# Visioneer XP100 Sheetfed Scanner Linux Drivers

Manual, API, and Development Notes

J.B. KOPENA[1], R.A. PRIMERANO
*Secure Wireless Agent Testbed Project*
Department of Computer Science,
Department of Electrical and Computer Engineering
College of Engineering, Drexel University
`http://swat.coe.drexel.edu/`

## 1 Introduction

This document outlines code and programs developed for using a Visioneer XP100 sheetfed scanner under Linux. The following section describes installation of the code and usage of included utility programs. This is followed in Section 3 by notes on development of this code, including the settings and protocols which have been discovered or determined for this particular scanner.

## 2 User's Manual

This section outlines installation and usage of the driver code.

### 2.1 Installation

The XP100 scanner provides only for USB access. Interaction with the scanner in this software through the USB interface is accomplished via the LibUSB library. LibUSB provides for user-space access to USB devices connected to the host. It is available from the LibUSB project's homepage at [2]. The library must be installed, as root, from either a source distribution or binary package before compiling the scanner software. Note that most common, up to date Linux distributions include a version of LibUSB which may be sufficient. This software has been tested and is known to work with LibUSB version 0.1.8, the latest stable version as of this writing.

The latest archive for the XP100 Linux driver code may be obtained from the same source as this document. This software does not have to be installed

[1] Contact author: `tjkopena@cs.drexel.edu`

as root, nor the scanner application run as root. In particular, it does not have to be run setiuid root.

After unpackaging the archive, execute the Makefile. This creates two files: `settings` and `scanhack`. The latter is a simple program for taking a scan. The `settings` file contains the register/value pairs which must be sent to the scanner before conducting a scan. By default, during execution the`scanhack` program looks for this file in the current working directory. However, another directory may be given as a command line parameter as discussed in the following section.

### 2.2 Execution

The `scanhack` program provides a simple utility for taking a scan with the XP100 as well as serving as a small example of using the API outlined in this document. In essence, the program simply starts a scan and creates an image file with the result. It accepts the following optional command line arguments:

- ○ `-nw` — Do not wait for paper to be inserted and the button on the XP100 to be pressed before conducting a scan. If this option is not specified, the program will pause until paper is inserted and the button pressed before scanning.
- ○ `-s` *dir* — Load the `settings` file from directory *dir*. The default is the location of the `scanhack` executable. Note: default is not the current working directory.
- ○ `-o` *file* — Write the output image to *file*. This defaults to `test.ppm` in the current working directory.
- ○ `-w` *float* — Width to scan, in inches. Default is 8.5”.
- ○ `-h` *float* — Height to scan, in inches. Default is 11”.
- ○ `-c` — Enable color scanning.
- ○ `-r[0|1|2|3|4|5|6|7]` — The optical resolution at which to conduct the scan. Defaults to 0. Values are as follows:

| **Parameter** | **DPI Resolution** |
|---|---|
| `-r0` | 600 |
| `-r1` | 400 |
| `-r2` | 300 |
| `-r3` | 200 |
| `-r4` | 150 |
| `-r5` | 100 |
| `-r6` | 75 |
| `-r7` | 50 |

- `-d[0|1|2|3]` — The color depth at which to conduct the scan. Defaults to 3. Values are as follows:

| Mode | Parameter | Bits per Pixel |
|---|---|---|
| | `-d1` | 2 |
| | `-d2` | 4 |
| | `-d3` | 8 |
| | `-d1` | 6 |
| | `-d2` | 12 |
| | `-d3` | 24 |

For example, the following command places into the file `out.ppm` a 3.5x2.5” grayscale scan at 150dpi without a pause before scanning:

```
./scanhack -r4 -nw -w 3.5 -h 2.5 -o out.ppm
```

While running, the `scanhack` program will display a small progress meter to indicate what percent of the scan has been completed. Note that the scanner may physically pause during a scan until data can be transferred from the on-board buffer memory to the host computer. This may happen frequently at higher resolutions and larger scans. In addition, the scan may not be complete even when the paper has physically passed through the scanner and there may still be data to transfer off the buffer. After the specified area has been scanned, the program ejects the remainder of the paper.

### 2.3 Output

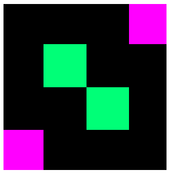
Image files are currently output in plain Portable PixMap (PPM) format. The exact specifications are available at [4], but output from this code adheres to the following slightly simplified format. All numbers in this, the plain format variant of the PPM specification, are given in human-readable ASCII decimal notation. Each item is deliminated by whitespace.

- A two character magic number: “P3”
- The width, $w$, in integer pixels
- The height, $h$, in integer pixels
- The maximum integer color value, based on the specified color depth: $2^b$ where $b$ is bits per pixel
- $w * h$ triplets of integer color values for the red, blue, and green components of each pixel, separated by whitespace

A very simple, example plain PPM file adapted from [4] is the following:

```
P3

4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```



As all numbers in this format are encoded in ASCII decimal and whitespace deliminated, these files tend to grow rather large. However, they are easy to parse or transform using common utility programs. ImageMagick's [1] `convert`, included in most Linux distributions, can be used quite simply:

```
convert test.ppm test.jpg
```

## 3 Development Notes

The Strobe XP100 scanner is marketed for use on the Windows 98/2000/ME/XP operating systems. Visioneer does not supply Linux drivers for this product. This section outlines the methodology used to reverse engineer the scanner and produce custom Linux driver software, along with notes on the results.

### 3.1 Scanner Control IC

The Strobe XP100 is controlled by a single integrated circuit, the LM9833 USB Image Scanner IC, developed by National Semiconductor. This device is a generic scanner controller which requires extensive initialization to work properly with the Strobe XP100 scanning hardware. This hardware includes a linear color image sensor, stepper motor, light source, and several interface switches. The LM9833 is designed to interoperate with a large variety of image sensing technologies, light sources, and stepper motors from a broad range of manufacturers. This is accomplished by setting ninety four registers corresponding to hardware parameters such as sensor resolution, motor steps per inch, and illumination mode. The reverse engineering process involved determining the proper values to write to each of these registers so that the Strobe XP100 will function correctly.

### 3.2 Development Process

Visioneer does not publish documentation on the internals of the XP100, so much of the development process consisted of determining appropriate register values. This task was greatly simplified by the data sheets which National Semiconductor publishes for the LM9833 [3], which lists and describes the parameterization registers.

Raw valid values were obtained by installing the scanner and supplied drivers on a Windows computer and using a USB sniffer to log its interaction with the scanner. Such programs record all traffic on the USB bus to and from the scanner. With this program running, we attached the scanner and scanned a page. The captured data was then matched with the registers on the IC using the LM9833 data sheets.

### 3.3 Register Values

Table 1 provides the full list of LM9833 register values for the XP100. Constant values such as sensor resolution, illumination type, stepper motor properties have been copied directly from recorded USB logs, with the exception of minor modifications to the paper present/absent state polling and transition event behavior. Of the ninety four active registers in the LM9833, all but twelve fall into this category. These twelve are either control lines or are calculated based on the properties of the scan to perform and must be reset each time a scan is conducted. The remaining registers are not used. Registers 0x00–0x07 are used to operate the scanner. Registers 0x09, 0x24–0x26, and 0x46–0x47 are the calculated, scan-dependent values. Refer to the National Semiconductor LM9833 Data Sheet and Programmers Reference or the driver source code for these calculations.

### 3.4 Proceduces

Protocols, calculations, and other specifics needed in interacting with the XP100 may be found in the driver source code, specifically in `scannerlib.c`. Basic functionality demonstrated includes: detecting the presence of a XP100 and its USB address; claiming and configuring the XP100's USB interface; confirming the LM9833 chipset; jumping the page (to provide initial traction); ejecting the page; setting gamma, offset, and gain tables for image correction and manipulation; waiting for paper to be loaded and using the scanner's button interface; and scanning. The following are some important high level points.

- **Detection.** The XP100's Vendor and Product codes are 0x04a7 and 0x0427, respectively. It may be found on the USB bus simply by enumerating all devices and searching for these ID codes.

- **Access.** The XP100's read/write endpoints are 0x02/0x03, respectively.

| Register | Label | Value | Register | Label | Value | Register | Label | Value |
|---|---|---|---|---|---|---|---|---|
| 00 | Image Buffer | | 20 | Line End—MSB | 2b | 40 | Fixed Multiplier—MSB | 40 |
| 01 | Image Data Available | | 26 | Color Mode | | 41 | Fixed Multiplier—LSB | 00 |
| 02 | I/O Flags | | 21 | Line End—LSB | F2 | 42 | Pixel Gain and Offset | 26 |
| 03 | DataPort Mode | | 22 | Data Pixel Start—MSB | 00 | 43 | n (Line Skipping) | 00 |
| 04 | DataPort Address—MSB | | 23 | Data Pixel Start—LSB | 0b | 44 | m (Line Skipping) | 00 |
| 05 | DataPort Address—LSB | | 24 | Data Pixel End—MSB | | 45 | Motor Settings | 13 |
| 06 | DataPort | | 25 | Data Pixel End—LSB | | 46 | Scan Step Size—MSB | |
| 07 | Command Register | | 27 | Color Mode | 00 | 47 | Scan Step Size—LSB | |
| 08 | Master Clock Divider | 0a | 28 | Reserved | 00 | 48 | Fast Feed Step—MSB | 02 |
| 09 | H-Res and Data Mode | | 29 | Illumination Mode | 00 | 49 | Fast Feed Step—LSB | 58 |
| 0a | Turbo and Preview | 00 | 2a | Lamp Control | 0b | 4a | Steps to Skip—MSB | 00 |
| 0b | Sensor Configuration | 0d | 2b | Lamp Control | B8 | 4b | Steps to Skip—LSB | 00 |
| 0c | Sensor Control | 00 | 2c | Lamp Control | 01 | 4c | Step counter—MSB | 00 |
| 0d | Sensor Control | 25 | 2d | Lamp Control | F4 | 4d | Step counter—LSB | 00 |
| 0e | Sensor Control | 00 | 2e | Lamp Control | 03 | 4e | Pause Scanning | 90 |
| 0f | Sensor Control | 18 | 2f | Lamp Control | E8 | 4f | Resume Scanning | 01 |
| 10 | Sensor Control | 01 | 30 | Lamp Control | 01 | 50 | Steps to Reverse | 00 |
| 11 | Sensor Control | 00 | 31 | Lamp Control | F4 | 51 | Acceleration Profile | 00 |
| 12 | Sensor Control | 04 | 32 | Lamp Control | 03 | 52 | Phase difference—MID | 0b |
| 13 | Sensor Control | 00 | 33 | Lamp Control | E8 | 53 | Phase difference—LSB | D0 |
| 14 | Sensor Control | 00 | 34 | Lamp Control | 01 | 54 | Lines to Process | 00 |
| 15 | Sensor Control | 00 | 35 | Lamp Control | F4 | 55 | Misc Motor Control | 09 |
| 16 | Sensor Control | 00 | 36 | Lamp Control | 03 | 56 | PWM Frequency | 02 |
| 17 | Sensor Control | 03 | 37 | Lamp Control | E8 | 57 | PWM Duty Cycle | 16 |
| 18 | Sensor Control | 07 | 38 | Static Offset | 00 | 58 | Paper Sense Settings | 01 |
| 19 | Integration Time | 00 | 39 | Static Offset | 00 | 59 | MISC I/O 1 & 2 | EE |
| 1a | Stepper Phase | 00 | 3a | Static Offset | 00 | 5a | MISC I/O 3 & 4 | 16 |
| 1b | Stepper Phase | 01 | 3b | Static Gain | 01 | 5b | MISC I/O 5 & 6 | 09 |
| 1c | Black Pixel Start | 01 | 3c | Static Gain | 01 | 5c | ADC Test | 00 |
| 1d | Black Pixel End | 02 | 3d | Static Gain | 01 | 5d | ADC Test | 00 |
| 1e | Active Pixel Start—MSB | 00 | 3e | Fixed Offset—MSB | 00 | 5e | ADC Test | 00 |
| 1f | Active Pixel Start—LSB | 0b | 3f | Fixed Offset—LSB | 00 | | | |

*Table 1* Register settings for the XP100. Registers and values are in hexadecimal.

- **Soft reset.** Writing the scanner configuration to the LM9833's registers requires that the scanner be in soft reset mode. The IC initializes to this mode upon being plugged into the host computer, or may be sent there by writing 0x20 to register 0x07. It may be returned to the normal, idle, mode afterward by writing 0x00 to register 0x07. Only the following registers may be written outside of that mode:

  | **Registers** | **Functionality** |
  |---|---|
  | 0x03–0x06 | The Dataport. |
  | 0x2a–0x27 | Lamp controls. |
  | 0x38–0x3d | Static gain and offset. |
  | 0x42 | Offset and gain source, bits 0–2. |
  | 0x45 | Stepper motor status. |
  | 0x58–0x5b | Paper sense and misc I/O settings. |

  The contents of the registers are not lost upon soft reset. They may be set once and then ignored during the session, except those which must be updated based on the scan parameters.

- **Memory.** The XP100 has a 256k by 16bit DRAM, used for buffering scan lines before transmission as well as storing the gamma, offset, and gain tables. The contents of the DRAM are lost upon soft reset.

- **Scan.** An important note on the scan process is that the gamma, offset, and gain tables must be rewritten for each scan as they are lost when the LM9833 is put into soft reset mode to write the scan parameters to the relevant registers. A scan therefore follows this general procedure:

  1. Calculate the settings based on the scan parameters.
  2. Enter soft reset mode, write the relevant registers, return to idle.
  3. Write the red, green, and blue gamma, offset, and gain tables. Only the blue tables are used in black and white scans.
  4. Jump the paper slightly to gain initial traction.
  5. Start the scan.

     (a) Collect scan lines in block reads from the output register, 0x00. The scanner automatically pauses and resumes if the buffer becomes too full (this is configurable, e.g. based on the DRAM size and scan width). It may be configured to either stop scanning upon reaching the end of the paper or to and continue (sending blank lines). Note that two status bytes are appended to the end of every scan line and are not part of the image data.

(b) Continue fetching scan lines until the correct number of lines has been read. This number is calculated based on the requested DPI and image height. The control IC itself does not know the height of the image and will simply continue to scan.

6. Return the scanner to idle mode.

- **Resolution.** The XP100 has an optical resolution of 600DPI.

Specifics and calculations may be found in the driver source code.

### References

[1] ImageMagick. `http://www.imagemagick.org/`. avail. July 2004.

[2] LibUSB project home. `http://libusb.sourceforge.net/`. avail. July 2004.

[3] LM9833 product folder. `http://www.national.com/pf/LM/LM9833.html`. avail. July 2004.

[4] PPM format specification. `http://netpbm.sourceforge.net/doc/ppm.html`. avail. July 2004.