Software Design Specification

for the

Colorado 2400u Device Driver

Prepared by Richard Murri

11/2/05

Contents

1) Introduction

1.1 - Purpose

The purpose of this document is to define how the Colorado 2400u driver will be structured. It can then be used to improve, understand, or redesign the code.

1.2 - Changes

This document complies with the Requirements Specification that has already been created. There are no changes in scope.

1.3 - Overall Design & Document Format

Since the SANE API defines how the driver will work, much of the design decisions are restricted. There are, however, a few items that are not straightforward, and are specific to the Colorado 2400u driver, which will be mentioned in this document.

The program will be written in the C programming language. This is a requirement of the SANE API. The document will require using some static arrays for performance and design reasons.

The layout of the document is broken down into two sections. The first section reviews the SANE API and will describe how this specific driver will fit to that specification. The second section defines how the driver will be implemented in those areas that are specific to this driver alone.

1.4 - References

SANE API, Version 1 - http://www.sane-project.org/html/doc009.html, 8/15/05.

USB specification - http://www.usb.org/developers/docs/usb_20_02212005.zip, 8/17/05.

2) SANE API

2.1 - Code Flow

Figure 1 shows the code flow in the SANE API. Each function described will need to be implemented by the driver.

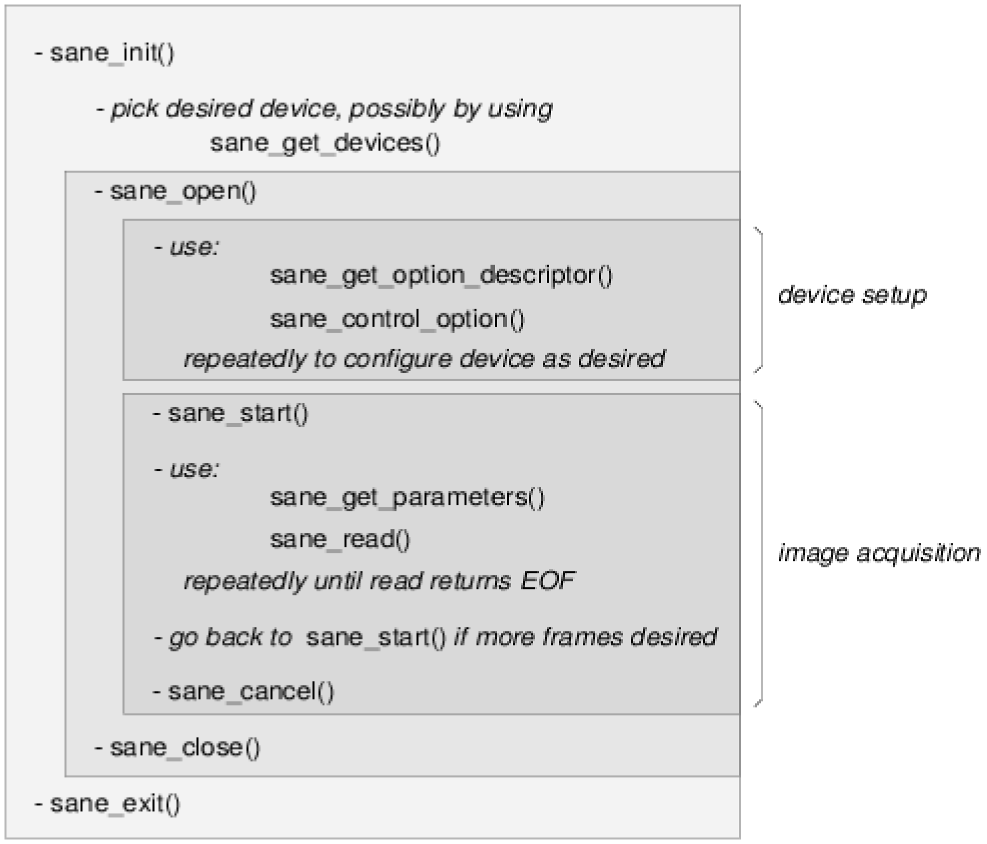
- sane_init()
- pick desired device, possibly by using sane_get_devices()
- sane_open()
- use: sane_get_option_descriptor() sane_control_option() repeatedly to configure device as desired
device setup
- sane_start()
- use: sane_get_parameters() sane_read() repeatedly until read returns EOF
- go back to sane_start() if more frames desired
- sane_cancel()
image acquisition
- sane_close()
- sane_exit()

Figure 1 – Code Flow (http://www.sane-project.org/html/doc009.html)

2.2 - Functions

2.2.1 - sane_init()

This function will initialize the back-end. Any variables that need to be initialized are done in this function.

2.2.2 - sane_exit()

This function will terminate the back-end.

2.2.3 - sane_get_devices()

This function will return a SANE_Device pointer of the Colorado 2400u scanner if the scanner is plugged in and has the correct permissions. This function does not necessarily need to be called before the open function is called.

2.2.4 - sane_open()

This function will establish a connection with the Colorado 2400u scanner. A SANE device handle will be returned for the newly opened scanner driver. This part of the driver will run through setting up the USB configuration setting, the USB interface, and the USB alternate interface.

If there are any errors with opening the device (such as incorrect permissions) a SANE error will be returned.

2.2.5 - sane_close()

This function will close the scanner after being opened.

2.2.6 - sane_get_option_descriptor()

Since there will be only one changeable option for scans, we must send the front end the appropriate option that it can use. This function will pass back the SANE structures that are needed so that the front end knows how to change the dpi.

2.2.7 - sane_control_option()

According to the SANE specification, this function will set the values to control the scanner. A value of 200 for the dpi will set the scanner in black and white mode with text scanning optimized. A value of 100 for the dpi will set the scanner to color scanning mode. The scanner afterward is required to scan at that resolution.

2.2.8 - sane_get_parameters()

If the options are set at 200 dpi we will scan in black and white mode. The scan will create a picture of 1656 x 2338 pixels. We will set the SANE parameters into gray mode and the depth will be one.

If the options are set at 100 dpi, we will scan in color mode. The pixels will be interleaved as red, green, blue, and the depth will be 16 (or one byte per pixel). The size setting will be 826 x 1169 pixels.

2.2.9 - sane_start()

This function starts the image acquisition. The scanner initialization data will be read in during this function and sent to the scanner.

2.2.10 - sane_read()

This function returns a number of bytes read from the scanner. This amount is specified by the function parameters. The 'Colorado 2400u Breakdown' section later on in this paper provides further details of how the read will be performed.

2.2.11 - sane_cancel()

When this function is called, the scan will cancel and the device will prepare for another scan.

2.2.12 - sane_set_io_mode()

This function will not be implemented for this device driver.

2.2.13 - sane_get_select_fd()

This function will not be implemented for this device driver.

3) Colorado 2400u Breakdown

3.1 - Functions

During a read, lots of data will need to be passed to the scanner and back. This data can be divided into several different categories.

3.1.1 - Control Transfers

The USB 2.0 specification defines a control transfer. This transfer includes five different fields: RequestType, Request, Value, Index, and Length. These fields are 1, 1, 2, 2, and 2 bytes long respectively. The values of these and the number of control transfers are all specified by the scanner.

3.1.2 - Bulk Transfers

A bulk transfer will send or receive a large amount of data from the scanner. Since transfers have different varieties, they will be separated into different functions based on what they are meant to do.

- Bulk reads. These reads only require a handle to the object that will be read from. They will then read the data and, if necessary, write it to the front end as image data.
- Bulk writes of the value '0'. For the configuration, it is necessary to write large blocks of 0's to the scanner. This function will need as a parameters where to write the 0s and how many to write.
- Bulk writes of calibration data. A calibration is necessary in the Colorado 2400u. A specific calibration function will be provided to send that data to the scanner.

3.1.3 - Repeated Control Transfers

The Colorado 2400u scanner will send a single control transfer many times while it is waiting for the scanner motor to move into position. The repeated transfer waits until a certain value is found before it stops repeating the transfer.

3.2 - Integration with SANE

The scan will take place throughout several different SANE functions. The communication between the scanner and the driver has been split into five different sections so that the scan can be broken up between the functions.

3.2.1 - Initialization

This communication will take place during the function sane_start(). Its main purpose is to set up the basic operations of the driver. This is similar to the same way the data flows in the Windows driver.

3.2.2 - Scanner Setup

Each scan mode (Color or Text) needs certain data passed to the scanner. This portion of the communication will also take place in sane_start().

3.2.3 - Calibration

After the scanner is setup, we need to send calibration data to the scanner and make sure it is in order. This calibration communication will also take place in the function sane_start().

3.2.4 - Scan

The actual scan will take place in sane_read(). This data must be done iteratively, but SANE only allows a certain amount of data to be received at once. sane_read() will need to maintain status of what data has been read and what data has not before proceeding on with the next read.

3.2.5 - Finalization

After all of the data has been read and passed to the front-end, proceed and finish off any remaining communication with the scanner.

3.3 - Control Representation

Now that the functions are defined to communicate with the scanner, how will the communication data be represented? Since the communication is very large and needs to be accessed in a systematic way, it will be represented as a static array of integers. The array will be two-dimensional with each row representing one instruction to communication with the scanner. The functions that need data will be able to cycle through the data and read from it as they need.

Data will be formatted as follows:

3.3.1 - Control Transfers:

(the default communication method)

Element 1 will be the RequestType

Element 2 will be the Request

Element 4 concatenated with 3 will be the Value

Element 6 concatenated with 5 will be the Index

Element 8 concatenated with 7 will be the Length

Any transferred data will be placed in subsequent elements starting with the first byte to be transferred.

3.3.2 - Repeated Control Transfers

Element 1 will be 0xfb showing that it will be a repeated control transfer.

Element 2 will be the RequestType

Element 3 will be the Request

Element 5 concatenated with 4 will be the Value

Element 7 concatenated with 6 will be the Index

Element 9 concatenated with 8 will be the Length

The number to be watched will be place in Element 10 (As soon as the watched element is seen the function will return)

3.3.3 - Bulk Read

Element 1 will be 0xfa, showing that it will be a bulk read

Element 2 will the endpoint that the data will be read from

Element 4 concatenated with 3 will be the size of the read

3.3.4 - Bulk Writes of the value '0'

Element 1 will be 0xff, showing that it will be a bulk write of 0s

Element 2 will the endpoint that the data will be written to

Element 4 concatenated with 3 will be the size of the write

3.3.5 - Bulk writes of calibration data

Element 1 will be 0xfd or 0xfc, showing that it will be one of two types of calibration data

Element 2 will the endpoint that the data will be written to

Element 4 concatenated with 3 will be the size of the write

The data to transmit will be determined by the driver.

3.4 - Error Checking

Each step of a scan will be checked to ensure that it performs correctly. If there has been an error, either in communicating with the hardware or in any other case, the software will notify the SANE front-end of the error and discontinue to scan.