Term Paper

Volumetric Display based on Inkjet-Technology

Author: Erwin Roth,
erwin.roth@weihenstephan.org

Technische Universität München (TUM)

Institute of Micro Technology and Medical Device Technology (MIMED)

Advisors: Dr.-Ing. Franz Irlinger
Prof. Dr. rer. nat. Tim C. Lüth
em. Prof. Dr.-Ing. Dr.-Ing. E.h. J. Heinzl

Date: 08th May 2006

Document Revision History

Version History	Date	Description of Change
Rev 0	12th April 06	Initial public release
Rev 1	08th May 06	Fig. 37, Table 11 corrected

Abstract

A volumetric display device is a graphical display device capable of producing volume-filling imagery, as opposed to planar images generated by traditional display techniques.
In comparison to stereoscopic displays which show the spatial information from the perspective of only one viewer, volumetric displays allow multiple viewers to look at the generated image from different perspectives at the same time.

Applications of volumetric displays can be found where complex 3D-data should be visualized, including medical imaging, telepresence and scientific visualization.

The aim of this term paper is to conduct basic research, whether the technology of drop creation by modern inkjet printheads can be used as an alternative approach for generating volumetric imagery and overcome certain drawbacks of existing approaches, like limited spatial resolution, dead display zones and the inability to display object occlusion.

This document is also concerned with the research for finding a suitable printhead and the required control signals therefore, to produce volumetric images.

Table of Contents

Abbreviations / Glossary

ASIC	Application Specific Integrated Circuit
DW	refers to a dataword, see 6.7.3.1 for details
DWID	refers to a dataword identification number, see 6.7.3.2 for details
DTB	datablock, see 6.7.3.3 for details
DPI	dots per inch, specifies a resolution
nozzle array	representing all nozzles of a printhead belonging to a certain color
nozzle group	subset of nozzles within a nozzle array containing nozzle sets, see 6.7.4 for details
nozzle set	subset of nozzles within a nozzle array, see 6.7.4 for details
PCB	printed circuit board
TTL	Transistor Transistor Logic
voxel	smallest addressable display entity within a volume. Relates to a volume, as a pixel to an area

Table Index

Image Index

1. Introduction

1.1 Initial Thoughts

The man-machine interface – especially the display part – is vital to many successful tasks in the field of engineering and medical science.
Advancements in computer technology allow the computation and analysis of increasingly complex objects and processes, e.g. 3D-computer aided design data, numerical finite element simulation or computer tomography data.
The requirement to present the data to users in a meaningful way for further interpretation often stands at the end of such calculations. Spatial three-dimensional models are in many cases the tool of choice, due to their rich level of information.

However, when it comes to displaying these spatial models on traditional (flat) CRT or LCD computer screens, the user is limited to seeing the 3D-model from one specific perspective at any given time.
Stereoscopic[38] displays relax this limitation, by creating the illusion of spatial depth, yet are limited to produce this effect only for a single viewer and limited viewing angle.

In order to produce actual volume-filling dynamic imagery, which can be seen simultaneously by multiple viewers from multiple perspectives, so called volumetric displays are the first-choice - allowing a team of surgeons for example to plan the safest approach for removing a brain tumor by looking at a volumetric image of the CT brain scan from several angles at the same time.

A number of methodologies are known for generating volumetric imagery. This paper suggests a new technical approach for producing volume-filling images by using inkjet printhead technology to create visible pixels called voxels within the volumetric display space overcoming certain depth cues and resolution limits of existing approaches.

1.2 Objectives

Work objectives are as follows:

- Analysis whether inkjet- printhead technology is basically capable of producing volumetric imagery.
- Selection of suitable, readily available printers/printheads to conduct image generation experiments.
- Analysis if standard printhead control electronics can be used for volume-filling imagery.
- Reverse engineering of printhead control signals.
- Development of own printhead control electronics/software as far as required.
- Execution of basic volumetric image generation experiments for proof of concept.

2. Volumetric-Displays

2.1 Definition

Volumetric displays belong to the group of autostereoscopic displays. These displays present a 3D-image to a viewer without requiring special viewing aids like glasses.

Three classes of autostereoscopic displays are known [48]:
- re-imaging displays
- parallax displays
- volumetric displays

This document will concentrate on volumetric displays.

Blundell and Schwarz defined "[that a] volumetric display device permits the generation, absorption, or scattering of visible radiation from a set of localized and specified regions within a physical volume." [37]

In simple words it could be said, "volumetric displays create imagery that appears to float in a volume. Typically the imagery can be seen from a wide variety of angles. Furthermore, goggles are generally not required to perceive 3-D imagery." [40]

2.2 Basic Principles of Volume Imagery

Aim of all volumetric displays is to create the impression of a floating object within a spatial display volume in the eyes of one or multiple observers.

There have been numerous research activities on this field in the past and recent years already, to find and implement technologies which can fulfill this goal.

The pursued techniques can be divided mainly into the following two categories:

- Swept volume displays
- Static volume displays

A comprehensive, but not complete overview of prior research can be found in Fig.1.

The merit of volumetric displays is primarily measured based on their ability to provide depth cues (motion parallax, occlusion, lighting, stereopsis and convergence) [47] and high resolutions. The volumetric-display approaches differ greatly in the way they can fulfill these attributes.

2.3 State of the Art

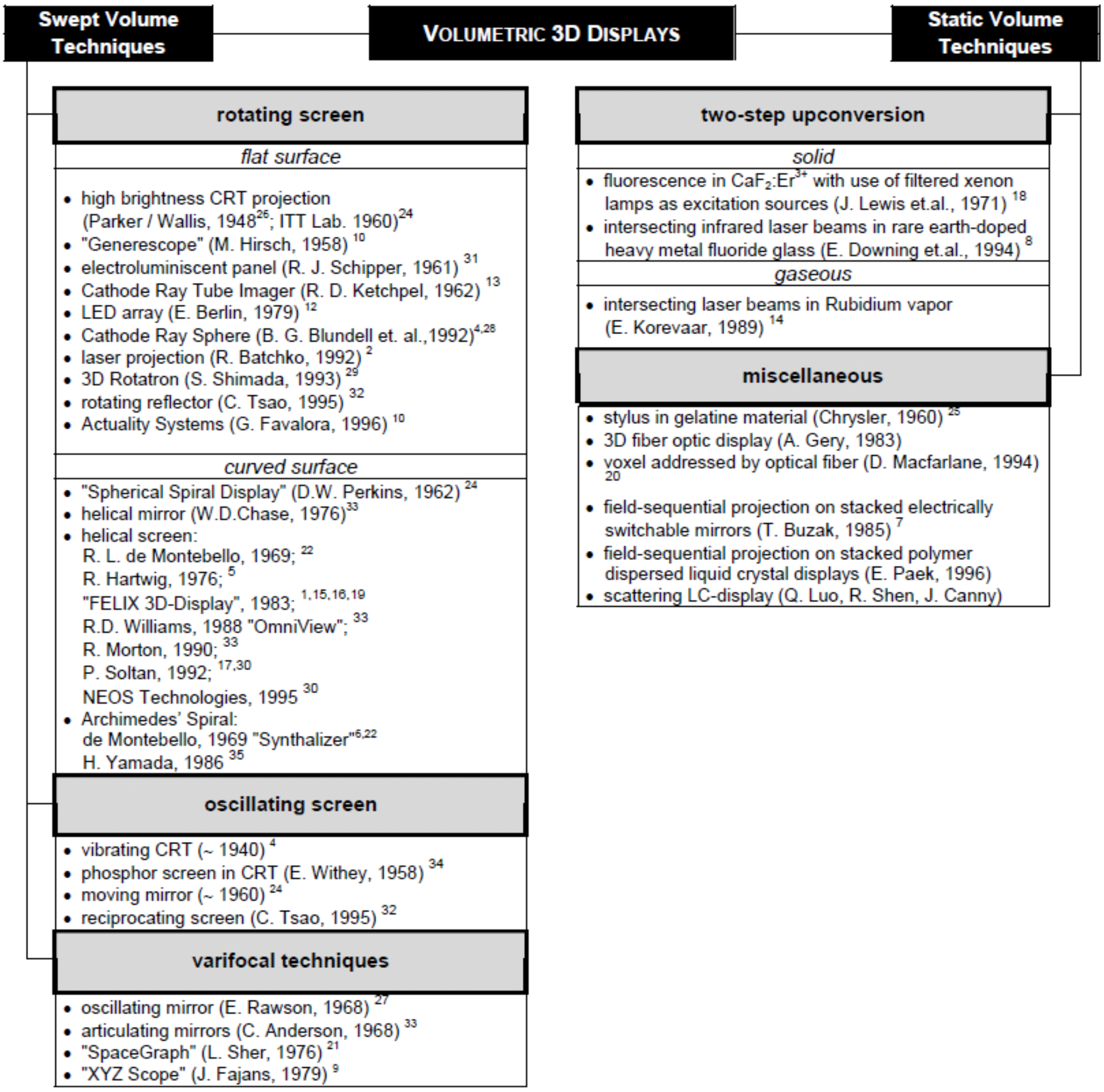


Fig.1 Overview of volumetric imaging technologies
Source: [36]

2.3.1 Swept Volume Displays

The following paragraphs give an explanation of the basic technique behind swept volume displays, based on existing implementations. At a later point in this document this explanation will be expanded to cover also inkjet-based volumetric displays.

"In the case of swept volume displays, the display volume is created by the mechanical motion, either vibrational or rotational, of a target screen. A periodically time-varying two-dimensional (2D) image is used to sweep out the volume cyclically at a frequency higher than the eye can resolve. Eventually a spatial image is formed through persistence of vision. Thus several of the usual cues of everyday depth perception are exhibited, especially the compelling ones of stereoopsis and parallax." [36]

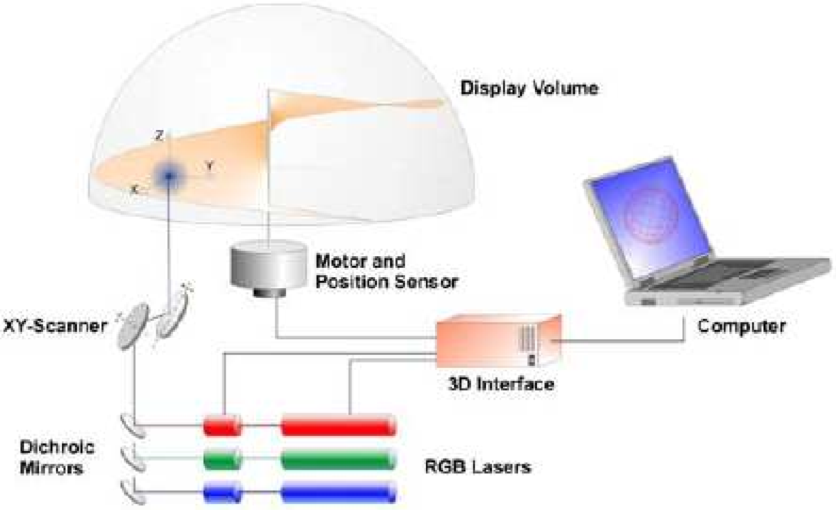


Fig. 2 Laser projection on a helical screen
Source: [36]

"The primary 2D pattern may be generated on either an emissive panel or a passive projection screen. The depth effect is achieved when the screen surface, which can take various shapes, is oscillated perpendicularly to itself, or when it is rotated, in synchronism with the cyclic 2D pattern. The screen serves as a plane with which a volume is scanned in order to provide a three-dimensional blackboard or raster upon which various spatial patterns may be written."[36]

Size and display accuracy of these displays is limited by mechanical factors:

- centrifugal forces caused by the rotating projection plane restricting the size of the projection plane
- dead display zones at locations near rotational axis, where a image projection is not possible

Moreover, visual depth cues like overlapping (occlusion) of two objects can't be provided, due to the transparent character of the projection screen.

However, most of the market available volumetric displays today are swept volume displays due to their moderate requirements for image generation hard- and software.

Example of a swept volume display device:

Fig. 3 - Actuality Systems Inc.'s Perspecta® Display
[39]

2.3.2 Static Volume Displays

Systems requiring no mechanical motion within the display volume are classified as static volume displays. "The goal of this technique is to provide emissive voxels at a large number of locations in a static setup incorporating no oscillating or rotating parts. Several interesting attempts have been made using transparent crystals, gases, electronic field sequential techniques, and others. However, current technology limits make the near term implementation of large scale displays difficult.

A common principle used to create 3D pictures in a static setup is the two-frequency, two-step (TFTS) upconversion technique. "It is based on the absorption of two photons (normally invisible IR photons) by optically active atoms, molecules or ions. The light sources used to excite the active material are monochrome and of different wavelengths. A first photon with wavelength λ1 excites the ion or molecule to an intermediate energy level E1 with lifetime t2, a photon of the second light source (wavelength λ2) then excites the dopant to the higher level E2. The decay back to the ground state goes along with the emission of a visible photon – a point of light is created within a doped display volume."[45]

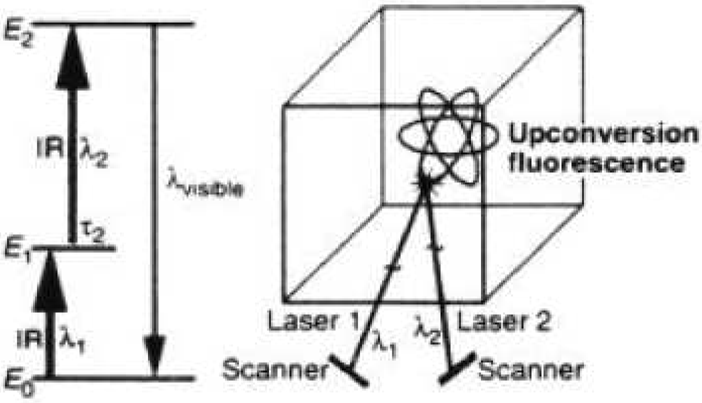


Fig. 4 Two-frequency, two-step (TFTS) upconversion methode
Source: [8]

Another more recent approach in the field of static volume displays is to generate voxels in ambient air by creating points of local plasma using short high energy laser pulses [46].

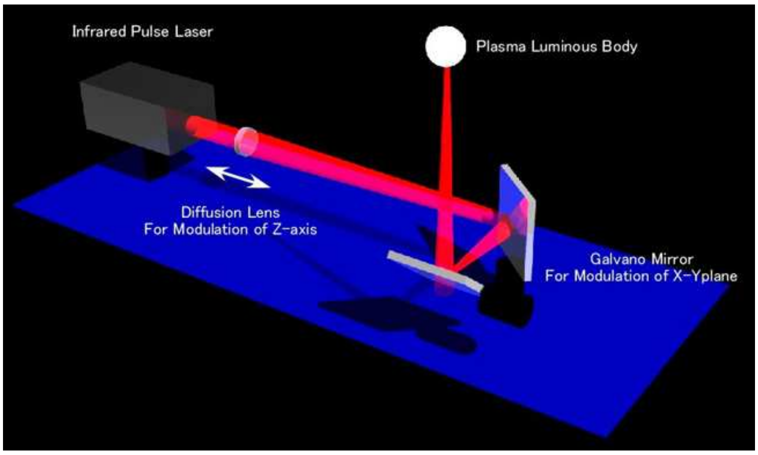


Fig. 5 Midair plasma generation with pulsed laser beam.
Source: [46]

Problem of both techniques is the limited resolution achievable with a single voxel generation source, due to mainly mechanical constraints in beam positioning for voxel excitation. Especially in case more than one voxel generation source is used, excitation of undesired voxels, so called ghost-voxels in the crossing point of two beams is a further problem.

The static-volume approach isn't able to present object occlusion and view-independent shading of objects, since active voxels emit omnidirectional and uniform wavefronts and inactive voxels are transparent.

3. Volumetric Display based on Inkjet-Technology

3.1 Basic Principle

Basic principle of a volumetric display based on inkjet technology is to generate a controlled rain of drops falling through the volumetric display area and the illumination of the falling drops by short light flashes at specific times. The reflected and scattered light at the surface of a single drop represents an active voxel.

The process of drop generation and illumination is repeated at a frequency, where the human visual system perceives an active voxel flicker-free at a position fixed in space.

Color images could be created by using alternating light flashes of different colors.

Fig. 6 Artistic drawing: volumetric-image with drops as voxels (zoomed view of 2D-pattern)

3.2 Principle Hardware Design

Fig. 7 sketches the principle hardware design of an inkjet based volumetric display.

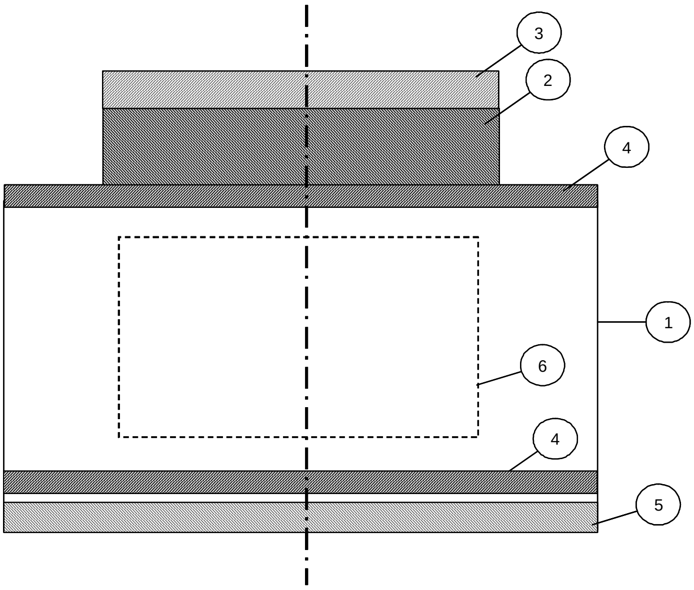


Fig. 7 Principle hardware design inkjet based volumetric display. Front view of a symmetric system.

A glass vessel (1) encloses the actual display volume (6). Atop the glass vessel sits the drop generation unit (2). This unit consists of inkjet printheads arranged in a matrix. Each printhead in turn contains a number of nozzles arranged in matrix form.

The drop generation unit nozzle matrix (2) has to cover with its horizontal dimensions the footprint of the actual display volume (6).

The drop generation unit gets its fluid out of the fluid reservoir (3) on top of it.

Circular arranged flash light sources (4) mounted above and below the actual display volume (6) are synchronized by a control unit with the drop ejection of the drop generation unit (2) and illuminate the sequence of falling drops in specific time intervals. The light reflected and scattered by the falling drops is sensed by the observer’s eyes.

Located at the bottom of the glass vessel is a mechanism to catch the falling drops (5), with the intention to minimize drop reflection when they hit the vessel ground.

Main criteria for the drop generation unit:

- high drop ejection frequency
- constant drop ejection velocity
- minor angular deviation of ejection trajectory
- high number of simultaneously ejectable drops
- high nozzle density per square inch

In order to minimize air resistance of falling drops, air pressure within the vessel, drop generation unit and ink reservoir shall be lower than atmospheric pressure level. Since the lowered air pressure also has influence on the boiling point of the used ink liquid and as further consequence on its ejection velocity, the optimal pressure ratio in correspondence to vertical resolution and vertical display size needs to be calculated.

For a monochromatic display the color of the ink used for drop generation should be the same as the color of the flash light sources. For a color display, the drop color should be white in order to reflect different wave lengths at a similar level.

The amount of ink stored in the ink fluid reservoir (3) has impact on the non-stop usability time of the volumetric display. The ink ejection volume of the display is dependent on drop size/volume, ejection frequency and number of drops ejected simultaneously, whereas the later two are directly dependent on the content to be shown.

In order to keep the ink reservoir small, a nonvolatile fluid should be used as ink and the mechanism to catch falling drops (5) at the bottom of the vessel should recover the liquid and feed it back to the ink reservoir.

The inkjet based volumetric display approach is a member of the class of swept volume displays since the ink drops representing the display media traverse through the volumetric display space.

3.3 3D-Image Generation Process

The following schematic gives an overview about the major steps required to transform a computer 3D-object model into a physical 3D-object represented by voxels within the volumetric display device.

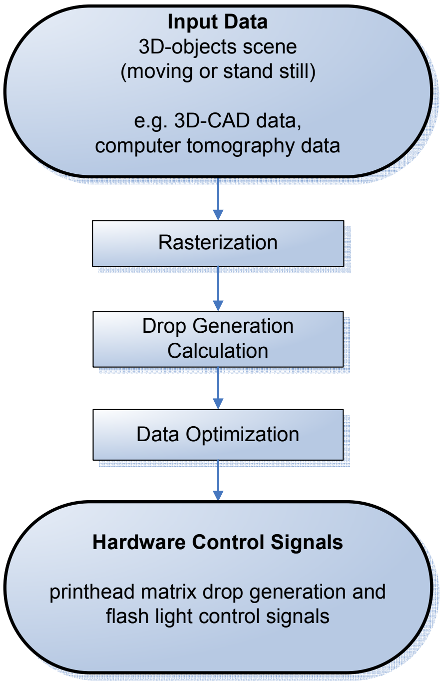


Fig. 8 Schematic 3D-image data processing diagram

Input Data
3D computer models which contain moving and/or fixed objects can be used as input data for further processing. The input data can be vector based, e.g. 3D-CAD data or rasterized, e.g. computer tomography data.

Rasterization
Each drop ejected by the printhead matrix corresponds to a single voxel. In order to display whole 3D-objects as congregation of voxels the input data needs to be rasterized corresponding to the horizontal and vertical resolution constraints of the volumetric display device.

The input data is split into horizontal layers from top to bottom corresponding to the vertical resolution of the volumetric display. Vertical resolution depends mainly on the printhead’s max. drop ejection frequency, drop size, drop ejection velocity, air resistance, vertical display height and display refresh frequency.

The max. horizontal resolution is determined by the min. distance between two nozzles in the printhead matrix.

Drop Generation Calculation
This step transforms the raster data into a sequence of drop ejection commands and calculates their timing. Attention has to be paid to the circumstance that the drops corresponding to the bottom layer need to be ejected first, since these drops have to cover the largest distance. Therefore the layer scanning direction for drop generation is bottom-up.

Data Optimization
Drop generation commands can be optimized for better image quality by considering physical effects, like e.g. differing air resistance in a sequence of direct successive drops.

Hardware Control Signals
Drop ejection commands are translated into their corresponding electrical control signals driving the printhead matrix. The electrical signals for the printhead matrix and the monochrome or color flash lights need to be synchronized.

3.4 Technical Advantages of Inkjet based Approach

Compared to other technological approaches for realizing volumetric displays, the inkjet based approach delivers the following advantages:

- High horizontal resolution achievable, due to high nozzle density of modern inkjet printheads and their ability to eject a large number of drops simultaneously.
- Scalable to a large degree in horizontal direction without loosing resolution, due to matrix (stepwise) arrangement of printheads.
- Possibility to realize object occlusion depth cue, due to the non-transparency property of small drops caused by light refraction combined with the possibility to eject drops not only acting as visible voxels but also for blocking light propagation, e.g. by positioning them within the edges of a solid body.
- Absence of moving mechanical parts, prevents blind spots within the display volume and allows better scalability due to missing mechanical constraints like centrifugal forces.

3.5 Technical Obstacles of Inkjet based Approach

Besides the advantages there are also a number of technical obstacles which need to be overcome in order to realize a practically usable volumetric display.

- Vertical resolution and display size is limited by drop ejection frequency, drop ejection velocity and air resistance.

Let’s look at some sample figures to make things more clear.
Formula to calculate min. required drop ejection frequency f_{eject}:

$$f_{eject} = \frac{vertical\_display\_size}{vertical\_resolution} * display\_refresh\_rate$$

Formula to calculate average vertical drop speed $\bar{v}_{drop}$ necessary to meet display refresh rate:

$$\bar{v}_{drop} = vertical\_display\_size * display\_refresh\_rate$$

Assumed values and results for a sample display:

Vertical display size:	25 cm
Vertical resolution:	200 dpi
Display refresh rate:	50 Hz

$$\boxed{f_{eject} =} \quad 98425 \text{ Hz}$$

$$\boxed{\bar{v}_{drop} =} \quad 12{,}5 \text{ m/s}$$

In comparison, a currently available consumer market thermo inkjet print head has a maximum drop ejection frequency of ca. 25 kHz and a drop ejection speed of ca. 8 m/s.

A standard LCD or CRT computer screen has a resolution of ca. 72 dpi. However the size of one computer screen pixel is considerable larger and the pixels lie side by side in contrast to the inkjet based volumetric display, where one voxel has only the diameter of a single drop, e.g. 15 um and the voxel to voxel distance within a vertical line has with the values from above a mark-space ratio of about 8.5.

In order to achieve the impression of a solid object within the display volume for an observer, a considerably lower mark-space ratio is necessary. Another possibility to achieve the visual impression of a lower mark-space ratio without actually changing it is to optimize drop positioning of surrounding vertical and horizontal layers in the same and consecutive refresh cycles.

- Drop deceleration and fog formation, due to air resistance

A major obstacle for scaling inkjet based volumetric displays in vertical direction is caused by the influence of air resistance and the resulting deceleration of the vertically moving drops.

Force due to air resistance (=drag):

$$F_r = \frac{1}{2}\rho_{gas} C_d A_{ref} v^2$$

ρ_{gas} = density of the gas within the volumetric display

C_d = drag coefficient

A_{ref} = reference area

v = drop velocity

ρ_{gas} can be influenced via the gas pressure:

$$\rho_{gas} = \frac{\hat{M}}{\hat{R}T} p$$

p = gas pressure

$\hat{M}$ = molar mass

$\hat{R}$ = ideal gas constant

T = gas temperature

This correlation allows to lower air resistance of the falling drops by lowering the gas pressure. By lowering the pressure to zero, air resistance would disappear, however this is not feasible, due to the connection of the ink's boiling point to surrounding gas pressure, which also affects drop ejection speed if thermal based inkjet printheads are used.

- Repeat accuracy of voxel positioning

The accurate positioning of a single voxel within the display volume is dependent on an accurately predictable drop trajectory. The trajectory depends on factors like constant drop ejection speed, angular deflection at the nozzle exit and since air friction is not negligible, also on the surrounding air flow conditions caused by preceding and/or neighboring drops.

Drop ejection speed and angular deflection should be as constant as possible. The impact of surrounding air flow conditions, main distance to and number of preceding drops needs to be calculated/estimated and drop ejection adjusted accordingly in order to achieve high repetition accuracy.

- "satellite droplets" causing ink fog and undesired light reflections

Satellite droplets are remains of ejected drops and can emerge during the drop ejection process. Satellite droplet formation can be hindered by nozzle design and optimization of driving conditions. Satellite droplets have negative impact on the picture quality, since they can cross the trajectory of adjacent drops and remain for a long time in the display volume, as they are much lighter compared to the actual drops representing a voxel.

3.6 Definition of Goals

- The goal is to create an experimental setup, which proves the usability of inkjet based drop generation for producing volume-filling imagery.
- Simple 3D-objects, like cubes, spheres or lines oriented in 3D-space shall be statically displayed within a display volume of about $1cm^3$.
- The generated images shall be visible for the human eye, without requirement of special tools for the observer.
- Results should be transferable to experiment setups with larger usable display sizes and drop generation units consisting of several printheads.

3.7 Definition of Experiment Environment

Experiments shall be executed under the following conditions:

- A single printhead equipped with a sufficient number of nozzles arranged in matrix style shall be used for proving 3D-image generation effect.
- Standard room atmosphere/air pressure and temperature.
- A single monochrome flash light source shall be used.

4. Selection of Suitable Printer/Printhead Models

Printheads need to fulfill the following criteria in order to be applicable to basic volumetric-imagery generation tests.

Priority (1=highest)	Criteria	Description
1	Physical matrix nozzle layout	The generation of a consecutive sequence of horizontal droplet layers, which form the visible voxels, requires that the nozzles of a single printhead are (evenly) distributed over the footprint area of the display volume. Thus the nozzles of the printhead shall be aligned in several rows, instead of a single row, as frequently found in low-cost consumer inkjet printers.
2	Parallelism of drop output	Predictable drop trajectory requires that angular deviation of ejected drops is kept to a minimum.
3	Nozzle density / physical resolution	Pitch between adjacent nozzles should be less than 0.085 mm (corresponds to 300DPI) in order to have a sufficient horizontal mark-space ratio (drop/no-drop).
4	Constant drop size, velocity and angular deviation	Necessary for a predictable drop trajectory.
5	Number of concurrently ejectable drops	High display resolution requires, that as many drops as possible can be ejected simultaneously.
6	Printhead drive frequency	Mainly related to drop size. A higher drive frequency enables a higher vertical display resolution.
7	Printer driver source code availability	Nice to have as information source for physical printhead properties.

Table 1 Printhead selection criteria

The range of potential printhead candidates reached from printers of the consumer product area over business products up to large format industrial inkjet printers.

However nearly all of the reviewed printers had only printheads consisting of a single nozzle row or two rows in a distance of ≥ 5mm.

Based on public available information related to selection priority 1 and 3, the following printhead / printhead-cartridge combinations were selected for further testing.

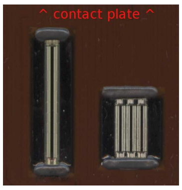
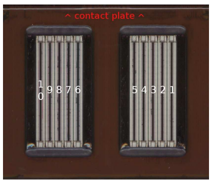
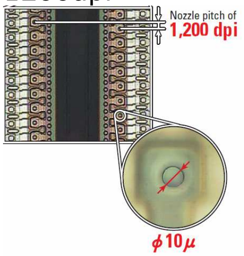
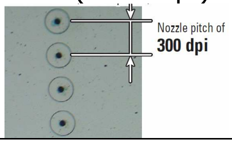
Model / **Property**	**Hewlett Packard 6578 ink-cartridge and printhead combination**	**Canon PIXMA ip2000 printhead**	**Canon PIXMA ip8500 printhead**
Technology	thermal	thermal (Full-photolitho-graphic Inkjet Noozle Engineering)	thermal (Full-photolitho-graphic Inkjet Nozzle Engineering)
Printhead type	disposable	permanent	permanent
Nozzle count& arrangement (vendor values)	72*6=432	1*320+3*256 =1088 	8*768=6144 
Physical resolution	600dpi	black: 600dpi non-black: 1200dpi	1200dpi 
Adjacent nozzle pitch	84um (=300dpi) 	black: 84um (=300dpi) non-black: 42um (=600dpi)	42um (=600dpi)
Drop volume	5 Pico liter	5-2 Pico liter	2 Pico liter
Number of electrical contacts	52	40	69
Information / image source	[43]	[42]	[41],[42],[43]

Table 2 Printhead comparison matrix

Due to the low price and widespread availability of the HP6578d printhead, it was decided to start with further investigations based on this model.

5. Hewlett Packard 6578 Printhead

The Hewlett Packard (HP) HP6578 is a three color printhead-cartridge combination, found mainly in the HP DeskJet 9xx printer series. Different variants (HP6578a, HP6578d) are known of this printhead-cartridge combination holding varying ink capacities. The printhead-cartridge combination is called disposable, since after the ink is empty it's disposed as a whole.

5.1 *Physical Nozzle Layout*

The nozzle plate holds 6 rows of 72 nozzles each. The rows are organized in groups of two, called nozzle array. One nozzle array for each color (cyan, magenta and yellow). Overall nozzle count is 432.

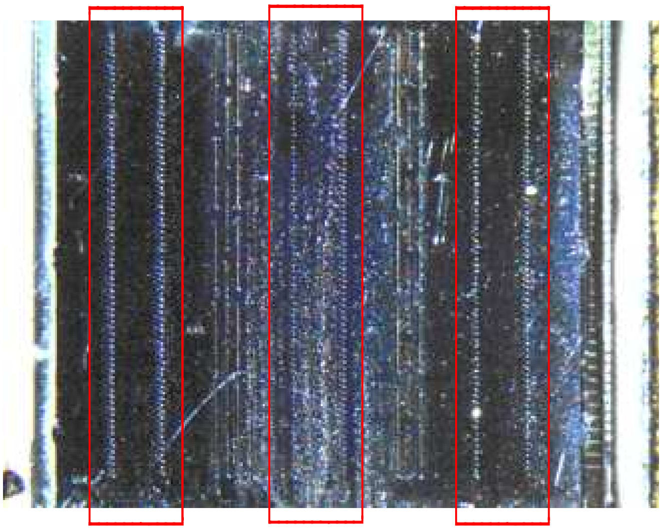


Fig. 9 Three nozzle arrays of HP6578

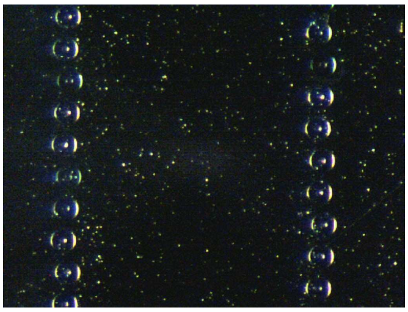


Fig. 10 HP6578 nozzle layout within one array

Nozzle pitch between two adjacent nozzles within one row is 84um leading to a physical resolution per row of 300dpi.

The cartridge has 52 electrical contacts.

5.2 *Nozzle Control Information Inquiry*

Building a volumetric display based on inkjet technology requires information about how to address and drive individual nozzles or a set of nozzles at the same time.

A research was conducted with the result that Hewlett Packard is not willing to give out technical information about how to control their cartridge. However a number of other information sources were found providing pieces of evidence about the required control signals:

- Hewlett Packard patent U.S. 5946012 [49], providing technical specifications and control signals of a similar printhead-cartridge combination.
- An article [50] in the magazine Elektor, describing the electrical circuitry required to drive an HP DeskJet 500C cartridge.
- Source code [51] of open source Inkjet Linux printer driver from Hewlett Packard.

The first analysis target was to find out, how to address a single nozzle and how many nozzles can be ejected at the same time. Table V of [49] indicates that nozzle addressing is done via a matrix addressing scheme with 22 groups of 14 nozzles each, allowing the addressing of maximum 308 nozzles. However, the HP6578 cartridge has 432 nozzles and the same number of electrical contacts. Leading to the conclusion that the HP6578 cartridge has some additional address selection circuits directly integrated into the printhead.

Based on this boundary conditions it is assumed that the HP6578 cartridge can eject at least 14 nozzles simultaneously.

A further information found in [49] indicates a maximum ejection frequency of 20kHz.
Table VI of [49] provides a pin assignment of electrical control signals. No tests were performed to actually check these information against the HP6578 cartridge.

Overall the analysis results indicate, that the HP6578 cartridge additionally fulfills the printhead selection priorities 4, 5 and 6 of Table 1.

In order to further investigate parallelism of drop ejection (selection criteria 2) and verify selection priorities 4, 5 and 6, actual drop ejection experiments shall be made.

A HP DeskJet 980Cxi printer was selected for execution of drop ejection experiments.

5.3 Drop Ejection Experiment Preparations

Since it’s not possible during normal print operation to directly look at the nozzle plate and observe the drop ejection process, it was necessary to modify the test device in order to fulfill this analysis requirement.

Due to the electrical signal requirements of the printhead (signals for 52 electrical contacts), it was decided, that it should be much easier using the standard printer control electronics to drive the printhead instead of creating own circuitry for generating the required electrical signals. This allows furthermore usage of the standard printer driver software at least for initial tests.

Therefore the printer was modified, so that the HP6578 cartridge could be separated from the cartridge carriage and mounted fixed on top of a glass vessel.

The separation caused a number of problems with the printer electronics, since the controller board expects certain signals of photoelectric barriers in order to finish printer initialization correctly and perform printing operation as usual.

The problems could be finally solved by combining two printers of similar kind and rerouting certain signals. The result can be seen in Fig. 11:

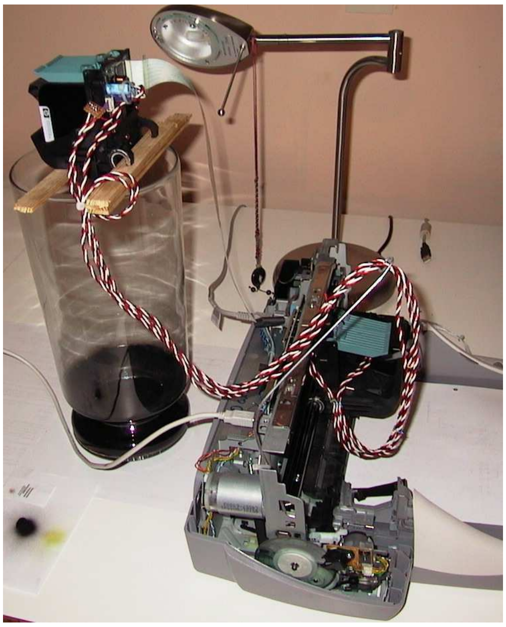


Fig. 11 HP6578 (Deskjet 980Cxi) drop ejection experiment setup

5.4 Printhead Test Software

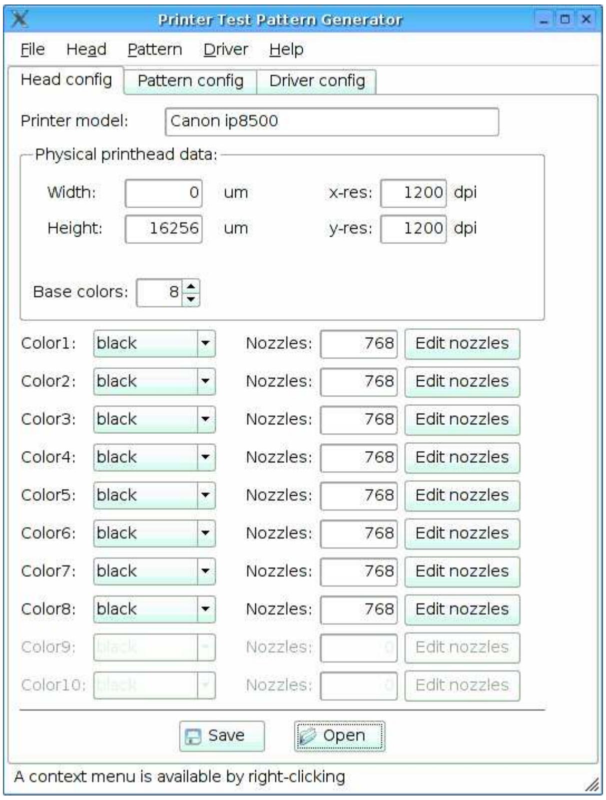
As specified above, the standard printer electronics are used for the drop ejection experiments described consecutively. The usage of the standard printer electronics has the advantage that the standard printer driver software can be used to trigger drop ejection.

Yet, forecasting which nozzles will be active in relation to a certain input pattern or in which sequence nozzles will become active is difficult, due to a number of software based output optimization techniques implemented in most printer drivers, affecting for instance whether the printer is ejecting drops while the carriage is going from right to left or unidirectional.

In order to get the best possible control about the nozzle selection process, an own printhead test software was written, since no suitable test software could be found on the Internet.

The printhead test pattern generator software takes as configuration values the physical parameters of the printhead, like number of nozzles per color, resolution, nozzle plate layout and printer/printhead quality mode and creates based on this settings and in combination with a definable nozzle activation pattern an uncompressed high resolution bitmap. This bitmap has the physical resolution of the printhead and is feed through the standard printer driver while most print optimization techniques should be disabled.

The printhead test pattern generator software is written in C++ with the platform independent toolkit Qt [52] version 4.1, allowing the usage under different operating systems.



Printer model: Canon ip8500

Physical printhead data:

Width: 0 um
x-res: 1200 dpi
Height: 16256 um
y-res: 1200 dpi
Base colors: 8

Color1: black Nozzles: 768
Color2: black Nozzles: 768
Color3: black Nozzles: 768
Color4: black Nozzles: 768
Color5: black Nozzles: 768
Color6: black Nozzles: 768
Color7: black Nozzles: 768
Color8: black Nozzles: 768
Color9: black Nozzles: 0
Color10: black Nozzles: 0

A context menu is available by right-clicking

Fig. 12 Screenshot: Printhead Test Pattern Generator software

5.5 Drop Ejection Experiments

Experiments with the printhead test software showed, that the optimization techniques embedded into the printer driver software and partly also printer controller hardware, make it impossible to use the standard printer driver and printer controller board as (software) interface for displaying volumetric objects.
For the purpose of displaying volumetric objects with the aid of inkjet printheads, a more direct control of the nozzle addressing and ejection process is required as the standard printer driver software interface allows.

A further test run in which as much nozzles as possible should be activated simultaneously made another more serious problem with the HP6578 cartridge obvious. If a large number of nozzles is active simultaneously it becomes evident, that not all nozzles eject their drops perpendicular to the surface of the nozzle plate, instead the trajectory of left- and rightmost ejected drops crosses at a distance of about 2.5 cm below the nozzle plate.

This effect could be observed with three different cartridges of the same series, leading to the assumption that this behavior is intended by design. However, this behavior is a major obstacle for using this type of printhead in further volumetric display tests. Therefore all further tests with HP6578 cartridges were stopped and Canon printheads were selected for further tests.

6. Canon FINE Printhead Family

The Canon FINE printhead family was chosen to conduct further analysis and experiments with. FINE stands for **F**ull-photolithographic **I**nkjet **N**ozzle **E**ngineering. Based on the public available information [43] all members of this printhead family fulfill the selection criteria as specified in section 4.

Compared to the HP6578 disposable printhead this class of printheads is from the technological point of view a big step ahead, due to its

- high amount of integrated control circuits,
- much higher nozzle count to area ratio,
- full photolithographic production process and
- smaller ejected drop sizes.

For example the current high-end model of this class, the ip8500 printhead is equipped with 6144 nozzles (8 colors) controlled via a 69 pin interface and ejects drops with a volume of 2 pico liters, compared to the HP6578 printhead with 432 nozzles controlled by 52 pins and an ejected drop volume of 5 Pico liters.

Initial reverse engineering experiments with this printhead family will be executed with an ip2000 printhead, since this model is cheaper and easier available on the (second hand) market compared to the high end variant ip8500.

Public available information and basic signal tapping tests with other Canon printers using FINE printhead models (Canon PIXMA ip4000) allow the strong assumption, that the basic findings won with the ip2000 printhead will be transferable to other (high-end) models within this class.

6.1 Nozzle Control Information Inquiry

Since Canon doesn’t provide any open source printer driver for the PIXMA series an E-Mail was sent twice to Canon Germany and the printing division of Canon Japan with the request to provide more detailed information about the capabilities of their FINE printheads and how to control specific nozzles in the context of the university project. No reply was received.

In parallel a patent inquiry was conducted, however the only related patents to FINE printheads found, e.g. U.S. 5478606, focus on the manufacturing process and don’t provide information about how to control the final printhead product.

Due to the lack of information, but good general qualification of the printhead based on the initial selection criteria, it was decided to start a reverse engineering process for the required control signals.

6.2 *ip2000 Printhead*

6.2.1 Physical Nozzle Layout

The printhead contains four nozzle arrays one for each color: black, cyan (=blue), magenta and yellow.
In the further proceeding focus is put on the non-black nozzles, as their arrangement can be conceived as nozzle-matrix with 3x2 rows. Such a matrix layout is a basic requirement to be applicable to volumetric image generation.

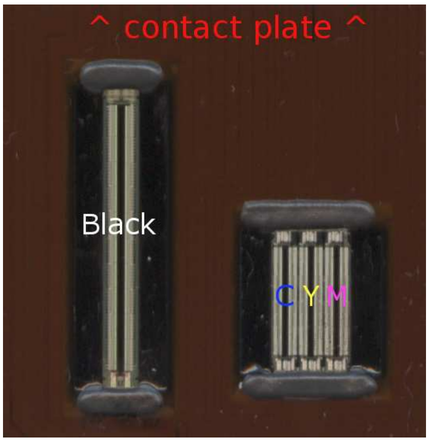


Fig. 13 ip2000 printhead nozzle plate layout

Values observed/measured with impinging light microscope.

6.2.1.1 Black Nozzle

Information for reference purposes only.

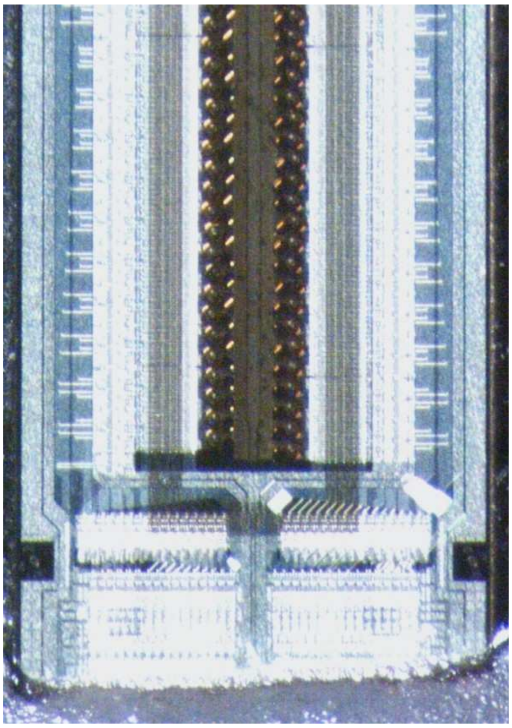


Fig. 14 ip2000 black nozzle, segment 1&2

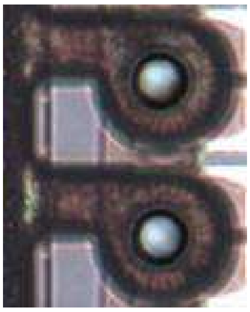


Fig. 15 ip2000 printhead black nozzle - zoomed

Property	Public Vendor Specifications [44]	Observed / measured Values*
Nozzle count	320	10segments*32nozzles +2segments*8nozzles= 336 nozzles
Inner nozzle diameter		24 um
Adjacent nozzle distance	84 um	90 um
Opposing nozzle distance		310 um

Table 3 ip2000 printhead black nozzle properties

*measured with Heidenhain MT12

6.2.1.2 Non-Black Color Nozzles

The printhead possesses three nozzle arrays, one for each color. From left to right: cyan, yellow and magenta.

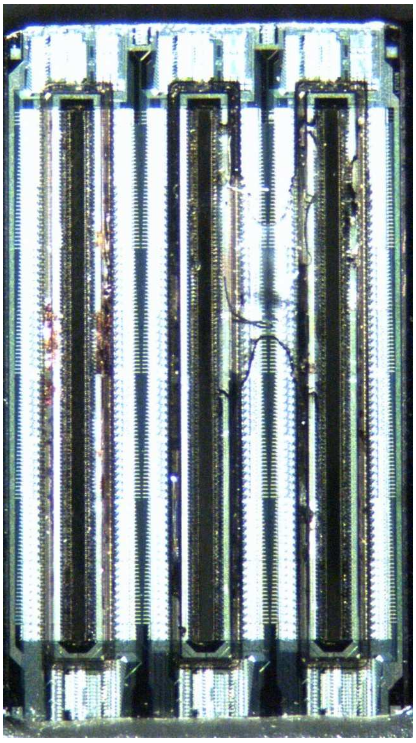


Fig. 16 ip2000 C – Y - M-nozzles, all segments

Fig. 17 ip2000 printhead non-black nozzles - zoomed

As can be seen in Fig. 17 the non-black color nozzle uses two different nozzle diameters. This configuration was only seen with the ip2000 printhead, not with ip4000 or ip8500 printhead.

Property	Public Vendor Specifications [44]	Observed / Measured Values*
Nozzle count identical for all 3 colors	256 per color	4 segments * 32 large nozzles +4 segments * 32 small nozzles +2 segments * 8 nozzles= 272 nozzles per color
Large nozzles inner diameter		16 um
Small nozzles inner diameter		10 um
Adjacent nozzle distance		41 um
Distance of horizontal opposing nozzles within one array		258 um
Distance of outmost vertical nozzles		5.72 mm

Table 4 ip2000 printhead non-black nozzles properties

*measured with Heidenhain MT12

6.2.2 Electronic Components

The following major electronic components can be found within the ip2000 printer case:

Power supply:

- Input 230V AC
- Output 0V, +24V DC
- Connected to printer controller board via 3 wires (0V, 2x24V)

Printer controller board:

- ASIC printer processor
- Connection with motors and photoelectric barrier sensors
- Connected to printhead carriage PCB via two 20-pin ribbon cables (connectors:J201, J202 on board)

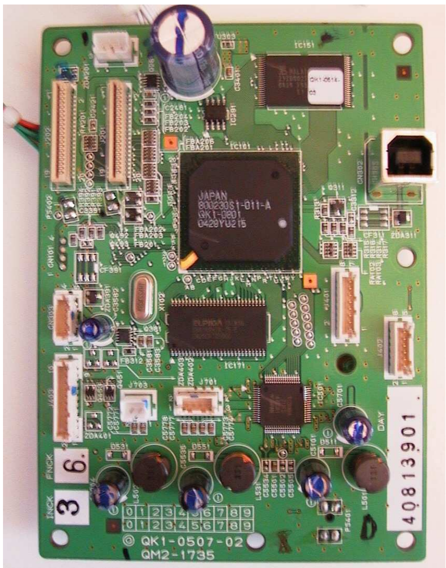


Fig. 18 ip2000 printer controller board

Printhead carriage PCB:

- Acts mainly as interface board for printhead, only few electronic components are found on this board.
- Holds optical sensor for incremental horizontal distance measuring.
- Connected to printhead via 40-pin connector.

Printhead:

- Nozzle control electronics directly integrated into nozzle plate.

In the process of reverse engineering the necessary control signals for selective nozzle control, focus is put on the interface between printer controller board and printhead carriage PCB, as this connection seems to have

- Direct influence on nozzle operation.
- Suitable for wire tapping, as the cable ending on the printer controller board side is a non-moving part during print operation.
- At maximum 40 concurrent wire signals (ip2000) have to be analyzed.

The ip8500 electronic consist also of these three main components, however there are differences in the power supply providing 23.2 V DC and 27.0V instead of 24V and differences in the cabling between Printer Controller Board and PCB caused by the higher number of electrical contacts of ip8500 printhead.

6.2.3 Electrical Interface Description

Fig. 19 shows Canon ip2000's printhead contact pin layout as it can be found on the backside of the printhead assembly.

Since no contact pin numbering was identifiable, an own numbering scheme was created.

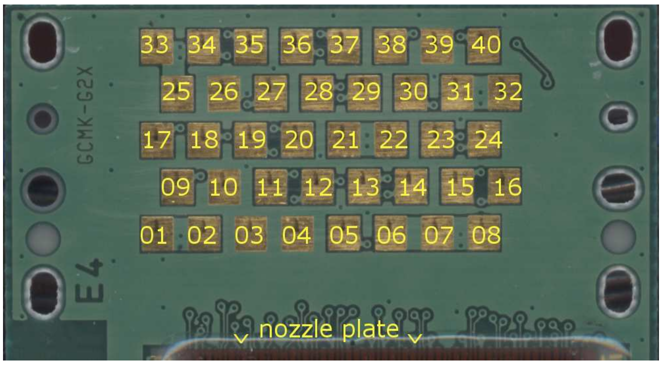


Fig. 19 ip2000 printhead contact pin layout

To be able to clearly set aside pin definitions of different interfaces, a naming scheme was created which assigns an alphanumeric prefix to all required interfaces.

Contact Pin Prefix	Interface-Description	
	Contact point 1	**Contact point 2**
PH-	printhead assembly, see Fig. 19.	printhead carriage
C1-	connector "J201" on printer controller board	cable to printhead carriage PCB
C2-	connector "J202" on printer controller board	cable to printhead carriage PCB

Table 5 ip2000 interface naming scheme

The electrical signals of all listed interfaces have been reverse engineered. There is absolutely no warranty that the findings comply with the vendor’s specifications.

Signal levels marked as TTL use 3.3V positive TTL logic. A pin assignment table for the ip2000 printhead control can be found in appendix A. Further information about the meaning of certain signals can be found in section 6.7.2.

Pin assignment of ip8500 is not identical with ip2000!

6.3 *ip8500 Printhead*

6.3.1 Physical Nozzle Layout

The ip8500 printhead is equipped with 2x5 nozzle arrays split into two nozzle array fields. All 10 nozzle arrays are electrically connected to the printer controller, however only nozzle array 1-8 are connected to ink cartridges. It is assumed, that the same printhead will be used in printer models with 10 ink cartridges.

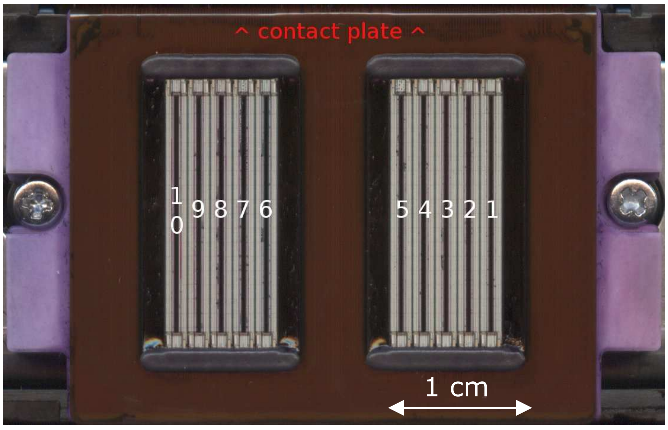


Fig. 20 ip8500 printhead nozzle plate layout

Nozzle Array Number	Ink Color
1	green
2	red
3	photo magenta
4	black
5	photo cyan
6	cyan
7	magenta
8	yellow
9	**not used/no ink supply**
10	**not used/no ink supply**

Table 6 ip8500 nozzle array – color assignment

Property	Public Vendor Specifications [43]	Observed / measured Values
Nozzle arrays	8	10
Nozzle count	8*768 = 6144	10 arrays * 768 nozzles = **7680 nozzles**
Inner nozzle diameter	10 um	
Adjacent nozzle distance	42 um	
Opposing nozzle distance		243 um

Table 7 ip8500 printhead physical properties

6.3.2 Electronic Components

Basically, the ip8500 has the same electronic components as the ip2000.

Power supply:

- Input 230V AC
- Output 0V, +23.2V DC (heating voltage),
  +27V DC (power supply for controller board)
- Connected to printer controller board via 9 wires

Printer controller board:

- ASIC printer processor
- Connection with motors and photoelectric barrier sensors
- Connected to printhead carriage PCB via
  - JCR1: 40-pin ribbon cable, mainly control lines
  - JCR2: 20-pin ribbon cable, mainly power supply lines
  - JCR3: 20-pin ribbon cable, mainly power supply lines

Printhead carriage PCB:

- Connected to printhead via 65-pin connector
  (not all printhead contact-pins are used)

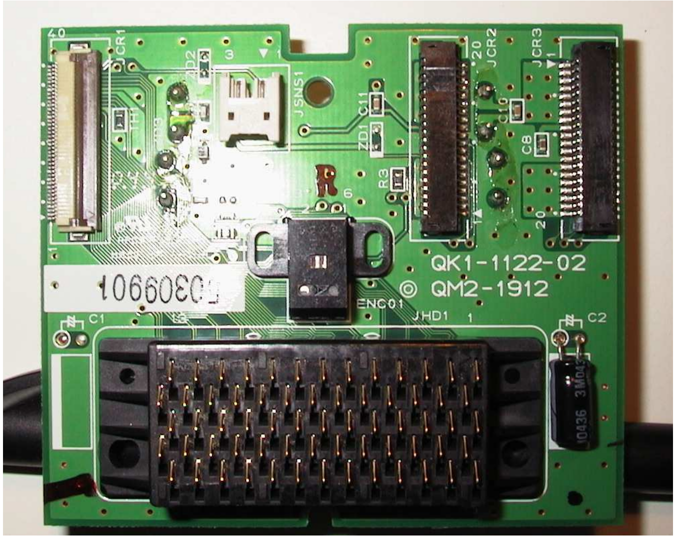


Fig. 21 ip8500 printhead carriage PCB front view

Printhead:

- Nozzle control electronics directly integrated into nozzle plate.

6.3.3 Electrical Interface Description

See Fig. 22 for Canon ip8500's printhead contact pin layout as it can be found on the backside of the printhead assembly.

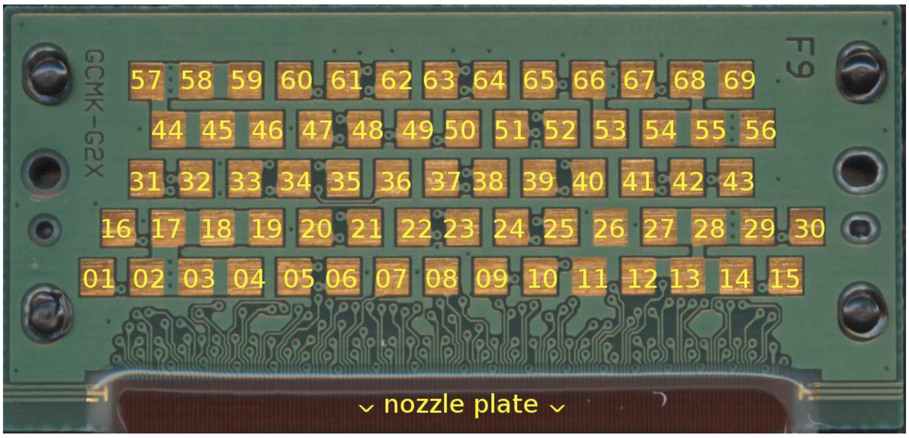


Fig. 22 ip8500 printhead contact pin layout

Alphanumeric prefix definition for electrical interfaces:

Contact Pin Prefix	Interface-Description Contact point 1	 Contact point 2
PH-	printhead assembly, see Fig. 22.	printhead carriage
C1-	connector JCR1 on printhead carriage PCB	cable to printer controller board
C2-	connector JCR2 on printhead carriage PCB	cable to printer controller board
C3-	connector JCR3 on printhead carriage PCB	cable to printer controller board

Table 8 ip8500 interface naming scheme

The electrical signals of all listed interfaces have been reverse engineered. There is absolutely no warranty that the findings comply with the vendor's specifications.

Signal levels marked as TTL use 3.3V positive TTL logic. A pin assignment table for the ip8500 printhead control can be found in appendix B. Attention has to be paid to the fact that the pin numbering of JCR1 is reversed on the printer controller board compared to the printhead carriage PCB. Table B.1 specifies the pin assignment on the printhead carriage PCB side.

Further information about the meaning of certain signals can be found in section 6.7.2.

6.4 Tools Used for Reverse Engineering

The following devices, components and software were used for reverse-engineering the nozzle control signals:

Name/Model	Vendor	Type	Usage
Digital Multimeter	Fluke http://www.fluke.com	device	measure signal level
Oscilloscope WaveMaster 7000	LeCroy http://www.lecroy.com	device	signal analysis
Oscilloscope HM312	Hamag	device	signal analysis
Spectrum Messtechnik MI.7010 digital I/O board 16 bit (+ suitable PC)	Spectrum Messtechnik http://www.spectrum-gmbh.com/mi7010.html	device	signal analysis and generation
Board for wire tapping	two variants one for ip2000, one for ip8500, self made	device	electrical interface
SBench 5.2	Spectrum Messtechnik http://www.spectrum-gmbh.com/sbench.html	software	signal analysis and generation
Matlab 6.5	Mathworks http://www.mathworks.com	software	signal analysis
Excel 2003	Microsoft http://www.microsoft.com	software	signal analysis
GCC 3.4.2	GNU http://gcc.gnu.org	software	signal generator software
Code::Blocks IDE	CodeBlocks http://www.codeblocks.org	software	C++ development IDE
QT 4.1 Open Source Edition	Trolltech http://www.trolltech.com	software	print pattern generator

Table 9 tools used for nozzle control signal reverse engineering

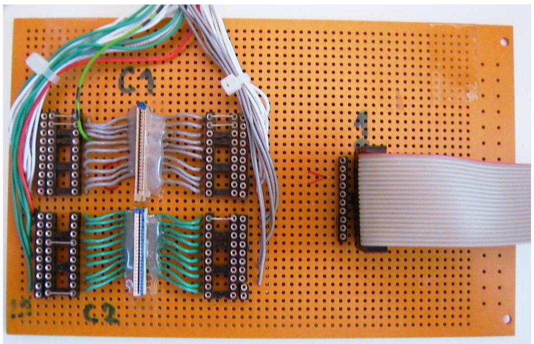


Fig. 23 ip2000 wire-tapping board

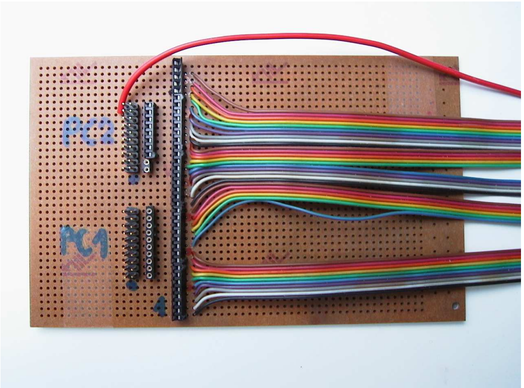


Fig. 24 ip8500 wire-tapping board

The wires of the ip2000 wire-tapping board are soldered to the respective connectors on the printer controller board. The ip8500 wire-tapping board is connected to the rear side of the printhead carriage PCB as shown in Fig. 25, due to better soldering conditions at this location.

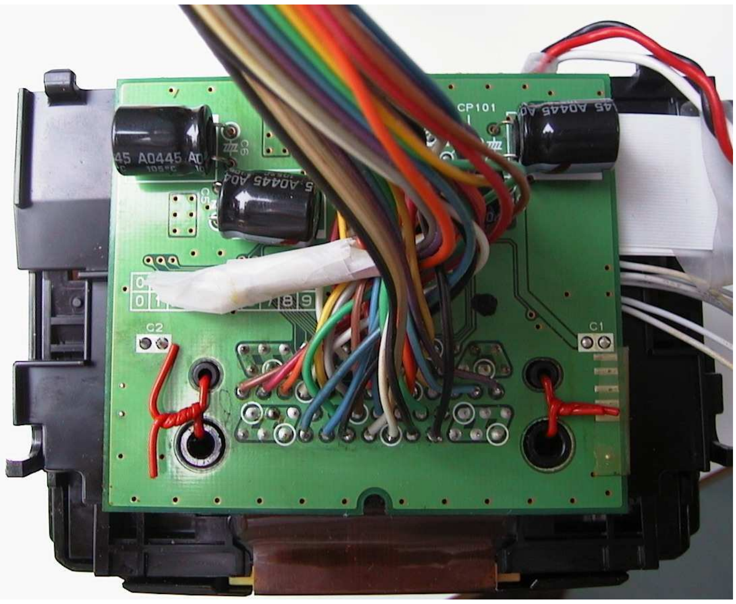


Fig. 25 ip8500 printhead carriage PCB rear side

6.5 *Signal Analysis Process*

Goal is to identify the necessary electrical control signals to drive the nozzles of the printhead without using the manufacturer's printer controller for control signal generation.
The manufacturer's printer electronics can still be useful in the role of a power supply for the printhead at a later step, when the printhead will be controlled by special signal generator software.

The overall process of signal reverse engineering is shown in the following diagram.
The next paragraphs give further details for the ip2000 reverse engineering process, though similar steps were made for the ip8500.

6.5.1 Signal Processing Overview

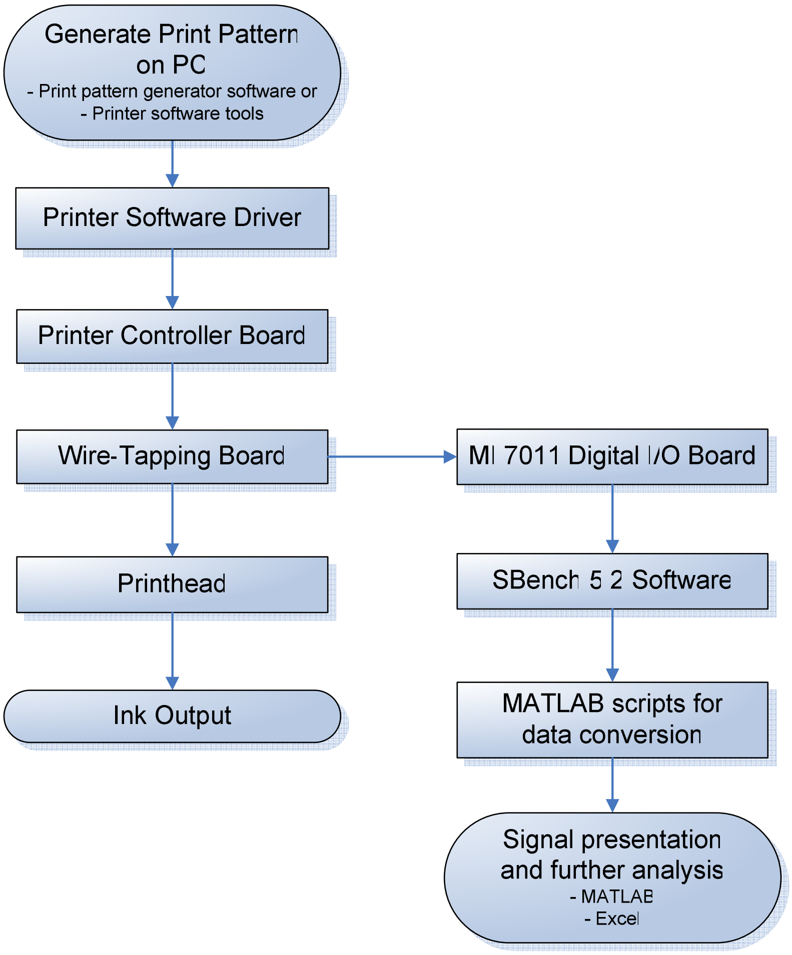


Fig. 26 Signal processing overview

6.5.2 Signal Generation

Printhead signal generation for reverse engineering purposes can be generated via the respective Canon printer driver software or the self diagnostic functions embedded directly into the printer.

It's important for printhead control signal reverse engineering to generate predictable and repeatable drop ejection sequences, which can be brought into relation to specific drop patterns.
The control signal sequences causing these ejections should fit into the sample buffer of the digital I/O card.

In case of the ip2000 a simple image consisting of 45-degree diagonal line with roughly the height of the vertical nozzle size in variation with different line thicknesses was used as input pattern.

Another enlightening input pattern was the nozzle check pattern available through the printer driver maintenance menu and the printer's self diagnostic functions.

6.5.3 Signal Recording

In order to be able to record electrical signal data sent from the ip2000 printer controller board to the printhead and vice-versa, a self-made wire tapping board was created, see Fig. 23.
The board is connected between printer controller board and printer carriage by removing the two 20-pin ribbon cables from their sockets J201 and J202 of the printer controller board and connecting them to corresponding sockets C1 and C2 on the wire-tapping board.
Connection between the printer controller board connectors J201, J202 and the wire-tapping board is established by using wires soldered on both ends.

Initial line classification into power supply lines, active and non-active data lines while printing is done by using a digital Multimeter and WaveMaster Oscilloscope, see Table 9. The tapped line signals are observed while a print job is going on.

The actual signal recording, replay and generation is done with a Spectrum-Messtechnik MI.7011 digital I/O board, see Table 9 for further information. The board is hosted in a standard Personal Computer running Microsoft Windows 2000 and connected to the wire-tapping board. The software SBench 5.2 (see Table 9) was used for actual signal recording and replay.

The used MI.7011 board provides 16 digital I/O ports with a sampling rate of up to 125MHz, 16MB on-board memory as signal buffer and additional input/output ports for triggering.

The card generates 3.3 V TTL compatible output levels and recognizes TTL input levels from 3.3 V – 5 V.
For the ip2000 a 50 MHz sampling rate was chosen in most analysis steps for the MI.7011 board, as the highest detected signal frequency is around 6.2 MHz and a good compromise between sampling accuracy and buffer memory should be achieved.

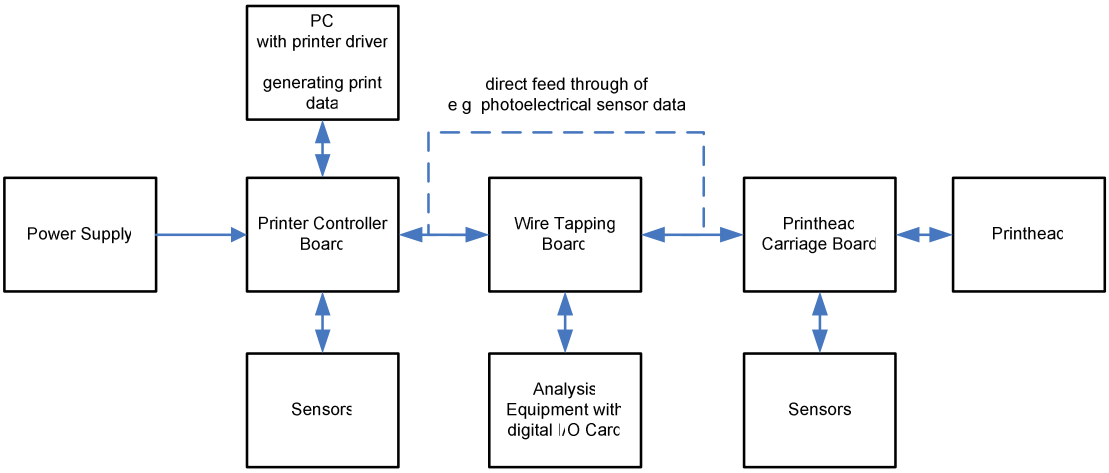


Fig. 27 Electrical Connections for Printhead Control

6.5.4 Signal Analysis

The following steps describe the tasks required for signal analysis.

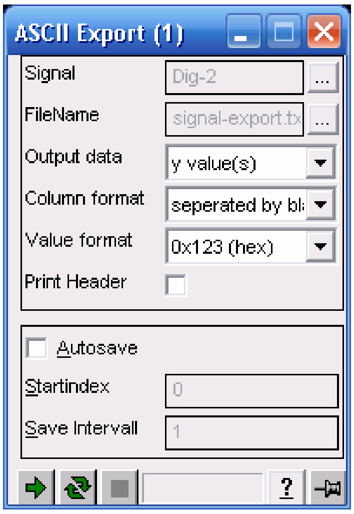
In a first step the signal data of several print test runs was visually analyzed by using the SBench software. Leading to a number of assumptions:

- Data transmission to printhead in serial synchronous mode (with clock line)
- Existence of a clock and reset signal
- Data transmission on every rising and falling clock signal edge

Based on these assumptions a MATLAB script was written to analyze the data further.

6.5.4.1 Data Export with SBench Software

After recording the tapped line signals with the SBench software, the stream of recorded digital samples is exported to an ASCII file with the samples in hexadecimal format. The ASCII file is used as input file for MATLAB analysis scripts. Required SBench export format configuration settings can be seen in Fig. 28:



Signal: Dig-2
FileName: signal-export.tx
Output data: y value(s)
Column format: seperated by bl
Value format: 0x123 (hex)
Print Header: ☐

☐ Autosave
Startindex: 0
Save Intervall: 1

Fig. 28 SBench 5.2 Screenshot, ASCII-Data export settings

6.5.4.2 Dataword Extraction

In this step, the MATLAB script `canon-FINE-bus-analyzer.m`, see appendix C.1, converts the digital samples into a more meaningful binary dataword representation. For a definition of dataword, see 6.7.3.1.

Main script tasks:

- Use clock-line and reset-line signals to identify the start and end of a dataword.
- Identify all raw bits within a dataword on every rising- and falling edge.
- Identify a predefined bit pattern within each dataword and the status of actual data bits.
- Calculate decimal dataword ID value based on predefined bit pattern.

Visualization of script results:

dataline1 dataword raw bits																									dataline2 dataword raw bits																								
																		predefined bit pattern						bit pattern sum																			predefined bit pattern						bit pattern sum
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	14
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	28
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	17
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	22
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	6

Table 10 Sample ip8500 dataword and bit pattern sum visualization

The MATLAB script was verified with several different input patterns and printing modes. Key verification criteria was to have identical recurring sequences of decimal dataword ID values.

6.5.4.3 *Dataword Sequence Analysis*

In the following step, the won data was visually analyzed by using Excel (see Table 9) for recurring patterns.
A thin diagonal line with a thickness of about 5 dots and a height similar to the vertical size of the nozzle array contained in the printhead was chosen as printer input pattern.

The results lead to following assumptions:

- The predefined bit pattern within every dataword is recurring after a number of datawords. The dataword sequence between two recurrences is called a datablock, see also 6.7.3.3.

- The raw bits left to the predefined bit block carrying the actual nozzle activation information.

- The pattern sequence of dataline1 and dataline2 of the same color is equal. Also it can be shifted a few datawords.

- The nozzles of one color are controlled by two datalines.

- All nozzles share the same clock and reset line.

- Control signals for different colors are equal, as long they have the same number of nozzles.

The assumptions were finally tested and proven by writing a printhead control signal generation software (see section 7) and controlling the printhead directly via the Spectrum digital I/O board (Table 9).

6.6 *Reverse Engineering of Nozzle Addressing*

A crucial condition for the applicability of the printhead as volumetric display drop generator is the exact knowledge of how to activate a nozzle at a certain position and point in time.

As analysis showed, the ejection timing problem for the Canon FINE printhead family is mainly related to the frequency of the clock signal and the signal pattern of the respective nozzle array's fireline. See 6.7.7 and 6.7.7.5 for more details.

This section will concentrate on how to identify the addressing of a nozzle at a certain position within the nozzle matrix.

Starting point is the knowledge previously obtained through signal analysis of how to select a specific nozzle array and how to activate a certain nozzle group with the included nozzle sets, see 6.7.4 for details.
Unknown at this point of time is the physical location of a specific nozzle group relative to the printhead's electrical contact plate and the physical location of a specific nozzle set within the nozzle group.

This section uses some terms defined in section 6.7.3, moreover a preliminary version of the printhead signal generator software is required, which allows direct controlling of the printhead via the digital I/O card, see section 7 for further information.

6.6.1 Nozzle Group Physical Location Identification

The process for identifying the physical location of a certain nozzle group consists of the following steps:

1. Generate with the signal generator software a DTB to be sent to the printhead where **all datawords** within the DTB have the same raw bit (see Fig. 36) set to "**1**", for instance raw bit 3, all other raw bits besides the predefined bits (=DWID) shall be set "0".

2. Import the DTB into SBench and send it as "Singleshot Output" (see register "Trig" in SBench hardware settings) about 10 times in short succession to the printhead.

Hint: It needs to be sent in SBench as "Singleshot Output" otherwise DTB transmission/repetition rate could be too high and exceed the max. allowed frequency by the printhead.

3. Watch the drop ejection result on a white paper. A short line covering 1/10 or 1/16 (dependent on the number of actually usable raw bits) of the nozzle array length should be visible.

4. Repeat the test with another raw bit set or decide directly about the relative position of the visible line in relation to the electrical contact plate of the printhead (see Fig. 37 for orientation). Find the nozzle group which creates a continuous line nearest to the electrical contact plate. This nozzle group should be labeled as nozzle group "0" and the corresponding raw bit as "usable bit 0".

5. With the help of a second test pattern the overall nozzle group sequence should be clear, as the relative position of the drop output and the position of the corresponding raw bit are linear.

6.6.2 Identifying the Relative Position of Nozzle Sets

To identify the relative position of a specific nozzle set within a nozzle group requires a different analysis strategy, since the printheads have a very high resolution, making it impossible to decide based on a printed output, which nozzles were active.

The usage of a microscope to look at the nozzle plate while drops are ejected would be a possible methodology, but not very efficient, since the relative location of all nozzles within a nozzle set would have to be determined, caused by the fact that the internal addressing scheme is not linear as in the case of the nozzle groups.

Therefore a more efficient methodology is suggested, the usage of the mathematical Least Squares optimization algorithm [53]. Based on a given series of measured data, the algorithm attempts to find a function which closely approximates the given data. "It attempts to minimize the sum of the squares of the ordinate differences (called residuals) between points generated by the function and corresponding points in the data."[53] Since the algorithm is often used to approximate curves, the process is also called curve fitting.

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

The following steps describe the proceeding used to identify the relative position of nozzle sets by using the Least Squares algorithm.

Starting point is that all previously won reverse engineering data is put into a MATLAB printhead simulation script, see C.2. The script simulates the output which would appear on paper, given a certain printhead control signal input.
The script can address all nozzles of an array and knows the relative location of a nozzle group to the printhead's electrical contact plate. However what is yet unknown is the relative position of all nozzle sets within a nozzle group and the horizontal displacement of the nozzles controlled by dataline1 and dataline2.
Since the relative nozzle set sequence is identical for all nozzle groups, the number of undetermined parameters equals the number of nozzle sets plus one unknown parameter for the horizontal displacement of even and odd nozzles within a nozzle array (all even or odd nozzles are arranged in a vertical line).

1. Generate the image of a diagonal line with a resolution of at least 600 dpi or similar to the hardware resolution of the printhead with a bitmap graphic or the printhead test pattern generator software. The height of the image should cover at least the vertical length of the nozzle arrays. The line should have monochrome color, e.g. black and a thickness of about 1 mm.

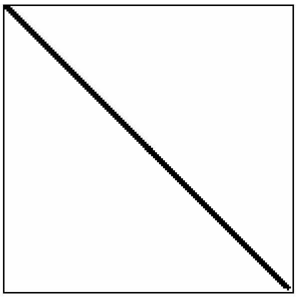


Fig. 29 Image input pattern for nozzle set analysis

2. Print the generated image via the standard printer driver. Simultaneously record the printhead control signals of the black nozzle array with SBench. "Memsize" in SBench should be set to maximum. Pay attention not to start recording before the printhead starts printing the image input pattern, otherwise the necessary samples could not fit into the digital I/O card buffer. The timely triggering can be achieved by using SBench's "Pretrigger" buffer feature, see register "Mem" in hardware settings dialog box, and electrically connect the external trigger input line of digital I/O card with dataline1 of the black nozzle array. Activate SBench's recording button just after printhead cleaning is finished and the printer starts to load paper.

3. Export the recorded data to an ASCII file, see Fig. 28 for format details.

4. Convert the data with the Canon FINE bus analyzer C.1.

5. Check the converted data for the actual start row of printhead control signals for input pattern image generation, by finding the line where the predefined bit pattern starts to be unequal to zero. Helpful commands can be found at the end of the Canon FINE bus analyzer source code, see C.1.

6. Use MATLAB commands to delete all rows preceding the predefined bit pattern start, as they would cause an undesired right shift of the simulated printing output plot.

7. Execute the Canon printhead simulator MATLAB function C.2 and provide the data from the previous step as input data. If all previous steps were executed correctly a plot similar to Fig. 30 should be visible, where a dot pattern from the upper-left to the lower-right corner should be recognizable. Otherwise repeat step 2 to 7.

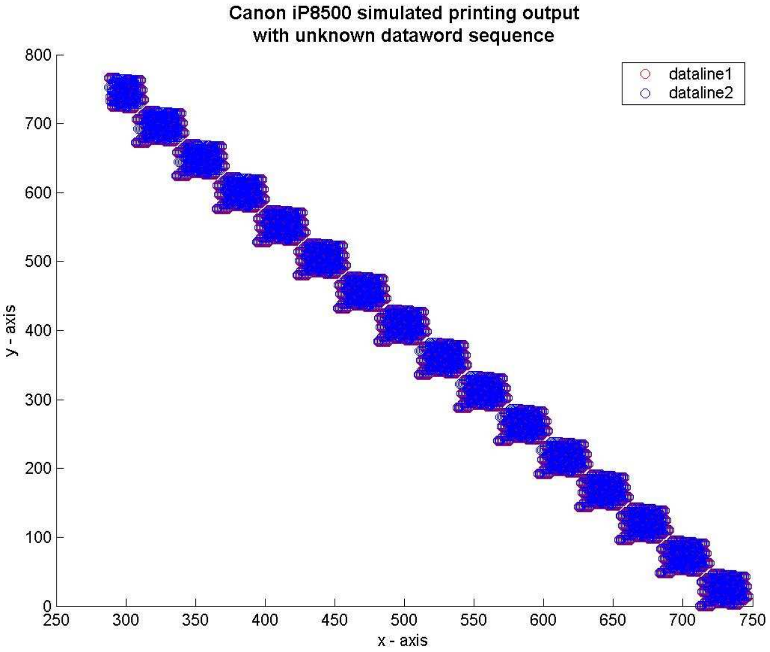


Fig. 30 Simulated printing output before nozzle set positioning optimization

8. Adjust the following parameter in the simulator function to the presented value and rerun the script in order to generate output with similar deviation intervals in positive and negative x-direction for all nozzle sets in relation to the diagonal line:
`MAP_REL_DW_IDX_TO_NOZZLE_SET=[12,12,12,12,12,12,12,12,12,12,12,12,…`
`12,12,12,12,12,12,12,12,12,12,12,12]`.

9. Adjust further the value of the parameter `DATALINE2_HORIZ_OFFSET` until the simulated output of dataline1 and dataline2 is congruent. A result similar to Fig. 31 should be visible.

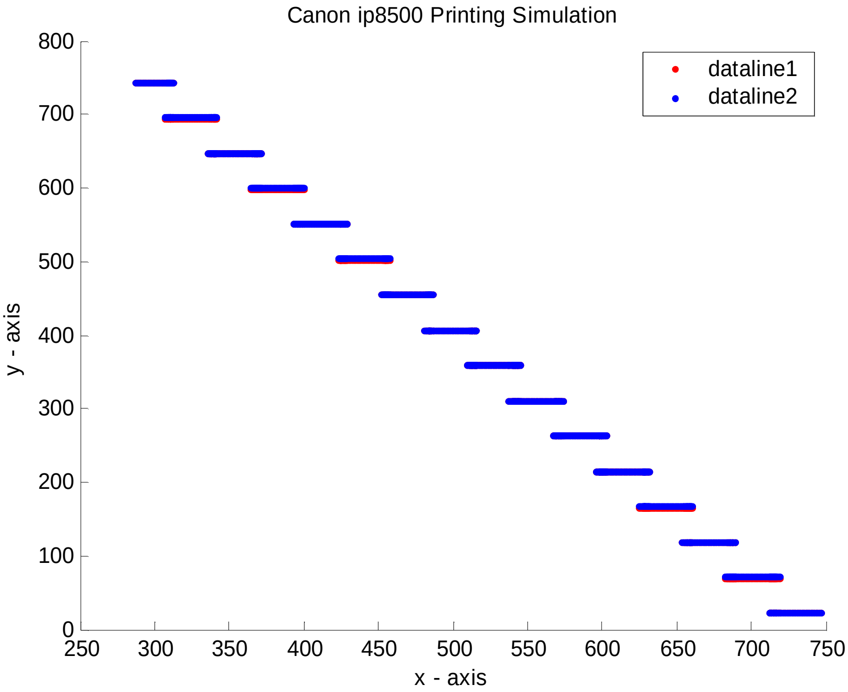


Fig. 31 Simulated printing output adjusted for optimization run

10. Execute a manual linear curve fitting with the help of the MATLAB figure tools. The resulting slope and y-axis offset parameter are used as parameters for the Least Squares optimization reference function, see Fig. 32.

11. Enter the slope and y-axis offset parameter into the Canon Printhead Nozzle Mapping Calculation MATLAB script, see C.4.

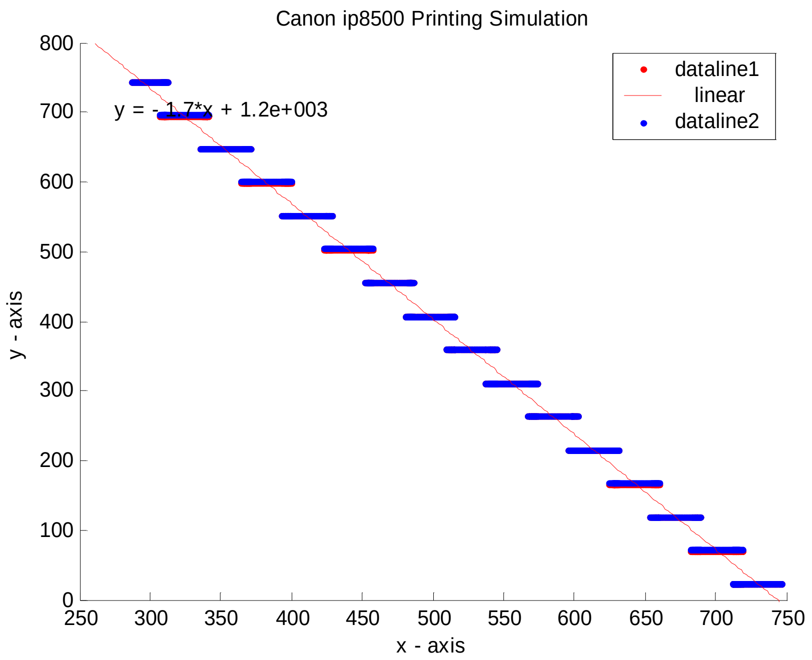


Fig. 32 Simulated printing output with linear curve fitting

12. Execute the `canon_nozzle_mapping_calc()` MATLAB function with the previously generated plot data as input data. The function calculates with the aid of the Least Squares algorithm the optimal nozzle set sequence, so that the simulated output corresponds with the input image pattern. It returns the two variables (= MATLAB vectors) `rel_dw_to_set_map` and `set_to_rel_dw_map`. The first variable reflects the mapping of the predefined dataword ID sequence to the relative nozzle set position. The second variable does the same mapping but in opposite direction from relative nozzle set position to a specific DWID.

13. The values in `rel_dw_to_set_map` shall be used to update the constant `MAP_REL_DW_IDX_TO_NOZZLE_SET` in the Canon Printhead Simulator function script, see C.2.

    In the same way the values in `set_to_rel_dw_map` are used to update the reverse mapping in the Signal Generator source code.

14. After updating the constant `MAP_REL_DW_IDX_TO_NOZZLE_SET` rerun the Canon Printhead Simulator function. The resulting plot should look now like Fig. 33. A zoomed view of the simulated output (Fig. 34) after the optimization run gives strong evidence that the calculated relative nozzle set positions reflect the real printhead parameters. The reason being, that the output plot shows alternating rows, where one row is controlled by dataline1 and the next by dataline2. This is a fact of the real system which was discovered earlier in the reverse engineering process.

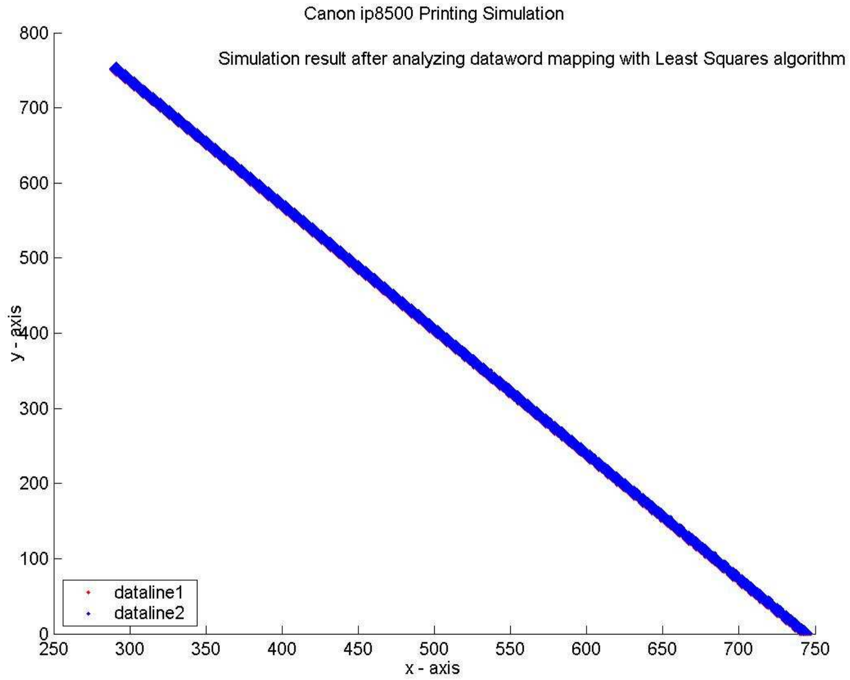


Fig. 33 Simulated printing output after nozzle set optimization

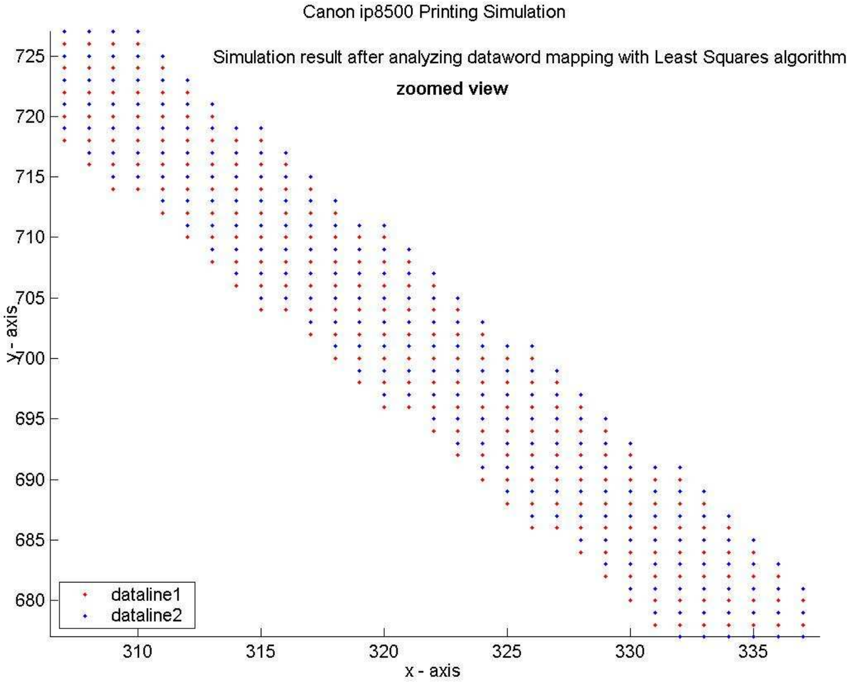


Fig. 34 Simulated printing output after nozzle set optimization – zoomed view

Now all nozzle control signal parameters are known and the theoretical capability is given to command a drop ejection at a certain position. What is still missing is a software which gives distinct and direct control over all printhead nozzles. A solution to this problem is the Signal Generator software presented in section 7.

6.7 Description of Printhead Control Signals

The circumstance, that the Canon FINE printhead family is equipped with a much higher number of nozzles compared to the HP6578 printhead, requires refraining from the matrix based nozzle selection algorithm as communication method between printer controller board and printhead, to avoid a high number of electrical contact pins.

Canon uses instead a synchronous serial data communication between printhead controller board and the printhead. Every nozzle array (color) embedded in the printhead has its **own two datalines** acting in combination with the common reset and clock line as serial buses, allowing to drive all nozzles.

6.7.1 General Findings

Based on the preliminary signal analysis the following findings were made:

- Data transmission between printer controller board and printhead in serial synchronous mode.
- TTL positive logic, high level 3.3 – 3.6V.
- Nozzle array control logic (=addressing scheme) of different colors is identical, as long as they have the same number of nozzles.
- 2 common firelines for all non-black color nozzles (ip2000).
- Cable connected to connector C1 mainly contains printhead control lines (ip2000).
- Cable connected to connector C2 mainly contains printhead power supply lines (ip2000).

6.7.2 Printhead Control Signals Overview

The following figure shows an extract of a typical data transmission sequence between the printer controller board and printhead to drive the nozzles of a single color.

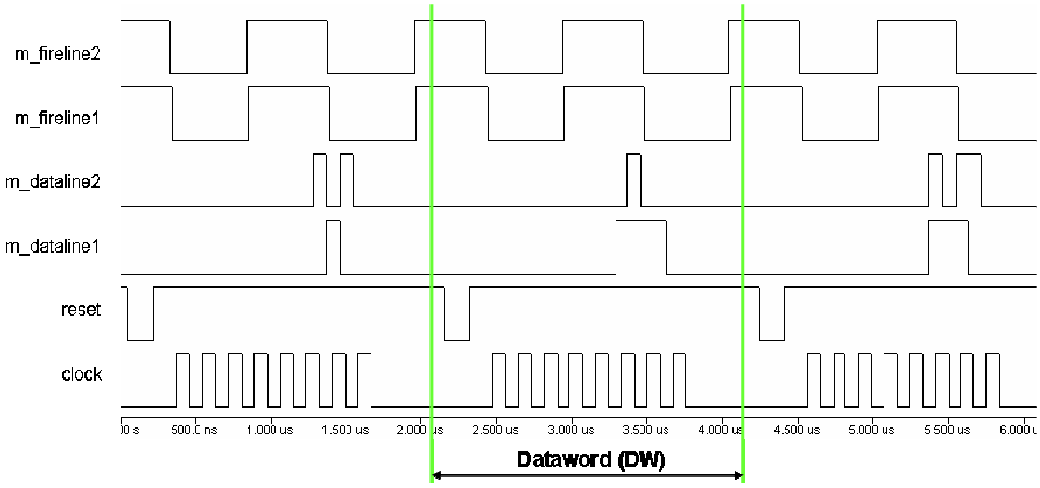


Fig. 35 Printhead control signal overview diagram.
Created with SBench 5.2 software

The "m_" in the above example stands for the control lines belonging to the magenta color nozzle array.

The above diagram is in general also valid for the ip8500, however the ip8500 has more bits per dataword and uses a different timing.

In order to have the ability to drive all nozzles of a specific color, the following control lines are required:

Line Name	Description
clock	common for all nozzle arrays (colors)
reset	common for all nozzle arrays (colors)
*_dataline1	controls all odd nozzles of respective nozzle array
*_dataline2	controls all even nozzles of respective nozzle array
*_fireline1	controls time of firing: o black color nozzles use a single fireline only: b_fireline1 o non-black color nozzles: controls nozzle group 1&3
*_fireline2	controls time of firing: o non-black color nozzles: controls nozzle group 2&4

Table 11 Required control lines per nozzle array

6.7.3 Definition of Terms

6.7.3.1 Dataword (DW)

A dataword (=DW) is defined as the signal sequence transmitted **on a single dataline** between two falling-edge impulses of the reset line, see Fig. 35.

One bit is transmitted with every rising and falling edge of the clock signal, which leads to a maximum of 16 bits (=16 raw bits) transferable per DW for the ip2000, see Fig. 36.
However, each DW includes a predefined bit pattern transmitted as bit 10 to 13 for the ip2000 printhead, reducing the maximum usable data bits to 12 per DW.

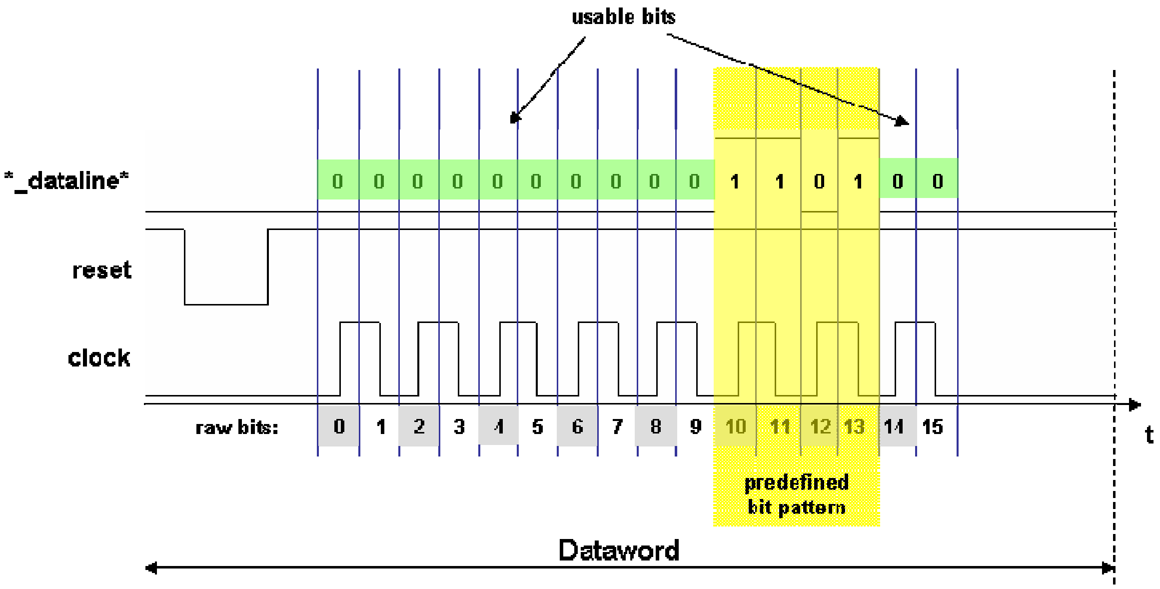


Fig. 36 Bits per dataword

The dataword structure is identical for all nozzle array datalines of a specific printhead. Yet, there may be differences in the number of reasonable bit settings, due to a varying nozzle count.

6.7.3.2 Dataword ID (DWID)

In the further progress of this document the periodically changing bit 10 to 13 bit patterns for ip2000 will be referred to as dataword identification number (=DWID) for easier referencing. The DWID is a decimal number calculated with bit 10 corresponding to the value 2^0 up to bit 13 corresponding to 2^3.

For the ip8500 one dataword comprises 24 raw bits, while the DWID is calculated analogously by using bit 18 to 22.

The DWID has a fixed relation to a corresponding physical set of nozzles.

6.7.3.3 Datablock (DTB)

For the ip2000 a sequence of 16 datawords (DWs) **on a single dataline** is called a datablock (=DTB). The DTB of the ip8500 consists of 24 DWs. Each DW within a DTB is unique, at least due to a changing dataword ID.

One DTB holds the information to address **all even or odd nozzles of a physical nozzle array** (nozzles belonging to a specific color), dependent on which dataline the DTB is transmitted.
Any combination of active nozzles within a DTB is possible.

In order to address all nozzle arrays (nozzles of all colors) of a printhead at the same time, one DTB needs to be sent per nozzle array dataline (each nozzle array having it’s own two datalines).

For a simple printing example it could be said, that one DTB is transmitted per color and horizontal position of the printhead carriage. In reality however, this depends on a number of conditions, like the selected printing- and paper quality, affecting how many drops are put to the same position on paper.

The sequence of DWIDs within one DTB is fixed, but dependent on the printhead direction mode. See 6.7.5.2 for further information.

The following table shows the sequence of DWIDs for the ip2000 and their corresponding bit 10 to 13 bit patterns based on the printhead direction mode. An important fact is the circumstance, that the DWID is not identical to the position of the DW within the DTB.

DW position in this case refers to the transmission point of time on a positive timeline. DW on position 0 is sent first within a DTB.

DWID	NORM Direction Mode					REV Direction Mode				
		Bit Pattern					Bit Pattern			
	DW Pos.	10	11	12	13	DW Pos.	10	11	12	13
0	0	0	0	0	0	15	0	0	0	0
1	10	1	0	0	0	5	1	0	0	0
2	4	0	1	0	0	11	0	1	0	0
3	14	1	1	0	0	1	1	1	0	0
4	8	0	0	1	0	7	0	0	1	0
5	2	1	0	1	0	13	1	0	1	0
6	12	0	1	1	0	3	0	1	1	0
7	6	1	1	1	0	9	1	1	1	0
8	1	0	0	0	1	14	0	0	0	1
9	11	1	0	0	1	4	1	0	0	1
10	5	0	1	0	1	10	0	1	0	1
11	15	1	1	0	1	0	1	1	0	1
12	9	0	0	1	1	6	0	0	1	1
13	3	1	0	1	1	12	1	0	1	1
14	13	0	1	1	1	2	0	1	1	1
15	7	1	1	1	1	8	1	1	1	1

Table 12 ip2000 dataword sequence within a datablock

It is assumed that the direction dependent dataword sequence is related to the circumstance, that dependent on the moving direction of the printhead carriage, varying nozzles pass over an unprinted area first. In order to equal the geometric nozzle offset of different moving directions, the dataword sequence is adjusted.

Since this effect has no impact on the further basic volumetric display tests it won't be investigated any further.

6.7.4 Internal Nozzle Addressing Scheme

The addressing and activation of a single nozzle happens at four levels:

- Level1:
  define the nozzle array to which the activate nozzle belongs to, indicating the required datalines (dataline1, dataline2) and the corresponding fireline.

- Level2:
  decide if the activate nozzle is on the even or odd side of the nozzle array, see Fig. 37. This defines if dataline1 or dataline2 has to be used.

- Level3:
  select a DW with a specific DWID, defines the physical nozzle set which should become active, see Fig. 37.

- Level4:
  setting one of the usable bits (see 6.7.3.1) within the selected DW to a logical "1" defines the physical nozzle group to become active.

These four levels of selection result in exactly one nozzle to be activated. By combining several selection iterations, it's possible to create any combination of active nozzles within one or more datablocks.

For the organization of a physical nozzle array with sets and groups see Fig. 37 on the following page.

Fig. 37 shows on the example of the ip2000 black-nozzle array the internal addressing scheme. Dataline1 of a nozzle array controls all even nozzles, dataline2 all odd ones.
The number of nozzle sets is 16 and corresponds to the number of datawords. The number of nozzle groups is dependent on the number of usable bits within one DW, see Fig. 36. The ip2000 uses bit 0-9 and bit 15 for nozzle group addressing. The usage of bit 14 seems to have no effect at least for the black nozzle array.

In general all nozzle sets are contained in all nozzle groups, however the ip2000 black nozzle has the specialty that the nozzles 0 to 7 and 328 to 335 belong to nozzle groups comprising only 4 nozzle sets each.

The physical location of the nozzle group addressed by raw bit 0 is the one located nearest to the electrical contact plate (seen from rear side onto nozzle plate). Nozzle groups corresponding to increasing raw bit numbers have subsequently larger distances to the electrical contact plate. An exception to this is the ip2000 black-nozzle array group 15, which is split into 2 segments surrounding the other groups.

The nozzle addressing scheme is analogous to the ip2000 non-black color nozzles and ip8500. Due to the varying nozzle count however, the sequence of datawords and corresponding raw bits is not transferable 1:1.

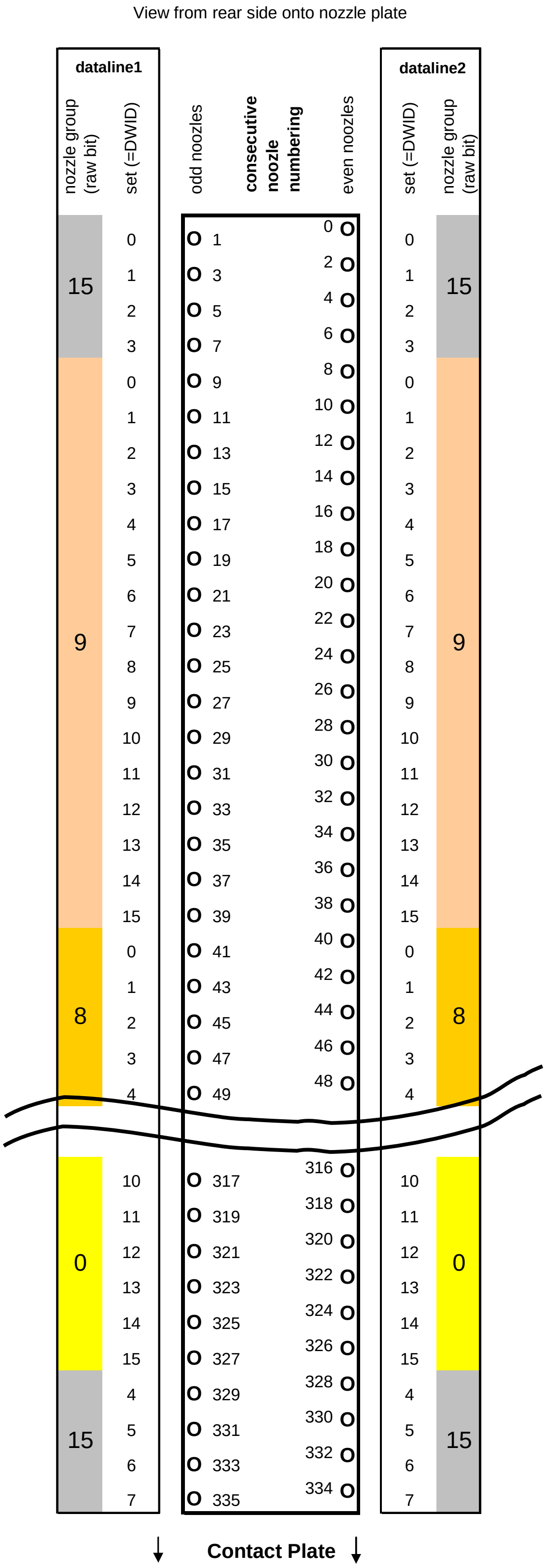


Fig. 37 Physical nozzle array organization of ip2000 black-nozzle

The nozzle addressing scheme is completely based on reverse engineering data, since no official information was available.

6.7.5 Printhead Operation Modes

6.7.5.1 Initialization and Preparation

The knowledge won during the reverse engineering process indicates that the printheads of the Canon FINE family need to be initialized first with an electrical signal sequence before they can be used for drop ejection.

However, there is no strong evidence for this indication, as nearly all experiments were executed by using the manufacturer's printer controller board for providing power to the printhead. These printer controller boards though, provide the required voltages for drop ejection (heating voltage) and the assumed initialization codes to the printhead only, if a number of power-on self tests were executed correctly.

Parts of these power-on self tests are:

- Printhead carriage movement and feedback via horizontal incremental position sensor.
- Check of light barrier sensors.
- Identification of carriage stopper position.
- Identification of printhead (serial number).

In consequence of the decision to keep the manufacturer's printer controller in the role of a printhead power supply and thus keep electrical engineering efforts at a minimum, measures had to be taken to let the printer controller pass power-on self tests even after the printhead was removed for experiment purposes from the moving printhead carriage.

In the case of the ip8500 printer, the solution for the problem was to replace the original printhead carriage assembly with a similar one from an ip4000 printer and use the original one as printhead interface and holder at a fixed position outside the printer.

Additionally the connectors of the two light barriers sitting on the original printhead carriage PCB were extended and connected to the ones located on the ip4000 printhead carriage PCB.

All lines required for passing the power-on/initialization phase successfully, can be found in the pin assignment tables in appendix A and B marked with the status flag "RI".

Crucial for an error free printhead operation is furthermore the assurance that all nozzles have a steady ink supply. As long as the printhead is located within the printer enclosure, the embedded vacuum pump ensures that fresh ink moistens all nozzles before any printing operation and that no air bubbles remain in the ink feed.

After removing the printhead from the printer's enclosure this task has to be done manually, e.g. by drawing ink with the help of an injection and a silicone tube, see Fig. 38, via the nozzle plate.
Experience has shown that a printhead missing this preparation after an interval of about four hours of printing inactivity is not able to produce a uniform drop output.

Fig. 38 Ink drawing via nozzle plate

6.7.5.2 Ink Ejection Modes

The Canon printer driver allows the selection of three different printing quality modes:

- Draft quality (DQ)
- Standard quality (SQ)
- High Quality (HQ)

Experiments showed that the printing quality setting mainly influences the printhead control signals by adjusting the time interval between DWs (see Fig. 39). In combination with different max. printhead carriage velocities, this leads to a higher count of ejectable drops per horizontal position in HQ mode compared to DQ mode.

Fig. 39 ip8500 printhead control signals – printing quality modes (the black squares represent the clock impulses of one DW)

The signaling data behind Fig. 39 shows further, that ip8500's minimum DW cycle time of 1567 ns is reached in HQ and SQ mode. This would lead to a maximum drop ejection frequency of approx. 26.6 kHz (=1/(24 DW*1567 ns)).

Under actual printing conditions however, the max. drop ejection frequency lies at about 23.8 kHz, due to a longer pause after every 6th DW.

It is assumed that the purpose of this pause after a certain number of DWs is used to synchronize drop ejection with horizontal carriage movement. No further investigations were made on this topic, as it is not relevant for volumetric display tests.

Besides drop ejection frequency, the printing quality mode selection also has influence on the printhead carriage movement direction during drop ejection.

In DQ mode drop ejection is done bidirectional, in HQ mode unidirectional.

The printhead's carriage directions are defined as:

Printhead Carriage Direction Name	**Description**
NORM	normal operation, printer seen from front: printhead carriage goes from right to left
REV	reverse operation

The printing direction is also reflected by the DWID sequence within a DTB, as shown in Table 12. Again, no further investigations were made on this topic, due to it's irrelevance for volumetric display experiments.

6.7.6 Initialization Signals

As mentioned in section 6.7.5.1, it is assumed that Canon FINE printheads need to be initialized first with a sequence of electrical signals before they can be used for drop generation.

A series of tests was done with an ip2000 printhead on this topic leading to the conclusion that only the lines C1-3, C1-11, C1-12, C1-13 are relevant for ip2000 printhead initialization, see appendix A. Similar lines can be identified for the ip8500 in appendix B.

The experiments indicate further that after a successful initialization the printhead can operate normally for about 24 hours without further initialization. After 24 hours the number of nozzles for which drop ejection is not possible raises steadily. However it was not possible to completely exclude that this behavior might be also related to a problem with the ink supply.

Samples of traced ip2000 initialization signals can be found on the CD accompanying this paper.

General findings concerning ip2000 initialization signals:

- Transmission of initialization signals is independent of clock and reset signals.
- Init signal triggering should be done with C1-13.
- Init lines show short time activity at power-on, after printhead replacement, short before actual printing operation starts, after printing operation ends and during soft power-off.
- The signal sequence changes slightly with every test run.
- Assumption: initialization communication is bidirectional, due to printhead serial number requirement for correct printhead nozzle alignment.

6.7.7 Printhead Control Signals

6.7.7.1 Signal Timings

Fig. 40 shows the timing characteristics of the traced printhead control signals. The respective quality mode dependent time values for the ip8500 can be found in Table 13.

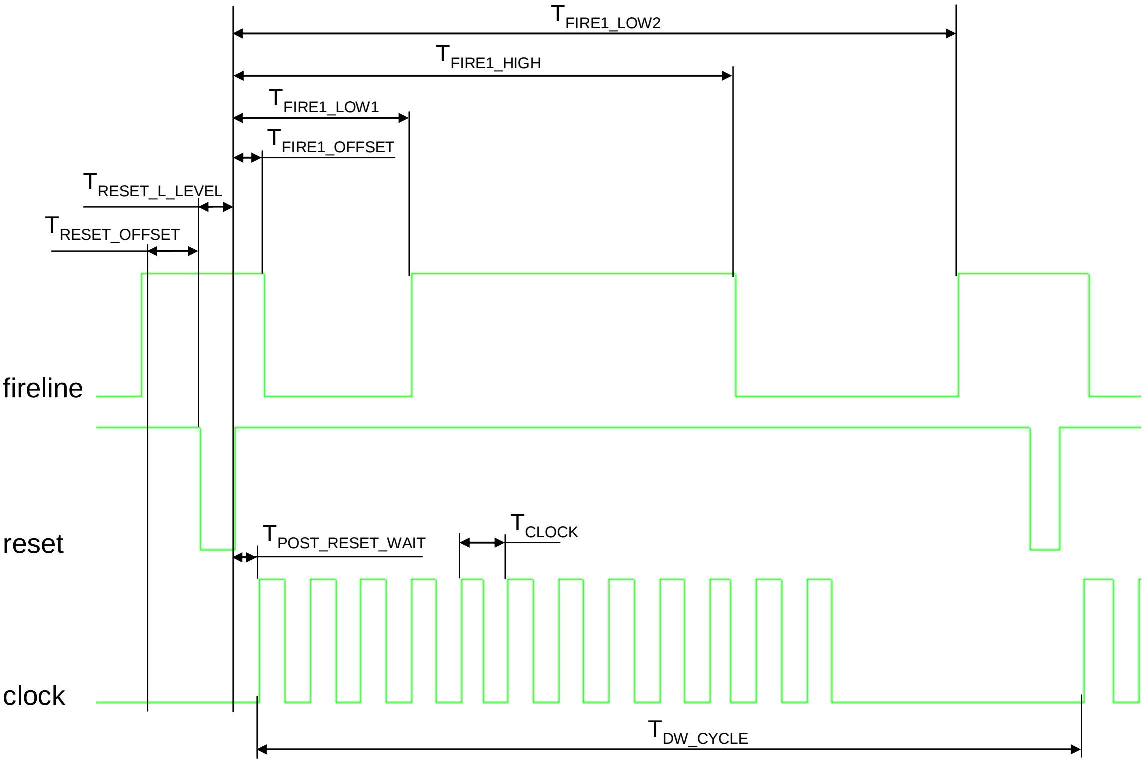


Fig. 40 Canon FINE printhead control signal timing characteristics

Time Variables	DQ Mode [ns]	SQ Mode [ns]	HQ Mode [ns]
T_{CLOCK}	95	95	95
T_{DW_CYCLE}	2122	1558	1570
$T_{RESET\_L\_LEVEL}$	64	64	64
$T_{POST\_RESET\_WAIT}$	46	46	46
T_{FIRE1_OFFSET}	54	54	54
T_{FIRE1_LOW1}	282	282	282
T_{FIRE1_HIGH}	622	622	622
T_{FIRE1_LOW2}	424	424	424

Table 13 ip8500 control signal timings

The values in Table 13 were measured at a sampling frequency of 125 MHz with subsequent mathematical average calculation over different nozzle arrays and at least four sample sets.

The signal timings for the ip2000 are more complex, as this printhead doubles it’s clock signal frequency in HQ and SQ mode compared to DQ mode and has overall four different timing modes for its firelines. The respective timings can be found in the source code of the Signal Generator software.

6.7.7.2 Clock Signal

The clock signal is a 3.3V TTL signal common for all nozzle arrays and required for any nozzle control operation. The printhead reads a data bit from its datalines with every rising and falling edge of the clock signal, as shown in Fig. 36.

The timing characteristics of the clock signal can be seen in section 6.7.7.1, the respective pin assignment in appendix B.

The clock signal should start at least one DTB before any actual data transmission begins on the datalines.

In phases of inactivity the line shows a LOW state.

6.7.7.3 Reset Signal

Like the clock signal, the reset signal is a 3.3V TTL signal common to all nozzle arrays and crucial for any nozzle control operation. The respective line (see appendix B) was given the name "reset signal", since the line shows a short low impulse before the clock impulses for the associated DW start. Respective timings can be found in section 6.7.7.1.

In phases of inactivity the line shows a HIGH state till the printer controller switches to stand-by mode and stops power supply to the printhead.

Like the clock signal, the reset signal starts at least one DTB before any actual data transmission occurs.

6.7.7.4 Dataline Signals

The datalines' purpose is to transmit the data bits required for nozzle selection to the printhead sequentially.
Each nozzle array is controlled by two datalines. Dataline1 for controlling all odd nozzles and dataline2 for all even nozzles See Fig. 37 for a definition of even and odd nozzles.

The addressing scheme is identical for all nozzle arrays with the same number of nozzles, see 6.7.4 for details.

Also, within a nozzle array the addressing scheme is identical for dataline1 and dataline2. There might be a DTB offset of e.g. 8 DWs, as is the case with the ip2000. More details can be found in the ip2000 Excel data analysis file on the CD accompanying this paper. Such a dataline offset doesn't exist for the ip8500.

Crucial for an error free data transmission to the printhead is the circumstance that the desired dataline bit state is present at the printhead's contact pins right before the clock signal changes its state and triggering the printhead to read bits from its datalines.
The respective pre-clock-signal edge offset for assuring this condition can be set in the Signal Generator software with the header constant `DATA_BIT_PRE_EDGE_OFFSET`.

For control signal analysis purposes, a comparable parameter can be set in the Canon FINE bus analyzer MATLAB script (see C.1) with the variable `bit_read_sample_offset`.
The correct setting of this parameter can be verified by creating a decimal sum over all DWIDs within a DTB, if the sum is identical for all DTBs the `bit_read_sample_offset` variable is set correctly.

6.7.7.5 Fireline Signals

The respective lines are called firelines, since it is assumed that these lines control the heating duration and as consequence of this the drop ejection point of time of the previously addressed nozzles. An indication for this assumption is the circumstance that no drop is ejected of a nozzle array if the respective fireline is not connected.

The fireline timing characteristics can be found in section 6.7.7.1. Due to limited time, no investigations could be made concerning the impact of modified fireline timings on the drop ejection process.

A fireline can be shared by several nozzle arrays.

Reverse engineering results showed for the ip2000 at least four different fireline signal timing modes. The respective timings can be found in the respective C++ header file of the Signal Generator source code. The non-black color nozzle arrays of the ip2000 represent furthermore a special case, as one array is separated into two fireline groups each controlled by its own fireline.

7. Canon Printhead Signal Generator Software

7.1 Purpose

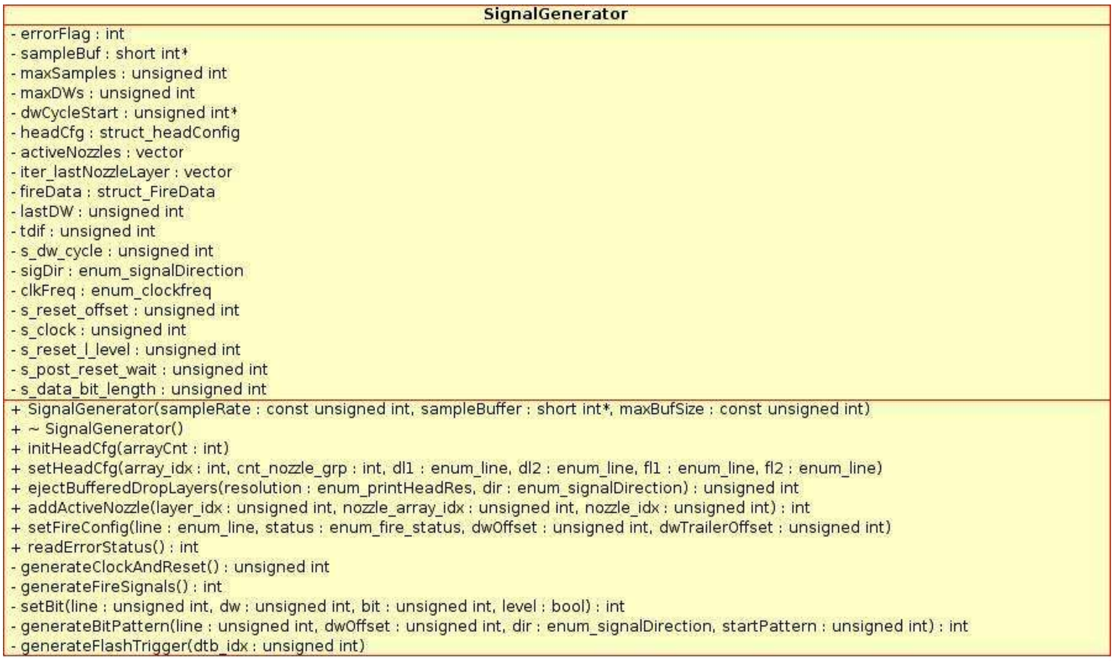
As reverse engineering analysis showed, it is not possible to use Canon’s printhead control electronics and related printer driver software to address nozzles in a way that volumetric output patterns can be generated, therefore another method had to be found generating the desired nozzle control signals.

The Signal Generator software’s purpose is to provide a simple interface for direct nozzle control of Canon FINE printheads by using a digital I/O board. The software should completely abstract timing settings for specific printheads. Additionally it should generate the trigger impulse for a stroboscope. On the other hand the software is not designed for real-time nozzle control based on changing input patterns.

7.2 Design

The software consists of a single class the `SignalGenerator` class see Fig. 41. The class is written in compliance to the ANSI C++ standard, uses the Standard Template Library and is therefore platform independent.

For direct signal output via the Spectrum digital I/O board additionally the class `Spectrum` is used (see Fig. 42), which is heavily based on a Spectrum GmbH driver programming example. The `Spectrum` class requires the Spectrum MI.7010 Windows driver library. See Table 16 for Spectrum digital I/O board reference information.



```
SignalGenerator
- errorFlag : int
- sampleBuf : short int*
- maxSamples : unsigned int
- maxDWs : unsigned int
- dwCycleStart : unsigned int*
- headCfg : struct_headConfig
- activeNozzles : vector
- iter_lastNozzleLayer : vector
- fireData : struct_FireData
- lastDW : unsigned int
- tdif : unsigned int
- s_dw_cycle : unsigned int
- sigDir : enum_signalDirection
- clkFreq : enum_clockfreq
- s_reset_offset : unsigned int
- s_clock : unsigned int
- s_reset_l_level : unsigned int
- s_post_reset_wait : unsigned int
- s_data_bit_length : unsigned int
+ SignalGenerator(sampleRate : const unsigned int, sampleBuffer : short int*, maxBufSize : const unsigned int)
+ ~ SignalGenerator()
+ initHeadCfg(arrayCnt : int)
+ setHeadCfg(array_idx : int, cnt_nozzle_grp : int, dl1 : enum_line, dl2 : enum_line, fl1 : enum_line, fl2 : enum_line)
+ ejectBufferedDropLayers(resolution : enum_printHeadRes, dir : enum_signalDirection) : unsigned int
+ addActiveNozzle(layer_idx : unsigned int, nozzle_array_idx : unsigned int, nozzle_idx : unsigned int) : int
+ setFireConfig(line : enum_line, status : enum_fire_status, dwOffset : unsigned int, dwTrailerOffset : unsigned int)
+ readErrorStatus() : int
- generateClockAndReset() : unsigned int
- generateFireSignals() : int
- setBit(line : unsigned int, dw : unsigned int, bit : unsigned int, level : bool) : int
- generateBitPattern(line : unsigned int, dwOffset : unsigned int, dir : enum_signalDirection, startPattern : unsigned int) : int
- generateFlashTrigger(dtb_idx : unsigned int)
```

Fig. 41 UML diagram of SignalGenerator class.
Created with Umbrello [54].

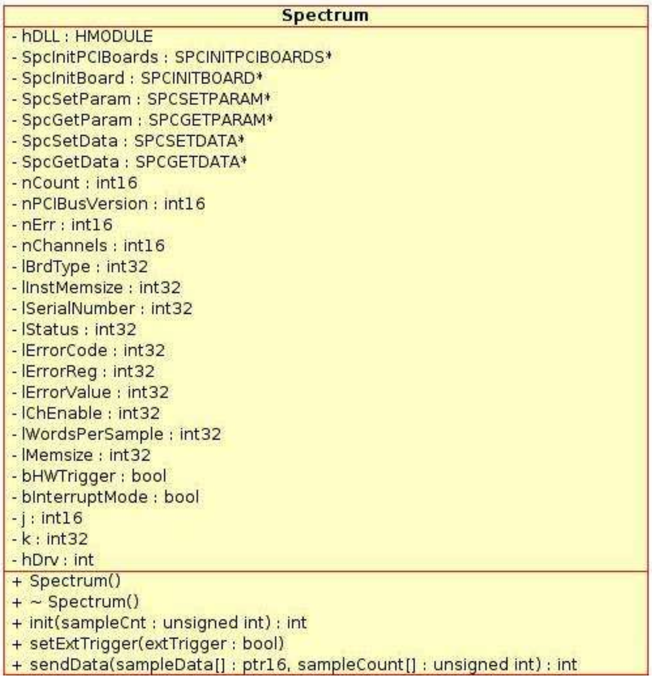


Fig. 42 UML diagram of Spectrum class.
Created with Umbrello [54].

Spectrum MI.7010 digital I/O board line assignments are currently hard coded for certain printhead control signals into the Signal Generator's source and can be found in Table 14.

Line (Signal) Name	Line Number
clock	0
reset	1
fireline1	4
fireline2 (for ip8500 same as fireline1)	5
stroboscope trigger	15
datalines	2,3,6-14

Table 14 Signal Generator software signal line assignment

7.3 Function Overview SignalGenerator Class

More details concerning the functions can be found in the Signal Generator's source code comments on the CD accompanying this paper.

Description of important public SignalGenerator class functions:

`initHeadCfg()`
Initializes a SignalGenerator object with the number of nozzle arrays of a specific printhead.

`setHeadCfg()`
Defines the number of nozzles per nozzle array and the respective signal lines to control an array. This function needs to be called for each nozzle array of a printhead.

`addActiveNozzle()`
Adds the coordinates of a nozzle which should be ejected to the output pattern. The function takes as first parameter a `layer_idx`, which corresponds to a DTB index, defining the relative point in time when a specific nozzle should be ejected. The second parameter is a `nozzle_array_idx` corresponding to a specific nozzle array (=nozzle color).
The third parameter defines the `nozzle_idx`, which is a linear index value of a certain nozzle within a nozzle array, see Fig. 37 for reference. The nozzle index starts with 0 (largest distance to electrical contact plate) and ends with n-1 (n=number of nozzles per array).

`ejectBufferedDropLayers()`
After adding nozzles to one or more layers with **`addActiveNozzle()`** is finished, the **`ejectBufferedDropLayers()`** function is called to generate the resulting output pattern. The output pattern is stored in a 16 bit wide (short int) sample buffer.

7.4 Usage

The Signal Generator software works as command line program in the Command Prompt Box of Windows. In the experimentation setup, the software's source code is edited and compiled with the Code::Blocks integrated development environment and the included MinGW compiler, see Table 16 for reference.

The software currently supports no input interface for nozzle control data, therefore all nozzle activation commands have to be "hardcoded" as C++ function calls in the source code. This was done to have maximum flexibility during experimentation and no limitations by a certain input file format.

Function calls to `addActiveNozzle()` like for instance generated by the `canon_2d_pattern_generator.m` MATLAB script (C.5), should be placed into the `main.cpp` file at the line marked by the corresponding comment.

After re-compilation the program can be executed with the following command on the Windows command prompt:

```
signal-generator.exe –o[filename_for_ASCII_HEX_signal_output]
```

Hint: no space is allowed between "-o" and the filename.

During execution the Signal Generator software creates a lot of debug information in the command window by default. Under normal conditions these information can be ignored. The generated signal output file can be imported in a next step by the SBench software. For a complete pattern generation workflow see section 8.4.

The other command line options of the Signal Generator software are not discussed here, since the direct signal output via the Spectrum board turned out to be helpful only in a very limited experimentation scope.

8. 3D-Image Generation Experiments

Numerous drop ejection test series with ip2000 and ip8500 printheads were conducted preliminary to the 3D-image generation experiments with the aim to verify the correct operation of the programmed Signal Generator software, to verify that all nozzle arrays can be addressed and to reach repeatable drop output results.
The drop output results were checked by placing a piece of white paper directly underneath the printhead nozzle plate and sending the identical drop output sequence several times to the printhead.

No further information on the preliminary tests will be given as this section focuses mainly on test series conducted to verify the basic principle of 3D-image generation by using inkjet printheads, as described in section 3.1.

8.1 Experiment Setup

A schematic of the experiment setup is shown in Fig. 43. The actual laboratory experiment setup with an ip8500 printhead can be seen in Fig. 44.

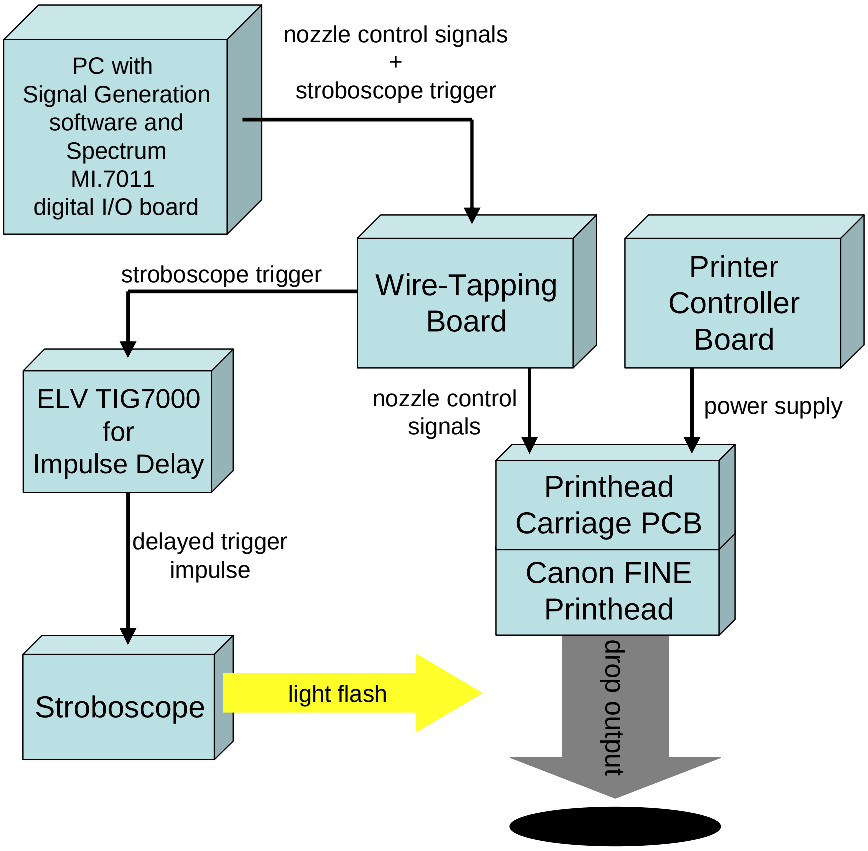


Fig. 43 3D-image generation schematic experiment setup

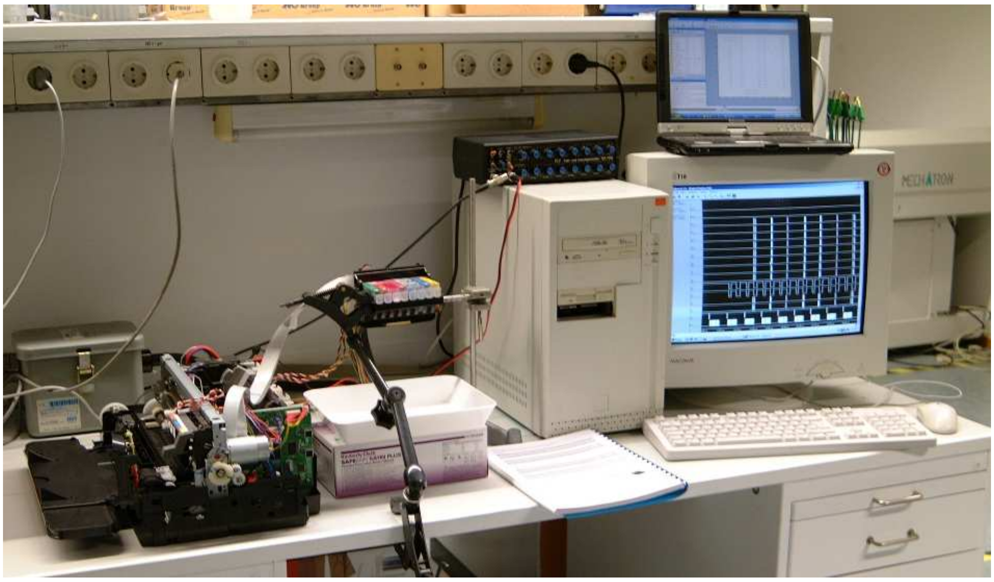


Fig. 44 3D-image generation laboratory experiment setup

8.1.1 Boundary Conditions

All experiments were conducted under the following boundary conditions:

- Usage of standard inkjet ink, same as used for printing.
- Normal surrounding air-pressure.
- Generation of static output patterns, due to constraint of 16 MB sample memory of digital I/O board.
- Sample rate 125 MHz for ip8500, 50 MHz for ip2000.

Part of the timing definitions for ip8500 experiments can be found in Table 15. The meaning of the timings can be seen in Fig. 40.

```
// all time values in nano seconds
#define T_RESET_OFFSET_LF        32
#define T_CLOCK_LF               96
#define T_RESET_L_LEVEL_LF       64
#define T_POST_RESET_WAIT_LF     48
#define T_DW_CYCLE_LF            2120
#define T_FIRE1_OFFSET_LF        56
#define T_FIRE1_LOW1_LF          280
#define T_FIRE1_HIGH_LF          624
#define T_FIRE1_LOW2_LF          424
#define T_FLASH_TRIG_OFFSET_LF   300400
#define T_FLASH_TRIG_HIGH_LF     200
```

Table 15 ip8500 experiment time settings for Signal Generation software

All Signal Generation software parameters can be found in the Signal Generator's C++ header files:

- `config.h`
- `signalgenerator.h`
- `spec-ip8500.h` (ip8500) or `spec-ip2000.h` (ip2000)

8.1.2 Tools used

The following special devices, components and software were used for conducting the 3D-image generation experiments:

Name/Model	Vendor	Type	Usage
Spectrum Messtechnik MI.7010 digital I/O board (+ suitable PC)	Spectrum Messtechnik http://www.spectrum-gmbh.com/mi7010.html	device	signal generation
ELV Takt- und Impulsgenerator TIG7000	ELV Elektronik AG http://www.elv.de	device	trigger delay for stroboscope
EG&G Xenon Stroboscope lamp	EG&G now: http://www.perkinelmer.com	device	generate flash light
Board for wire tapping	two variants one for ip2000, one for ip8500, self made, see section 6.4.	device	electrical interface
Windows 2000 Workstation	Microsoft http://www.microsoft.com	software	operating system
SBench 5.2	Spectrum Messtechnik http://www.spectrum-gmbh.com/sbench.html	software	Spectrum board control
Matlab 6.5	Mathworks http://www.mathworks.com	software	input pattern transformation
MinGW Compiler Set	MinGW http://www.mingw.org/	software	compiler for Signal Generator
Code::Blocks IDE	CodeBlocks http://www.codeblocks.org	software	C++ development IDE

Table 16 Experimentation tools

8.2 Crucial Experimentation Findings

- **In order to prevent electrical damage to the printhead, the digital I/O board must not be connected to the printhead (wire-tapping board) during PC power-on or off.** The digital I/O card should be connected to the printhead only after successful initialization through SBench or Signal Generator software.

- When sending nozzle control signals to the printhead by using SBench's "Continuous Output" function **NEVER** stop or abort the signal while playing **and the printhead heating power supply is on**, otherwise nozzle heating elements will burn through.

- Signal triggering for nozzle control signal analysis should be done by connecting the external trigger line of the digital I/O card to the printhead's reset line.

- The printhead nozzles are sensitive to a continuous ink supply, after several hours of drop ejection inactivity, ink should be drawn over the complete width of a nozzle array with the help of an injection and a silicone tube, see Fig. 38.

8.3 Conducted Test Series

This section lists some of the 3D-image generation experiments conducted with ip2000 and ip8500 printheads. Values defined in this section reflect settings for the ip8500 printhead if not specified otherwise.

8.3.1 Stroboscope Trigger Timing Investigation

Objectives:
Identify timing for stroboscope trigger impulse in order to get a steady image impression directly underneath the nozzle plate.

Test Patterns:
Horizontal lines using all even (odd) nozzles of a single nozzle array with different thicknesses and different vertical spacing.

Test patterns were created directly as C++ source code.

Results:

1. Trigger timing:
Set to a hard coded value of 311.1 µs in the Signal Generator software, see function `generateFlashTrigger()` in the Signal Generator source code for reference.
One identical trigger impulse is generated per frame by the Signal Generator software.

2. DTB repetition (line thickness):
An identical DTB has to be repeated in direct succession at least 3 times in order to get the visual impression of a horizontal line.
Cause unknown, needs to be further investigated.

3. Inability to use ip2000 non-black nozzle arrays:
Numerous trigger tests were conducted with ip2000 non-black nozzle arrays, however an adequate trigger timing could not be found to get a clear visual impression of horizontal lines. Yet similar tests with the ip2000 black-nozzle array were successful.
Based on the test results it was concluded, that the alternating nozzle size of ip2000 non-black nozzle arrays (see Fig. 17) with the resulting unequal drop sizes, make it impossible to get a clearly recognizable image output, if all nozzles of the respective arrays are in use.

Therefore all further tests with ip2000 printheads were stopped. Subsequent tests will focus on ip8500 printheads.

8.3.2 Max. Vertical Display Size and Resolution

Objectives:
Identify max. vertical display size and vertical resolution.

Test Patterns:
Diagonal lines using all even (odd) nozzles of a specific nozzle array with different slopes.

Results:

1. **Max. vertical display size:**

The vertical display size in which geometric objects are clearly identifiable is directly related to the vertical drop velocity. As more drops are ejected of a specific nozzle in direct succession, the average vertical drop velocity underneath the respective nozzle increases.
Cause: Air friction and inertia of surrounding air influencing drop trajectory.

A vertical display size of approx. 3 cm was reached by using a single nozzle array displaying a diagonal line with a slope of about 50 degrees. One frame consists of 3.8 Mio samples. 580 DTBs are generated per frame. One voxel consists of 8 drops ejected in direct succession (=8 drop layers or DTBs). 580 DTBs divided by 8 DTBs leading to a resolution of about 72 voxels per 3 cm.

Listing of the corresponding diagonal line image pattern source code, part of Signal Generator file `main.cpp`:

```
    int noz=0;
    int noz_max=24*16;
    int dtb=0;
    int dtb_shift=0;
    double slope=30*16/noz_max;

    for (; noz<noz_max; noz++) {
        dtb_shift=0;
        dtb=100+static_cast<int>(slope*noz);

        for (;dtb_shift<8;dtb_shift++) {
            sigGen.addActiveNozzle(dtb+dtb_shift,0,noz*2);
            sigGen.addActiveNozzle(dtb+dtb_shift,0,noz*2+1);
        }
    }
```

2. **Max. vertical resolution:**

The vertical resolution is reciprocally proportional to vertical drop velocity. Besides that it’s highly dependent on a number of other factors as described in section 3.5.

Due to experiment conditions under normal air pressure, experiments were focused towards an acceptable vertical display size.

8.3.3 Tests with Multiple Nozzle Arrays

Objectives:

- Test synchrony of multiple nozzle arrays.
- Number of max. simultaneously ejectable drops.
- Dependency of synchronously ejected drop count to drop ejection velocity.

Test Patterns:

Horizontal and diagonal line patterns using all nozzles of a single nozzle array were sent to multiple nozzle arrays simultaneously.

Results:

1. Synchrony:

As expected, multiple nozzle arrays can eject an identical drop ejection sequence simultaneously.

2. Max. simultaneously ejectable drops:

No limitations were found in the number of simultaneously ejectable drops when all nozzles of multiple nozzle arrays were activated simultaneously. The highest number of synchronously activated nozzle arrays was 8.

However, further investigations should be made in the future concerning the size of occurring electrical current peaks and their potential impact on printhead lifetime in case all printhead nozzles are active simultaneously.

Drop ejection velocity:

Based on picture analysis, the estimated drop ejection velocity is about 8 m/s for the given parameters.

No exact drop ejection velocity measurements were made in order to investigate a potential dependency between the number of simultaneously ejected drops and drop ejection velocity. Based on a pure visual impression, no difference was recognizable.

3. Estimation of max. ink output

The volume of an ejected drop is based on vendor’s data (see Table 2) 2 pico liters. In order to calculate the max. ink output per hour of operation, it is assumed that 8x768 nozzles are active per DTB cycle. In combination with a drop ejection frequency (=1/DTB_cycle_length) of 19.6 kHz (see Table 15, f = 1/(2120 ns * 24 DW)) this results in an overall ink ejection volume of 0,87 liters per hour.

For the actual experiment conditions it’s assumed that less than 5% of this volume will be required, since the estimated number of nozzles which will be simultaneously active in average is far lower, considering the content to be displayed.

4. Problems with specific nozzle arrays:

When using multiple nozzle arrays simultaneously together with the red nozzle array, drop ejection problems were recognized with the red nozzle array. The standalone operation of the red nozzle array however was nominal. The cause for this needs to be further investigated.

8.3.4 Output of Basic Image Patterns

Objectives:
Test optical recognizability of simple image patterns.

Test Patterns:
Basic output patterns are generated with the help of a MATLAB script. See section 8.4 for details.

- The letters "TUM" are generated as output pattern.
- An image formed of simple geometric objects is created.

The output patterns are displayed via a single nozzle array and also with increased visual depth by using multiple nozzle arrays.

Results:
"TUM" writing input pattern:

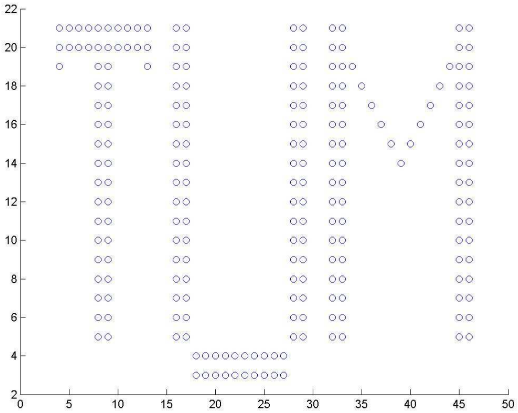


Fig. 45 Input pattern: `tum-disp-pattern-short14-48x25-wide-T-U-optim5.mat`

Fig. 45 shows the `low_res_pattern` MATLAB input pattern with a size of 48x25 pixels.

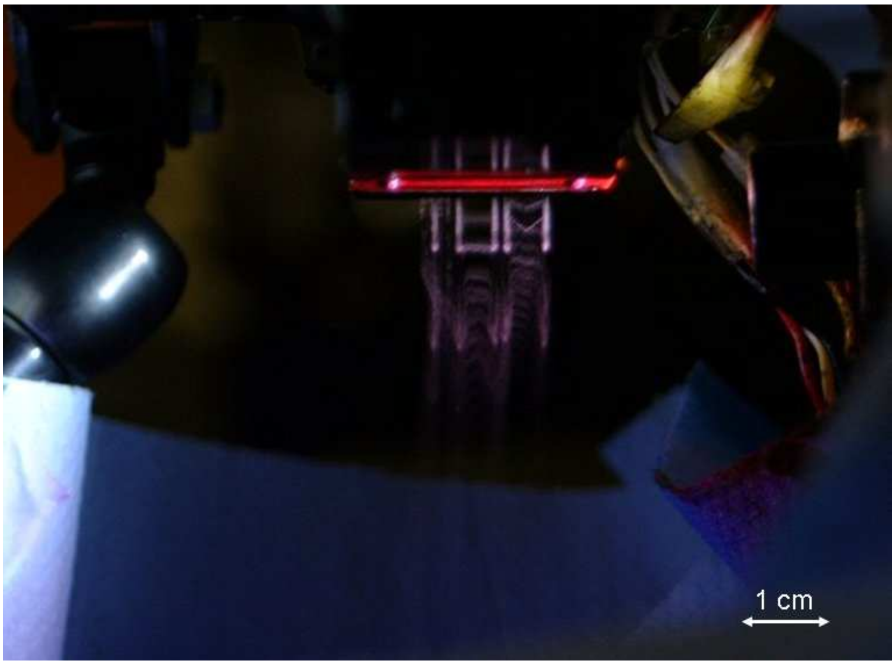


Fig. 46 Output pattern TUM 384x25x2

Fig. 46 shows the corresponding input pattern scaled to a size of 384 voxels horizontally, 25 voxels vertically and 2 voxels perpendicular to the viewing plane (= Z-axis).

One frame consists of 800,000 samples resulting in an image refresh frequency of 156.25 Hz. Trigger delay for TIG7000 is set to 900 µs. The pattern is displayed by using one nozzle array. Since one nozzle array consists of two opposing nozzle rows (even and odd nozzles, see Fig. 37), two layers in Z-direction can be displayed with one nozzle array.

Picture explanation: The Xenon flash light source is mounted at the right side of the picture. The printhead carriage assembly is mounted at the top with the printhead nozzle plate facing downwards. The red light at the nozzle plate is caused by light reflection from red ink particles brought there by previous test series.

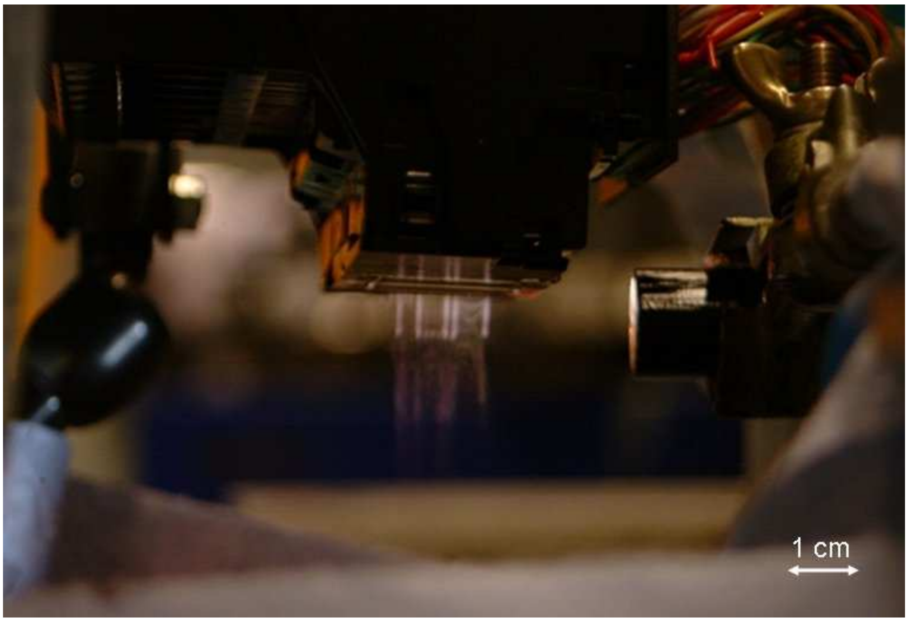


Fig. 47 Output pattern TUM 384x25x2, daylight conditions

Fig. 47 shows same output pattern as Fig. 46 under “daylight” conditions.

Fig. 48 Output pattern TUM 384x25x6, 45 degree

Fig. 48 shows same output pattern as Fig. 46, however visual depth is increased by ejecting the identical drop pattern simultaneously via 3 adjacent nozzle arrays (nozzle array 3, 4 and 5; see Fig. 20). Viewing angle rotated by 45 degrees.

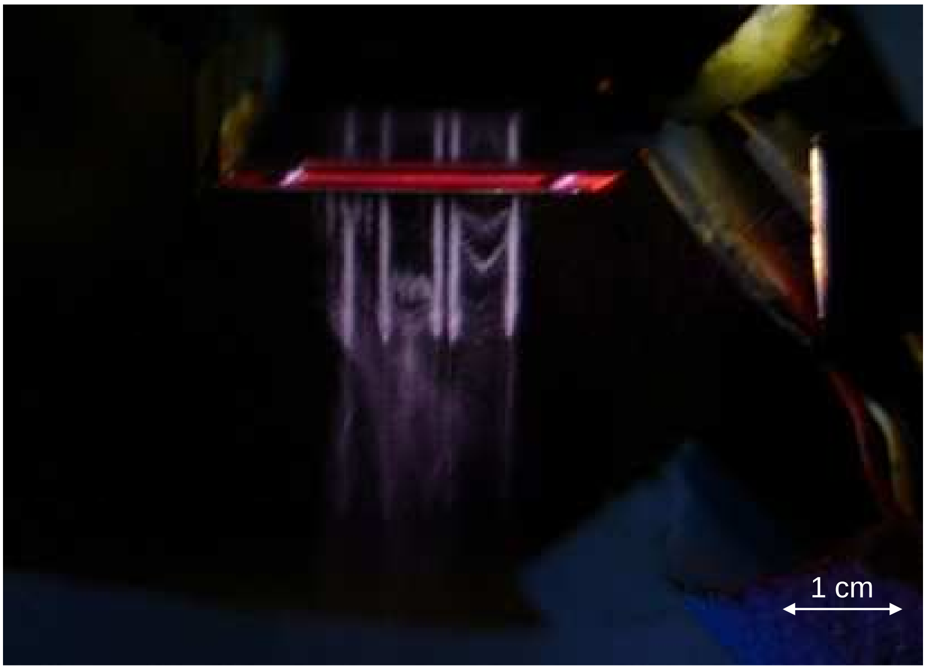


Fig. 49 Output pattern TUM 384x75x4

Fig. 49 shows input pattern Fig. 45, however vertical pixel count is scaled by factor 3 to 75 voxels, output pattern is displayed by two nozzle arrays.

The strong impact of air-resistance becomes visible in areas with relatively low overall drop ejection frequency and large distance to the nozzle plate (lower dash of letter "U").

Input pattern with simple geometric objects:

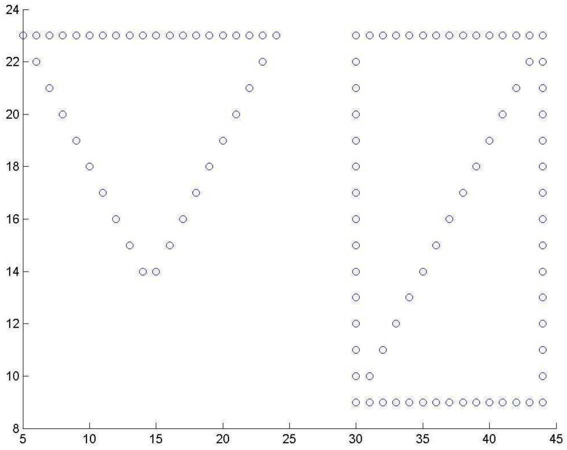


Fig. 50 Input pattern: `geom1-48x25.mat`

Fig. 51 Output pattern geom1 384x25x2

Fig. 51 shows the corresponding input pattern scaled to a size of 384 voxels horizontally, 25 voxels vertically and 2 voxels perpendicular to the viewing plane. One frame consists of 800,000 samples.

Input pattern with square geometry:

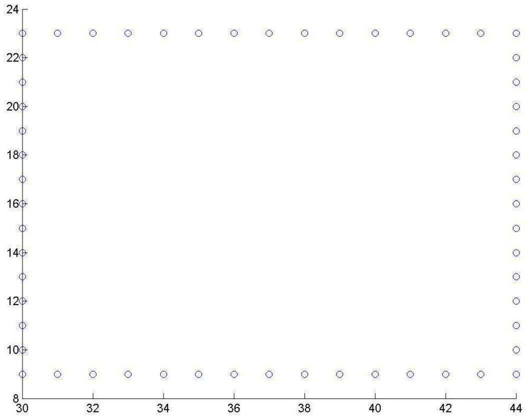


Fig. 52 Input pattern: `square-48x25.mat`

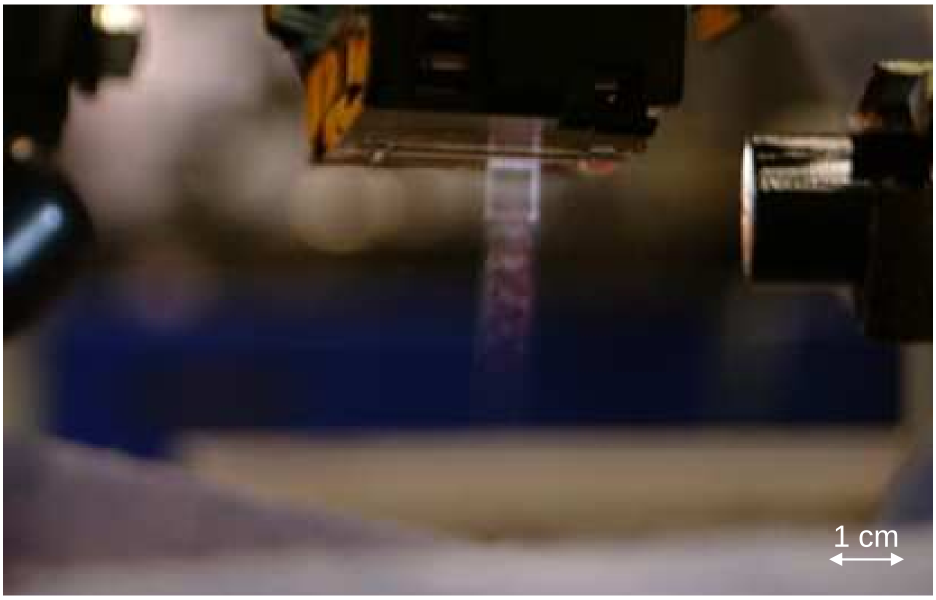


Fig. 53 Output pattern square 384x25x2

Fig. 53 shows the corresponding input pattern scaled to a size of 384 voxels horizontally, 25 voxels vertically and 2 voxels perpendicular to the viewing plane. One frame consists of 400,000 samples leading to an image refresh frequency of 312.5 Hz.

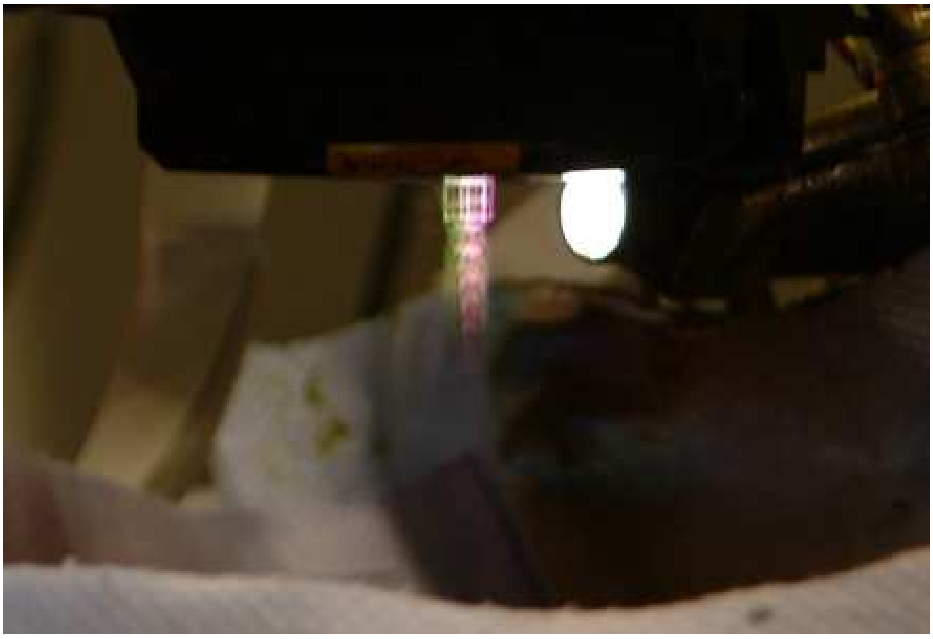


Fig. 54 Output pattern square tunnel 384x25x8

Fig. 54 shows the square pattern as in Fig. 53, but simultaneously ejected via 4 nozzle arrays, leading to a visual depth of 8 voxels. The nozzle arrays 1, 3, 4 and 5 are used.

8.4 Generation of Basic Volumetric Output Patterns

This section gives step by step instructions for creating static extruded 2D volumetric output patterns like the letters "TUM". Less complex patterns like single horizontal, vertical or diagonal lines can also be created directly by adding a few lines of C++ code to the Signal Generator source code.

Goal:
Generate "TUM" volumetric image output as shown in Fig. 46.

Security advice:
Check section 8.2 before starting experimentations.

Proceeding:

1. Create a matrix in MATLAB by name `low_res_pattern` with the size 25 rows by 48 columns filled with zeros. Background Info: The number of rows is selected arbitrarily, the number of columns is set to 48 since a scale factor of 16*48=768, allowing all nozzles of a nozzle array to be activated.

2. Edit the matrix with the MATLAB Array Editor by double clicking on the variable's name in the Workspace. Put a "1" in all cells where a voxel should be placed, all other cells should be "0".

3. Edit the MATLAB `canon_2d_pattern_generator.m` script (see section C.5) and adjust horizontal and vertical scaling factors as desired.

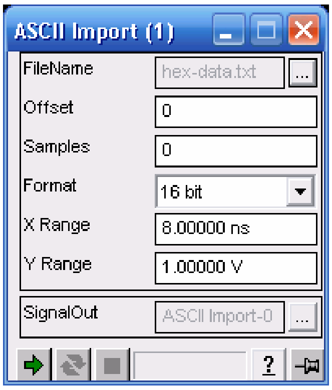
4. Execute the `canon_2d_pattern_generator.m` MATLAB script. A file `voxel-commands.txt` is created in the same folder. The file contains C++ function calls to be placed in the Signal Generator source code.

5. Copy all lines of the `voxel-commands.txt` file into the `main.cpp` source code file of the Signal Generator software, see section 7 for reference.

6. Re-compile the Signal Generator software.

7. Execute the Signal Generator software on the Windows Command Prompt. Store the hexadecimal signal output by using the command line flag "`-o[filename]`" (attention: no space between the "o" and the filename is allowed!).

8. Import the resulting hexadecimal signal file via SBench's ASCII import function into SBench. See Fig. 55 for import settings.

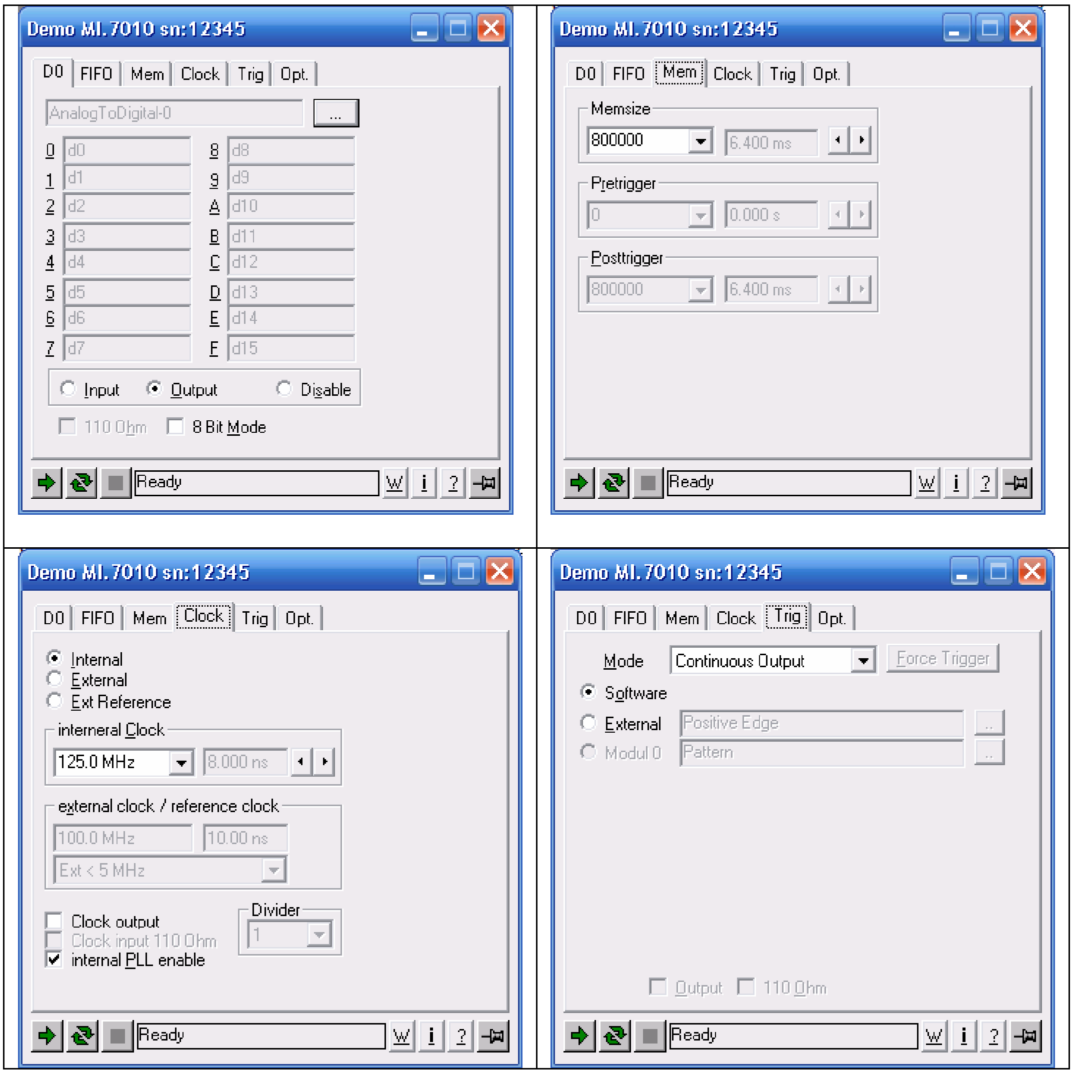


FileName	hex-data.txt
Offset	0
Samples	0
Format	16 bit
X Range	8.00000 ns
Y Range	1.00000 V
SignalOut	ASCII Import-0

Fig. 55 ASCII import settings for hex signal data with 125 MHz sampling rate

9. Convert imported signal data to digital data set with SBench’s function “Format Convert->AnalogToDigital”.

10. Configure SBench for signal output by using its hardware configuration dialog box (see Fig. 56). “TUM” example output parameters:
    - 800,000 samples per frame (adjustable via “Memsize”)
    - Sample rate of 125 MHz
    - “Continuous Output” triggering



AnalogToDigital-0

0 d0, 1 d1, 2 d2, 3 d3, 4 d4, 5 d5, 6 d6, 7 d7, 8 d8, 9 d9, A d10, B d11, C d12, D d13, E d14, F d15

○ Input ● Output ○ Disable

☐ 110 Ohm ☐ 8 Bit Mode

Memsize: 800000, 6.400 ms

Pretrigger: 0, 0.000 s

Posttrigger: 800000, 6.400 ms

● Internal ○ External ○ Ext Reference

interneral Clock: 125.0 MHz, 8.000 ns

external clock / reference clock: 100.0 MHz, 10.00 ns, Ext < 5 MHz

☐ Clock output
☐ Clock input 110 Ohm
☑ internal PLL enable

Divider: 1

Mode: Continuous Output — Force Trigger

● Software
○ External: Positive Edge
○ Modul 0: Pattern

☐ Output ☐ 110 Ohm

Fig. 56 SBench 5.2 signal output configuration

11. Connect the digital I/O board with the wire-tapping board. Establish all required electrical connections on the wire-tapping board between digital I/O board and printhead. See appendix B and section 7.2 for pin assignments.

12. Connect the TIG7000 trigger input line to line 15 of digital I/O board.

13. Switch-on the printer. Self-test of printer should finish successfully.

14. Press SBench’s play signal button to start generating volumetric output patterns. Hint: Visual depth of the output pattern can be increased by ejecting it via several adjacent nozzle arrays.

15. Stopping output pattern generation. **WARNING: NEVER** press the signal play button in SBench a second time to stop image generation, **while the printhead gets heating power**, otherwise nozzle arrays can be seriously damaged. The assumed cause for this behavior is that not all signal lines of Spectrum I/O card are set immediately to 0V level, when signal playing should be stopped.

    Safe way to stop output pattern generation: Firstly pull the printer’s power plug or wait till the printer controller board automatically drops heating power to the printhead due to printer controller stand-by mode. Secondly press SBench’s stop button to stop nozzle control signal generation.

9. Resume and Outlook

The experimentation results show that the principle of drop generation and short time illumination of the falling drops is basically suitable for generating a volumetric display effect.
Beyond that, it was proven that market available thermal based inkjet printheads with a high nozzle density are capable of producing images showing such a volumetric display effect.

Under the given experimentation conditions, air-friction played - as assumed - a critical role in the deceleration of the ejected drops with the consequence that the vertical display space is very limited.

Future research work on this topic should tackle the following issues next:

- **Air-friction**
  Analyze the impact of reduced gas-pressure on the drop ejection process, the trajectory of ejected drops and drop trajectory interaction by creating simulation models and execution of experiments.

- **Optimal drop size**
  Investigate the optimal drop size in relation to volumetric display size, (vertical) resolution and image refresh rate under consideration of technical and physical boundary conditions.

The research on this basic volumetric image generation principle showed further that it provides numerous points of attack for parameter optimization and modification. In an "inverse" occurrence of this basic principle, for instance gas bubbles could be ejected into a denser fluid.

10. References

[1] D. Bahr, K. Langhans, D. Bezecny, D. Homann, and Carsten Vogt, "FELIX: a volumetric 3D imaging technique", *New Image Processing Techniques and Applications*, Proceedings of SPIE, Volume 3101, pp. 202-210, Munich, June 1997.

[2] S. Bains, "Radial scanning produces 3-D image on a flat screen", *Laser Focus World*, pp. 41-42, January 1993.

[3] R. Balakrishnan, G. W. Fitzmaurice, G. Kurtenbach, "User Interfaces for Volumetric Displays", *IEEE Computer*, Vol. 34, No. 3, 2001.

[4] B. G. Blundell, A. J. Schwarz, and D.K. Horrell, "Volumetric three-dimensional display systems: their past, present and future", *Engineering Science and Education Journal*, pp. 196-200, Oct. 1993.

[5] U. Brinkmann, "Lichtgrafik im Dreidimensionalen", *Laser und Optoelektronik*, Vol.15, pp. 41-42, March 1983.

[6] T. F. Budinger, "An analysis of 3-D display strategies", *Processing and Display of Three-Dimensional Data II*, Proceedings of SPIE, Vol. 507, pp. 2-8, 1984.

[7] T. S. Buzak, "A Field-Sequential Discrete-Depth-Plane Three-Dimensional Display", Society for Information Display, *SID 85 Digest*, pp. 345-347, 1985.

[8] E. Downing, L. Hesselink, J. Ralston, and R. Macfarlane, "A Three-Color, Solid-State, Three-Dimensional Display", *Science*, Vol. 273, pp. 1185-1189, August 1996.

[9] J. Fajans, "Three-dimensional display", *Advances in Display Technology*, Proceedings of SPIE, Vol. 199, pp. 23-28, 1979.

[10] G. Favalora, R. K. Dorval, D. M. Hall, M. Giovinco, J. Napoli "Volumetric three-dimensional display system with rasterization hardware", *Stereoscopic Displays and Virtual Reality Systems VIII*, Proceedings of SPIE, Vol. 4297A, San Jose, CA, 2001.

[11] C. Hand, "A Survey of 3-D Input Devices”, *Technical Report CS TR94/2*, De Montfort University, 1994.

[12] D. G. Jansson, E. P. Berlin et. al., "A Three-Dimensional Computer Display", *Computer Graphics in CAD/CAM Systems*, Annual Conference, Cambridge, April 1979.

[13] R. D. Ketchpel, "Direct-View Three-Dimensional Display Tube", *IEEE Transactions on Electron Devices*, pp. 324-328, September 1963.

[14] I. I. Kim, E. Korevaar, and H. Hakakha, "Three-dimensional volumetric display in rubidium vapor", *Projection Displays II*, Proceedings of SPIE, Vol. 2650, pp. 274-284, San Jose, CA, 1996.

[15] K. Langhans and D. Bahr, "Science Education in Action: Germany's FELIX Project", *Journal of Laser Applications*, Vol. 8, No. 5, pp. 221-224, October 1996.

[16] K. Langhans, D. Bezecny, D. Homann, D. Bahr, C. Vogt, C. Blohm, and K.-H. Scharschmidt "New portable FELIX 3D display", *Projection Displays IV*, Proceedings of SPIE, Vol. 3296, San Jose, CA, 1998.

[17] M. Lasher, P. Soltan, W. Dahlke, N. Acantilado, and M. McDonald, "Laser Projected 3-D Volumetric Displays", *Projection Displays II*, Proceedings of SPIE, Vol. 2650, pp. 285-295, San Jose, CA, 1996.

[18] J. D. Lewis, C. M. Verber, and R. B. McGhee, "A True Three-Dimensional Display", *IEEE Transaction on Electron Devices*, Vol. ED-18, No. 9, pp. 724-732, September 1971.

[19] K.-D. Linsmeier, "Raumbilder auf dreidimensionalem Monitor", *Spektrum der Wissenschaft*, pp. 24-25, January 1996.

[20] D. L. Macfarlane et. al., "Voxel-based spatial display", *Stereoscopic Displays and Virtual Reality Systems*, Proceedings of SPIE, Vol. 2177, pp. 196-202, 1994.

[21] D. F. McAllister (Ed.), *Stereo Computer Graphics and Other True 3D Technologies*, pp. 230-246, Princeton University Press, Princeton, New Jersey, 1993.

[22] R. L. de Montebello, "Optical Dissector", U.S. Patent No. 3,428,393, 1969.

[23] N.N., "CRT Provides Three-Dimensional Displays", *Electronics*, November 1962.

[24] N.N., "3-D display", *Space/Aeronautics*, pp. 60-67, Sept. 1962.

[25] N.N., "Flight Patterns Displayed in 3-D", *Aviation Week*, p. 85, May 30th, 1960.

[26] E.Parker, P.R. Wallis, "Three-Dimesional Cathode-Ray Tube Displays", *The Journal of the Institution of Electrical Engineers*, Vol. 95, Part III, pp. 371-390, London 1948.

[27] E. G. Rawson, "Vibrating varifocal mirrors for 3-D imaging", *IEEE Spectrum*, pp. 37-43, September 1969.

[28] A. J. Schwarz, and B. G. Blundell, "Regions of Extreme Image Distortion in Rotating-Screen Volumetric Display Systems", *Computer & Graphics*, Vol. 18, No. 5, pp. 643-652, 1994.

[29] S. Shimada, "A New Approach to the Real-Image 3D Globe Display", Society for Information Display, *SID 93 Digest*, pp. 1001-1004, 1993.

[30] P. Soltan, J. Tiras, W. Dahlke, M. Lasher, M. McDonald, "Laser-Based 3-D Volumetric Display System (The Improved 2nd Generation)", NRaD, San Diego, CA, Jan. 1995

[31] D. Solomon, "Volumetric Imaging Launches Graphics Into A 3-D World", *Photonics Spectra*, pp. 129-134, June 1993.

[32] C. Tsao and J. Chen, "Moving Screen Projection: a new approach for volumetric three-dimensional imaging", *Projection Displays II*, Proceedings of SPIE, Vol. 2650, pp. 254-261, San Jose, CA, 1996.

[33] R. D. Williams and D. Donohoo, "Image Quality Metrics for Volumetric Laser Displays", *Electronic Imaging*, Proceedings of SPIE, Vol. 1457, 1991.

[34] E. L. Withey, "Cathode-Ray Tube Adds Third Dimension", *Electronics engineering edition*, pp. 81-83, May 1958.

[35] H. Yamada, C. Masuda, K. Kubo, T. Ohira, K. Miyaji, "A 3-D Display Using a Laser and a Moving Screen", *Japan Display*, pp. 416-419, 1986.

[36] Knut Langhans, Detlef Bahra, Daniel Bezecnya, Dennis Homanna, Klaas Oltmanna, Krischan Oltmanna, Christian Guilla, Elisabeth Riepera, Götz Ardeyb, “FELIX 3D Display: An Interactive Tool for Volumetric Imaging”, Proceedings of SPIE, Vol. 4660, "*Stereoscopic Displays and Virtual Reality Systems IX*", http://www.felix3d.com/paper_pw_02.pdf

[37] B.G. Blundell and A. J. Schwarz, “Volumetric Three-Dimensional Display Systems”, p. 4, John Wiley & Sons, New York, 2000

[38] N.N., “Stereoscopy”, http://en.wikipedia.org/wiki/Stereoscopic

[39] Won-Suk Chun et. al., “Spatial 3-D Infrastructure: Display-Independent Software Framework, High-Speed Rendering Electronics, and Several New Displays”, http://www.actuality-systems.com/site/content/pdf/Actuality_SPIE_2005_Chun_et_al.pdf

[40] Gregg E. Favalora et. al., “100 Million-voxel volumetric display”, SPIE2002, Cockpit Displays IX

[41] N.N., “Canon PIXMA ip8500”, http://www.druckerchannel.de/artikel.php?ID=809&seite=2

[42] N.N., “Canon Deutschland”, http://www.canon.de

[43] N.N., “Canon PIXMA Technology Guide 2004 English Version 1.1”, Inkjet Printer Products Creative Communication Div

[44] N.N., “Canon PIXMA ip2000 specifications”, http://consumer.usa.canon.com/ir/controller?act=ModelTechSpecsAct&fcategoryid=117&modelid=10237

[45] K. Langhans, C. Guill, E. Rieper, K. Oltmann, D. Bahr, “SOLID FELIX A Static Volume 3D-Laser Display”, *Stereoscopic Displays and Applications XIV*, Proceedings of SPIE, Vol. 5006, 2003

[46] N.N., “National Institute of Advanced Industrial Science and Technology (AIST)”, http://www.aist.go.jp/aist_e/latest_research/2006/20060210/20060210.html

[47] N.N., “Depth perception”, http://en.wikipedia.org/wiki/Depth_perception

[48] M. Halle, “Autostereoscopic displays and computer graphics”, *Computer Graphics*, ACM SIGGRAPH, 31(2), May 1997, pp 58-62

[49] Courian et. al., “Reliable high performance drop generator for an inkjet printhead”, *U.S. Patent No. 5,946,012*, 1999

[50] Michael Schwingen, “TINTENSTRAHL DIGITAL”, Elektor Magazin 4/92, pp. 32-33, 1992

[51] N.N., “HP Linux Inkjet Driver Project”, http://hpinkjet.sourceforge.net/

[52] N.N., “Trolltech Qt C++ Toolkit”, http://www.trolltech.com/products/qt/index.html

[53] N.N., “Least Squares”, http://en.wikipedia.org/wiki/Least_squares

[54] N.N., “Umbrello UML Modeller”, http://uml.sourceforge.net/index.php

A. Canon ip2000 Pin Assignments

A.1 Pin Assignment of ip2000 Printer Controller Board Connector J201 (C1-)

C1-Pin	PH-Pin	Line Name	Description / signal purpose	Signal level	Status			Knowledge level	Refer-ence data
					RI	RB	RC		
1	5,6,15,16,29,36,opto3		common ground	0V	x	x	x	high	
2			unused					med	
3	37		mass for printhead initialization	TTL	x			low	
4			same as C1-1		x	x	x	high	
5	24	c_dataline2	dataline2 cyan nozzle, see also C1-6	TTL			x	high	
6	23	c_dataline1	dataline1 cyan nozzle, see also C1-5	TTL			x	high	
7	39	b_fireline1	fireline for all black nozzles	TTL		x		high	
8	14	cmy_fireline2	fireline for cyan, magenta and yellow nozzle group 2&4, see also C1-15	TTL			x	high	
9	13	reset	printhead reset /synchronization signal	TTL		x	x	high	
10	38	b_dataline2	dataline2 black nozzle, see also C1-16	TTL		x		high	
11	21	c1_11_init	related to printhead identification by controller board, see also C1-12, C1-13	TTL	x			med	
12	28	c1_12_init	related to printhead identification by controller board, see also C1-11, C1-13	TTL	x			med	
13	27	c1_13_init	related to printhead identification by controller	TTL	x			med	

			board, see also C1-11, C1-12						
14	11	y_dataline2	dataline2 yellow nozzle, see also C1-18	TTL			x	high	
15	18	cmy_fireline1	fireline for cyan, magenta, yellow nozzle group 1&3, see also C1-8	TTL			x	high	
16	33	b_dataline1	dataline1 black nozzle, see also C1-10	TTL		x		high	
17	12	clock	clock signal for data transfer to printhead	TTL		x	x	high	
18	4	y_dataline1	dataline1 yellow nozzle, see also C1-14	TTL			x	high	
19	10	m_dataline2	dataline2 magenta nozzle, see also C1-20	TTL			x	high	
20	9	m_dataline1	dataline1 magenta nozzle, see also C1-19	TTL			x	high	

Legend:
TTL = 3.3 V TTL signal

RC	required for color printing
RB	required for black printing
RI	required for initialization

A.2 Pin Assignment of ip2000 Printer Controller Board Connector J202 (C2-)

C2-Pin	PH-Pin	Line Name	Description / signal purpose	Signal level	Status			Knowledge level	Refer-ence data
					RI	RB	RC		
1	-		pin 2 of horizontal incremental position sensor, see also C2-3	TTL				low	
2	-		power supply for horizontal incremental position sensor LED	3,3V				high	
3	-		pin 1 of horizontal incremental position sensor, see also C2-1	TTL				low	
4			same as C1-1			x	x	med	
5			same as C1-1			x	x	med	
6			same as C1-1			x	x	med	
7	17		heater element power supply	+24V DC		x	x	high	
8	25,26		heater element power supply	+24V DC		x	x	high	
9			same as C2-8			x	x	high	
10			same as C2-8			x	x	high	
11	1,2,3		assumed: heater power supply common ground	0V		x	x	low	
12			same as C2-11			x	x	low	
13			same as C2-11			x	x	low	
14	32,40		heater element power supply	+24V DC		x	x	high	
15			same as C2-14			x	x	high	
16			same as C2-14			x	x	high	
17			same as C2-14			x	x	high	
18	7,8		assumed: heater power supply common ground	0V		x	x	low	
19	19,20		assumed: heater power supply common ground	0V		x	x	low	
20			same as C2-18			x	x	low	

Legend:
TTL = 3.3 V TTL signal

RC	required for color printing
RB	required for black printing
RI	required for initialization

B. Canon ip8500 Pin Assignments

B.1 Pin Assignment of ip8500 Printhead Carriage PCB Connector JCR1 (C1-)

Attention: pin numbering on printer controller board side is reversed compared to printhead carriage PCB!

C1-Pin	PH-Pin	Line Name	Description / signal purpose	Signal level	Status			Knowledge level	Reference data
					RI	RB	RC		
1	10	r_dataline1	dataline1 red nozzle array, see also C1-8	TTL			X	high	
2	9	b_dataline2	dataline2 black nozzle array, see also C1-3	TTL		X		high	
3	8	b_dataline1	dataline1 black nozzle array, see also C1-2	TTL		X		high	
4	43	r_g_fireline	fireline for red and green nozzle array	TTL			X	high	
5	42	pm_fireline	fireline for photo magenta nozzle array	TTL			X	high	
6	25	pm_dataline1	dataline1 photo magenta nozzle array, see also C1-13	TTL			X	high	
7	41	b_pc_fireline	fireline for black and photo cyan nozzle array	TTL		X	X	high	
8	24	r_dataline2	dataline2 red nozzle array, see also C1-1	TTL			X	high	
9	39	g_dataline2	dataline2 green nozzle array, see also C1-12	TTL			X	med	
10	38	pc_dataline2	dataline2 photo cyan nozzle array, see also C1-11	TTL			X	high	
11	23	pc_dataline1	dataline1 photo cyan nozzle array, see also C1-10	TTL			X	high	
12	65	g_dataline1	dataline1 green nozzle array, see also C1-9	TTL			X	med	

13	51	pm_dataline2	dataline2 photo magenta nozzle array, see also C1-6	TTL			X	high	
14	37	c_dataline1	dataline1 cyan nozzle array, see also C1-16	TTL			X	high	
15	50	m_dataline2	dataline2 magenta nozzle array, see also C1-17	TTL			X	med	
16	22	c_dataline2	dataline2 cyan nozzle array, see also C1-14	TTL			X	high	
17	7	m_dataline1	dataline1 magenta nozzle array, see also C1-15	TTL			X	med	
18	49	y_dataline2	dataline2 yellow nozzle array, see also C1-23	TTL			X	high	
19	36		dataline for nozzle array 9 or 10?	TTL			X	low	
20	48		dataline for nozzle array 9 or 10?	TTL			X	low	
21	6		dataline for nozzle array 9 or 10?	TTL			X	low	
22	21		dataline for nozzle array 9 or 10?	TTL			X	low	
23	35	y_dataline1	dataline1 yellow nozzle array, see also C1-18	TTL			X	high	
24	33	c_m_fireline	fireline for cyan and magenta nozzle array	TTL			X	high	
25	32	y_fireline	fireline for yellow nozzle array	TTL			X	high	
26	31		fireline for nozzle array 9 and 10?	TTL				low	
27	11,26		ENC01 pin2, ENC01 pin4, mass for printhead initialization	0V	X			med	
28	34	reset	printhead reset /synchronization signal	TTL		X	X	high	
29	-		same as C1-27		X			low	
30	-		unknown		X			low	
31	5	clock	clock signal for data transfer to printhead	TTL		X	X	high	
32	64		related to printhead initialization/identification	TTL	X			low	
33	66,53		unknown					very low	
34	-		ENC01 pin3 of horizontal incremental position sensor on printhead carriage PCB		X			low	
35	61		related to printhead initialization/identification	TTL	X			low	

36	63		related to printhead initialization/identification	TTL	X			low	
37	?		unknown					very low	
38	-		JSNS1 pin1, light barrier connector on printhead carriage PCB		X			low	
39			ENC01 pin5		X			low	
40			same as C1-27, JSNS1 pin2		X			low	

Legend:
TTL = 3.3 V TTL signal

RC	required for color printing
RB	required for black printing
RI	required for initialization

B.2 Pin Assignment of ip8500 Printhead Carriage PCB Connector JCR2 (C2-)

C2-Pin	PH-Pin	Line Name	Description / signal purpose	Signal level	Status			Knowledge level	Reference data
					RI	RB	RC		
1	12,13, 14,27, 28,29		assumed: heater power supply common ground	0V				med	
2			same as C2-1						
3			same as C2-1						
4			same as C2-1						
5			same as C2-1						
6			same as C2-1						
7			same as C2-1						
8	40		unknown	0V				very low	
9			not used? / unknown					very low	
10			not used? / unknown					very low	
11	11,26		common ground	0V				high	
12	67		20,69V DC from printer controller board required for printing	+20,69V DC		X	X	low	
13	54,55, 56,68, 69		heater element power supply	+23,2V DC		X	X	high	
14			same as C2-13						
15			same as C2-13						
16			same as C2-13						
17			same as C2-13						
18			same as C2-13						
19			same as C2-13						
20			same as C2-13						

Legend:

TTL = 3.3 V TTL signal

RC	required for color printing
RB	required for black printing
RI	required for initialization

B.3 Pin Assignment of ip8500 Printhead Carriage PCB Connector JCR3 (C3-)

C3- Pin	**PH- Pin**	**Line Name**	**Description / signal purpose**	**Signal level**	**Status**			**Knowledge level**	**Reference data**
					RI	**RB**	**RC**		
1	44,45, 46,57, 58,59		assumed: heater power supply common ground	0V		X	X	med	
2			same as C3-1						
3			same as C3-1						
4			same as C3-1						
5			same as C3-1						
6			same as C3-1						
7			same as C3-1						
8			same as C3-1						
9	53, 66		unknown	0V				very low	
10	20		unknown	0V	X			very low	
11	60		unknown	0V				very low	
12	52		unknown		X			very low	
13	11,26		common ground, JSNS1 pin2		X			high	
14	2,3,4, 17,18, 19		assumed: heater power supply common ground	0V		X	X	med	
15			same as C3-14						
16			same as C3-14						
17			same as C3-14						
18			same as C3-14						
19			same as C3-14						
20			same as C3-14						

Legend:
TTL = 3.3 V TTL signal

RC	required for color printing
RB	required for black printing
RI	required for initialization

C. MATLAB Scripts

C.1 canon_FINE_bus_analyzer.m

```
% Canon FINE Printhead Bus Data Analyser
%
% Reads the export file of SBench 5.2, where the samples of
% the digital I/O card are stored as 16bit ASCII Hex ('0xfff').
% One sample per line.
% Converts provided samples into a binary representation of datawords.
% The script can convert only the two datalines of a single color
% per time.
% A single dataword consists out of e.g. 24 raw bits.
% Results are stored in data1, data2, data1_bit_pattern_sum,
%                          data2_bit_pattern_sum variables
%
% Copyright 2006, Erwin Roth, erwin.roth@weihenstephan.org.
% 2006-01-14

%% ***************************** Constants ****************************
% Input file (path + filename.ext)
inputfile = 'c:\signal-generator\test3-TUM-light2.txt';

% define bit position of signal data lines to analyse
% ATTENTION: MATLAB starts with bit 1 and not 0!!!
% Bit 1 = least significant bit
line_clock = 1;
line_reset = 2;
line_data1 = 3;     %dataline1 of a specific nozzle color
line_data2 = 4;     %dataline2 of a specific nozzle color

% Bits included in one dataword (=DW)
bits_per_dw = 24;
% sample rate in MHz, necessary for correct signal analysis
sample_rate = 125;

% offset in samples, for correctly detecting bit status after a rising
% or falling edge. default = 1
bit_read_sample_offset = 1;

% all timings in nano seconds
% minimum timespan of reset signal low level
t_reset_min = 50;

%% -------------------- actual script code -----------------------------
% read file
inputfid = fopen(inputfile, 'r');

% initialize data array
samples = uint16([]);  % store sample data as 16 bit array

reset = [];         % vector holding end of reset signal sample indices

dw_bit_idx = [];           % matrix holding dataword bit sample indices

data1 = logical([]);         % output data for dataline1
data2 = logical([]);         % output data for dataline2

% read input file contents
line_count = inf;        % number of lines or inf for end of file
samples = fscanf(inputfid, '0x%4x\n',line_count);
fclose(inputfid);
clear tline;                    % delete unnecessary variable
samples_cnt = length(samples);     % store number of overall samples

%% calculate values required for data analysis
tdif = 1/sample_rate*1e+3;      % time per sample in ns
min_reset_samples = ceil(t_reset_min/tdif);

%% detect and store beginning of new datawords by finding falling and
```

```
%% rising edge of reset signal
idx = 2;

old_value = uint8([]);          % variables to store signal level
value = uint8([]);

while idx <= samples_cnt
    old_value = bitget(samples(idx-1), line_reset);
    value = bitget(samples(idx), line_reset);

    if old_value > value        % detect falling edge
        reset_start = idx;         % restart counter
        idx = idx + 1;
        continue;
    end

    % detect valid rising edge of reset signal
    if ((old_value < value) && (idx-reset_start>=min_reset_samples))
            reset(end+1) = idx;       % store end of reset sample index
    end
    idx = idx + 1;
end
% now we have a reset vector, containing all sample index numbers where a
% valid reset signal ends.

% a data bit is read on every rising and falling edge of the clock signal
% between two reset signals.
% Now work on all samples between two reset signals and identify the
% samples when a dataword bit is set

reset_cnt = length(reset);
if reset_cnt < 2
    error('Error: too less datawords identified');
    return;
end

reset_idx = 2;
while reset_idx <= reset_cnt
    idx = reset(reset_idx-1);
    stop_sample_idx = reset(reset_idx);

    bit_idx = 1;        % set bit index to first bit of dataword
    while idx < stop_sample_idx
        old_value = bitget(samples(idx-1), line_clock);
        value = bitget(samples(idx), line_clock);

        if old_value ~= value    % detect change (=rising and falling edge)
            dw_bit_idx(reset_idx-1,bit_idx) = idx+bit_read_sample_offset;
            bit_idx = bit_idx + 1;
            idx = idx + 1;
            continue;
        end
        idx = idx + 1;             % next sample
    end
    % check for faulty dataword due to clock signal error
    if bit_idx ~= (bits_per_dw+1)
        faulty_dw(end+1) = reset_idx-1;
        disp('Warning: faulty dataword detected!')
    end
    reset_idx = reset_idx + 1;
end
% now we have the dw_bit_idx matrix, which contains for all dataword bits
% the respective sample indices.

%% Scan datalines based on dw_bit_idx sample indices
dw_count = size(dw_bit_idx,1);

idx = 1;
bit_idx = 1;
```

```
while idx <= dw_count       % go through all datawords
    bit_idx = 1;
    % create array with logical values
    while bit_idx <= bits_per_dw    % go through all bits
        % scan dataline1
        % 1. index: dataword number
        % 2. index: bit number
        data1(idx, bit_idx) = bitget(samples(dw_bit_idx(idx,bit_idx)), ...
                                     line_data1);

        % scan dataline2
        data2(idx, bit_idx) = bitget(samples(dw_bit_idx(idx,bit_idx)), ...
                                     line_data2);

        bit_idx = bit_idx + 1;
    end
    % create decimal sum of bit pattern fields (19-23) for better
    % visual pattern recognition bit 23 is least significant bit!!
    bit_idx = 23;
    data1_bit_pattern_sum(idx) = 0;
    data2_bit_pattern_sum(idx) = 0;
    while bit_idx >= 19    % go through all bits
        data1_bit_pattern_sum(idx) = data1_bit_pattern_sum(idx) ...
            + (data1(idx, bit_idx))*2^-(bit_idx-23);
        data2_bit_pattern_sum(idx) = data2_bit_pattern_sum(idx) ...
            + (data2(idx, bit_idx))*2^-(bit_idx-23);
        bit_idx = bit_idx - 1;
    end

    idx = idx + 1;      % next dataword
end

%% Helpful commands for doing some manual data analysis. Copy commands as
%% needed to command line.

%display bit pattern with bit pattern sum
%disp(sprintf('%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d ...
      %d %d %d %d %d || %d\n', [data1, data1_bit_pattern_sum']'))

%find indices of DWs with set data bits
%test = find(sum(data1(:,1:18)'))

% compare bit pattern sequence of dataline1 and 2
%[data1_bit_pattern_sum; data2_bit_pattern_sum]'
```

C.2 canon_printhead_sim.m

```
% Canon ip8500 Printhead Simulator
% Aim of this script is to simulate the printhead output on paper in
% order to analyze and verify the correct dataword to dataset mapping.
%
% Asumptions of this function:
%   - bit pattern sequence on both datalines is null or synchronous
%   - both dataline matrices have same size
%   - One DTB reflects one horizontal printhead carriage step
%
% Limitations:
%   - the script supports currently only the NORM dataword sequence
%     (=direction)
%   - A carriage return of the printhead carriage is not supported,
%     all printhead output is displayed horizontally.
%
% Dependencies:
%   - the external function file calc_dw_pattern_sum.m is required
%
% Copyright 2006, Erwin Roth, erwin.roth@weihenstephan.org.


% Function returns value:
% for plotting the first two columns of plotdataX are most interesting:
% 1.col: x-value (datablock index)
% 2.col: y-value
function [plotdata1, plotdata2] = canon_printhead_sim(dataline1, dataline2)

%% Definition of Constants
DW_PER_DTB = 24;                        % datawords per datablock
DATALINE_BINARY_PATTERN_START_BIT = 19; % raw bit# of pattern begin
DATALINE_BINARY_PATTERN_BITS = 5;       % number of pattern bits
DATALINE_LSB = 17;                      % least significant data bit (nozzle
                                        % group 0 = closest to contact plate)

DATALINE_MSB = 2;                       % most significant data bit (highest
                                        % nozzle group)

% value measured via LSQ algorithm
DATALINE2_HORIZ_OFFSET = -6;            % horizontal offset of nozzles
                                        % controlled by line two compared
                                        % to dataline1

% required for dataword_sum calculation
DATALINE_PATTERN = [0,0,0,0,0;
                    1,0,0,0,1;
                    0,1,0,1,0;
                    1,1,0,0,0;
                    0,0,1,0,1;
                    1,0,1,1,0;
                    0,1,1,0,0;
                    1,1,1,0,1;
                    0,0,0,0,1;
                    1,0,0,1,0;
                    0,1,0,0,0;
                    1,1,0,0,1;
                    0,0,1,1,0;
                    1,0,1,0,0;
                    0,1,1,0,1;
                    1,1,1,1,0;
                    0,0,0,1,0;
                    1,0,0,0,0;
                    0,1,0,0,1;
                    1,1,0,1,0;
                    0,0,1,0,0;
                    1,0,1,0,1;
                    0,1,1,1,0;
                    1,1,1,0,0];
```

```
% IMPORTANT!!!
% vector to hold mapping of DWID bit patterns to nozzle set
% the nozzle set number controls the relative distance of the controlled
% nozzle to the contact plate within a nozzle group

%% Used initially to get data on which the Least Squares algorithm can be
%% applied
MAP_REL_DW_IDX_TO_NOZZLE_SET=[12,12,12,12,12,12,12,12,12,12,12,12,12,12,...
                              12,12,12,12,12,12,12,12,12,12];

%% **** Result of mapping after Least Squares algorithm was used to find
%% correct mapping ****
%MAP_REL_DW_IDX_TO_NOZZLE_SET=[24,0,14,21,4,11,18,1,8,15,22,5,12,19,...
% 2,9,16,23,6,13,20,3,10,17];

%calculate #nozzle groups
nozzle_grp_cnt = DATALINE_LSB+1-DATALINE_MSB;

% calculate easier comparable pattern sums
dataline_pattern_sum=calc_dw_pattern_sum(DATALINE_PATTERN);
dataline1_pattern_sum=calc_dw_pattern_sum(dataline1(:,...
    DATALINE_BINARY_PATTERN_START_BIT:DATALINE_BINARY_PATTERN_START_BIT+...
    DATALINE_BINARY_PATTERN_BITS-1));
dataline2_pattern_sum=calc_dw_pattern_sum(dataline2(:,...
    DATALINE_BINARY_PATTERN_START_BIT:DATALINE_BINARY_PATTERN_START_BIT+...
    DATALINE_BINARY_PATTERN_BITS-1));

% set datablock start identifier
dtb_start_pattern = dataline_pattern_sum(1,1);

% both matrices have same size
dw_cnt = size(dataline1,1);
dw_idx = 1;

while dw_idx <= dw_cnt
    % Search start of 1.DTB on dataline1
    if (dataline1_pattern_sum(dw_idx,1) ~= dtb_start_pattern)
        dw_idx = dw_idx+1;
        continue;
    end

    break;
end

if (dw_idx == dw_cnt)
    error('No DTB start found');
    return;
end

plotdata1 = [];  % initialize plotdata matrix
plotdata2 = [];  % initialize plotdata matrix

% init data
fill_vect = zeros(1,DW_PER_DTB);

dtb_idx = 0;
% now we are at 1. DW of 1. DTB
while dw_idx <= dw_cnt
    if ((dataline1_pattern_sum(dw_idx) == dtb_start_pattern) || ...
          (dataline2_pattern_sum(dw_idx) == dtb_start_pattern))
        % increase DTB index when we recognize beginning of new DTB
        dtb_idx = dtb_idx+1;
        rel_dw_idx = 1;
    end

    % skip creation of plot points for DWs with mapping zero
    % only necessary for easier data analysis
    if (MAP_REL_DW_IDX_TO_NOZZLE_SET(rel_dw_idx) == 0)
```

```
        dw_idx = dw_idx+1;
        rel_dw_idx = rel_dw_idx+1;
        continue;
    end

    for bit_idx=DATALINE_LSB:-1:DATALINE_MSB
        % asumption: dataline1 controls odd nozzles
        if (dataline1(dw_idx,bit_idx))
           % add point to plot data
           % calculate y-coord (factor 2, since we have two datalines)
           y_coor = (DATALINE_LSB-bit_idx)*DW_PER_DTB*2+...
               MAP_REL_DW_IDX_TO_NOZZLE_SET(rel_dw_idx)*2-2;
           % sets coord of point to be plotted
           plotdata1(end+1,:) = [dtb_idx, y_coor, fill_vect];
           % write y-values of different dataword id's in different
           % columns, for individual plotting
           plotdata1(end,rel_dw_idx+2)=y_coor;
        end
        % asumption: dataline2 controls even nozzles
        if (dataline2(dw_idx,bit_idx))
           % add point to plot data
           % calculate y-coord (factor 2, since we have two datalines)
           y_coor = (DATALINE_LSB-bit_idx)*DW_PER_DTB*2+...
               MAP_REL_DW_IDX_TO_NOZZLE_SET(rel_dw_idx)*2-1;
           plotdata2(end+1,:) = [dtb_idx+DATALINE2_HORIZ_OFFSET,...
                   y_coor, fill_vect];   % sets coord of point to be plotted

           % write y-values of different dataword id's in different columns
           % for individual plotting
           plotdata2(end,rel_dw_idx+2)=y_coor;
        end

    end
    dw_idx = dw_idx+1;
    rel_dw_idx = rel_dw_idx+1;
end

% Output control values
disp(sprintf('DTB Count: %d',dtb_idx));  % reflects horizontal deflection
disp(sprintf('Plot points: %d',length(plotdata1)));

% Plot simulated printing output
figure
hold on
title('Canon ip8500 Printing Simulation');
plot(plotdata1(:,1),plotdata1(:,2),'r.')
plot(plotdata2(:,1),plotdata2(:,2),'b.')
legend('dataline1', 'dataline2');
xlabel('x - axis');
ylabel('y - axis');

return;
```

C.3 calc_printhead_sim.m

```
% Canon FINE Printhead analysis
% Function to generate a better manageable dataword identification, by
% multiplying each raw pattern bit of a dataword by 2^rel_bit_pos
% Attention: the function gets as input only the dataline columns holding
% the predefined bit pattern.

% Copyright Erwin Roth, 16.01.2006, erwin.roth@weihenstephan.org.

function dw_pattern_sum = calc_dw_pattern_sum(dataline_dw_patterns)
% row(=DW) count and number of pattern bits
[dw_cnt, pattern_bit_cnt] = size(dataline_dw_patterns);

dw_idx = 1;
while (dw_idx <= dw_cnt)
    % go through all DWs

     % assumed bit order is big endian, so rightmost bit is 2^0
    bit_idx = pattern_bit_cnt;
    dw_pattern_sum(dw_idx,1) = 0;
    while bit_idx >= 1
        % go through all bits
        dw_pattern_sum(dw_idx,1) = dw_pattern_sum(dw_idx,1) + ...
            (dataline_dw_patterns(dw_idx, bit_idx))*2^-...
            (bit_idx-pattern_bit_cnt);
        bit_idx = bit_idx - 1;
    end

    dw_idx = dw_idx + 1;      % next dataword
end
```

C.4 canon_nozzle_mapping_calc.m

```
% Canon Printhead Nozzle Mapping Calculation
%
% Script to calculate relativ dataword index to nozzle set mapping
% by using Least Squares Algorithm
%
% Function returns the relativ DW index numbers sorted by their location
% relativ to the printhead contact plate.
%
% Copyright 2006, Erwin Roth, erwin.roth@weihenstephan.org.


function [rel_dw_to_set_map, set_to_rel_dw_map] = ...
    canon_nozzle_mapping_calc(plotdata)

idx = 1;
while idx <= 24
    x = plotdata(find(plotdata(:,idx+2)),1);
    % TILT LINE REFERENCE VALUE
    ref = -1.6556*x+1231.2;     % got by manual linear curve fitting
    y = plotdata(find(plotdata(:,idx+2)),idx+2);

    % calculate first order polynom coefficients to minimize error
    % use Least Squares algorithm by using polyfit()
    coeff(idx,:) = [polyfit(x, y-ref, 1)];

    %debug output
    y_calc = coeff(idx,end-1)*x(1)+coeff(idx,end);
    disp([ref(1), y(1), y_calc]);

    idx = idx+1;
end

[s1,s2] = sort(coeff(:,end));
rel_dw_to_set_map = length(s2)+1-s2(:,end);

% calculate map in opposite direction, necessary for signal-generator
% program
idx=1;
while idx <=length(rel_dw_to_set_map)
    set_to_rel_dw_map(rel_dw_to_set_map(idx))=idx;
    idx=idx+1;
end

return;
```

C.5 canon_2d_pattern_generator.m

```
% Canon 2D C++ Voxel Command Generator
%
% Transforms a low resolution input pattern into the right display
% resolution and generates the required C++ function
% calls to activate voxels at the desired positions.
%
% The output generated by this script is put as C++ source code part
% into the Signal Generator main.cpp Source Code file.
%
% Copyright 2006
% Author: Erwin Roth, erwin.roth@weihenstephan.org

%% Define Constants
l_row_cnt = 25;            % ATTENTION: must reflect row and column
l_column_cnt = 48;         % count of low_res_pattern!!

scaling_factor_x = 16;     % horizontal scaling factor (48*16=768!)
scaling_factor_y = 1;      % vertical scaling factor, experiments
                           % showed, the resulting DTB count should
                           % not be higher than 150.

% Filename of output file with C++ voxel creation commands
filename='voxel-commands.txt';

% create empty high res pattern with zeros filled
clear high_res_pattern;
high_res_pattern=zeros(l_row_cnt*scaling_factor_y, l_column_cnt*...
    scaling_factor_x);

% create plot for input pattern checking
figure;
hold on;
for l_row = 1:l_row_cnt
    for l_column = 1:l_column_cnt
        % check if voxel is set in low res pattern
        % ATTENTION!! the image needs to be flipped vertically for
        % correct DTB output sequence
        % here we do the checking & flipping
        if (low_res_pattern(l_row_cnt+1-l_row,l_column))
            plot(l_column,l_row,'bo');
            % if low res bit voxel is set, create in high res pattern a
            % square with scaling_factor*scaling_factor size at the
            % corresponding position
            h_row_start = (l_row-1)*scaling_factor_y;
            h_column_start = (l_column-1)*scaling_factor_x;
            for h_row = h_row_start+1:h_row_start+scaling_factor_y
                for h_column = h_column_start+1:h_column_start+...
                                scaling_factor_x
                    high_res_pattern(h_row,h_column) = 1;
                end
            end


        end
    end
end

% Enable file output
file_id = fopen(filename,'w');

%% create C++ commands to set bit (=voxel) in a hardcoded way
for h_row = 1:l_row_cnt*scaling_factor_y
    for h_column = 1:l_column_cnt*scaling_factor_x
        % check if voxel in high resolution pattern is set
        if (high_res_pattern(h_row, h_column))
            fprintf(file_id,'sigGen.addActiveNozzle(dtb+%d,array,%d);\n',...
                h_row-1,h_column-1);
```

```
            %disp(string);
        end

    end

end

%close file
fclose(file_id);
```