmd): cmd should be one of the value under #Second byte for an information update. This asks the device to return the current value for the command.
- SendCommand(cmd): cmd should be one of the values that don't include SET or PULSE under #Second byte for a command message.
- SendCommandValue(cmd, val): cmd should be one of the values that include SET under #Second byte for a command message. Val should be the integer value to set the command to.
- SendCommandPulseValue(cmd, val): cmd should be one of the values that include PULSE under #Second byte for a command message. Val should be the integer value to set the command to.

Non-public methods:
You shouldn't need to call any of these commands to interact with the device; they are provided for informational purpose only.

- int_to_4_bits(val): Used for SendCommandValue to convert the given integer to a 4 bit value.

TAG Optics, Inc. P.O. Box 1572 Princeton NJ 08542
Tel. 609-356-2142 www.tagoptics.com info@tagoptics.com

- int_to_2_bits(val): Used for SendCommandPulseValue to convert the given integer to a 2 bit value.
- writeusb(msg): Sends a string to the device. Parameter 'msg' should be a hex bytestring of the values to send.
- readusb(): Read from the device what is currently in the buffer. Sometimes this doesn't get the whole message, so it may need to be called multiple times until it does.
- CreatePacket(data): Takes the given input list of values from 0-255, and converts it to a bytestring the USB device will accept. Also deals with swapping values and creating the checksum.
- ProcessPacket(): This function is really messy. It will continue to read the device until the string returned is valid. It loops reading from the USB until it finds a non-blank string, and then checks it to be valid. If it is not, it keeps reading and replacing certain values until it is valid, or returns 0 if everything fails.

main.py GUI: This overall is really a quick and dirty way to get things done. It is not pretty at all, and lacks comments. A lot of last minute changes went into this, so some functions are never called or are redundant. Two main functions are the frequency scan and lock. The scan function takes a range of frequencies, and checks the phase difference 10 times at an interval of 1/100th of the range, and then loops over the range 10 times. It should result in a 5-10% drop in phase difference in a 1kHz section around a resonance. The lock function reads the phase difference every third of a second about 100-150 times, and either tries to correct the frequency by 3Hz if it's not lower in phase count than the previous value.

TAG Optics, Inc. P.O. Box 1572 Princeton NJ 08542
Tel. 609-356-2142 www.tagoptics.com info@tagoptics.com