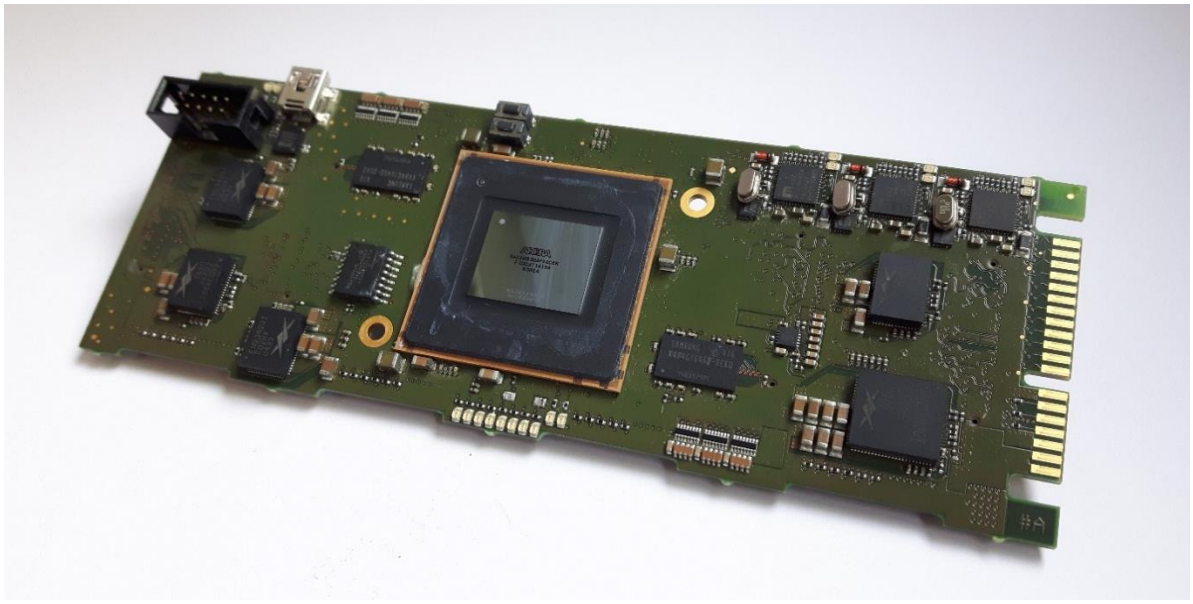
Fachhochschule Nordwestschweiz
Hochschule für Technik

SAPE FPGA
User Manual



Verfasser/in: Dino Zardet

Windisch, 10.04.2019

Version: 1.1

Institut für Sensorik und Elektronik
Klosterzelgstr. 2
5210 Windisch
dino.zardet@fhnw.ch
www.fhnw.ch

Inhaltsverzeichnis

1 SAPE FPGA

The SAPE FPGA is used to control up to four Fuji ink jet print heads. Each print head has a print resolution of 1'200 dpi and thus a print resolution of 21.167 µm. The 2048 jets, which are distributed over 608 pixel lines, are able to print up to three drops per dot using user-configured waveforms. The print speeds is specified and limited to 90 kHz or 1.905 m/s or 114 m/min. The FPGA is driven by a clock frequency of 140 MHz.

1.1 Function

After configuration and before print start (print-go – PG) the host needs to provide image data and image-information parameters. The image data is transferred as UDP packets through two dedicated 1 Gbit Ethernet interfaces and stored into an external 1GB DDR3-memory. The sets of image-information for each single image is provided by writing registers, which are then copied into a dedicated image-information FIFO.
The encoder board provides the current print position as telegram. The absolute position with a step resolution of 2.646 µm corresponds to an eight times oversampling (8 x 2.646 µm = 21.167 µm). An encoder firepulse is internally fired every 21.167 µm where the print heads are loaded with the drop (image) data. Additionally the external DACs are started to generate an analog pulse sequence, the so-called waveform. The waveform has to be loaded in the configuration phase into the DACs internal 4096 × 14 pattern memory. The following analog pulse generator amplifies the pulse sequence to the specified voltage levels needed by the print heads.
At the point where the image print start position is equal to the encoder position, a corresponding PG command initiates the print process for the current image.

The block diagram in Figure 1 below shows the internal structure of the SAPE FPGA and the surrounding components.

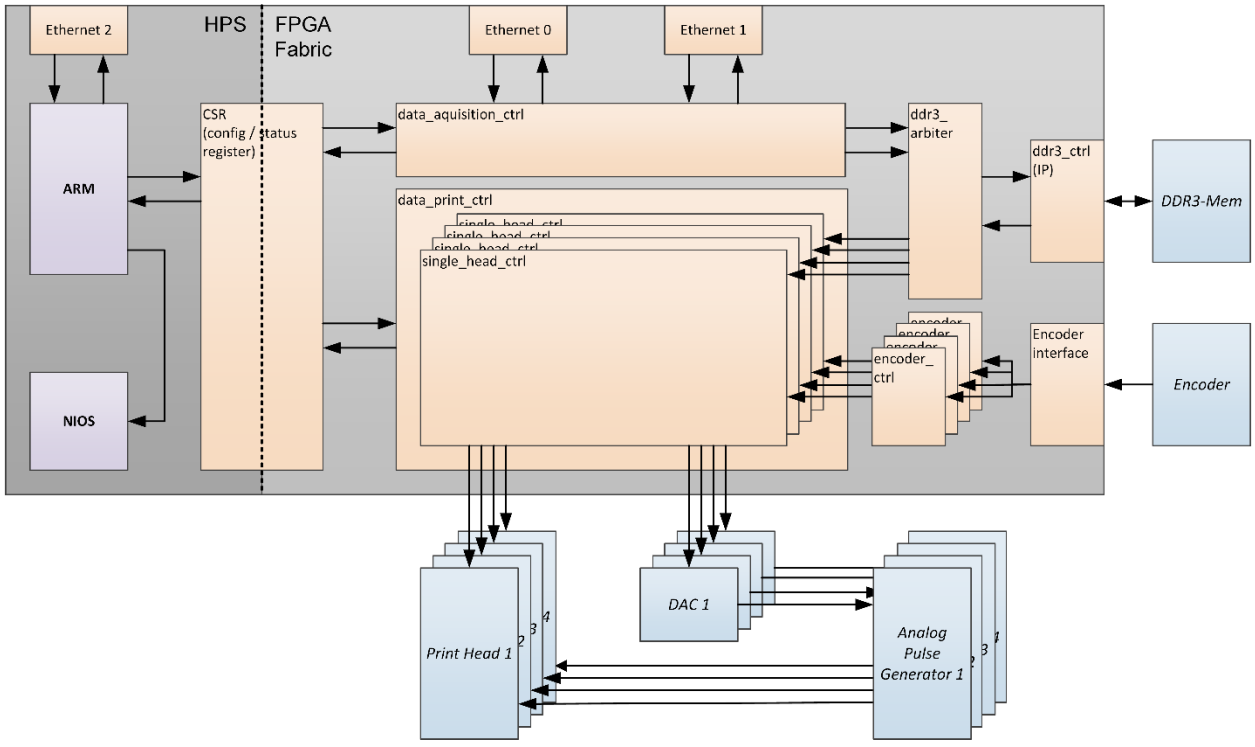


Figure 1: SAPE FPGA block diagram

1.2 Different phases after power-up

After power-up of the FPGA there are different phases as can be seen in the Figure 2 below:

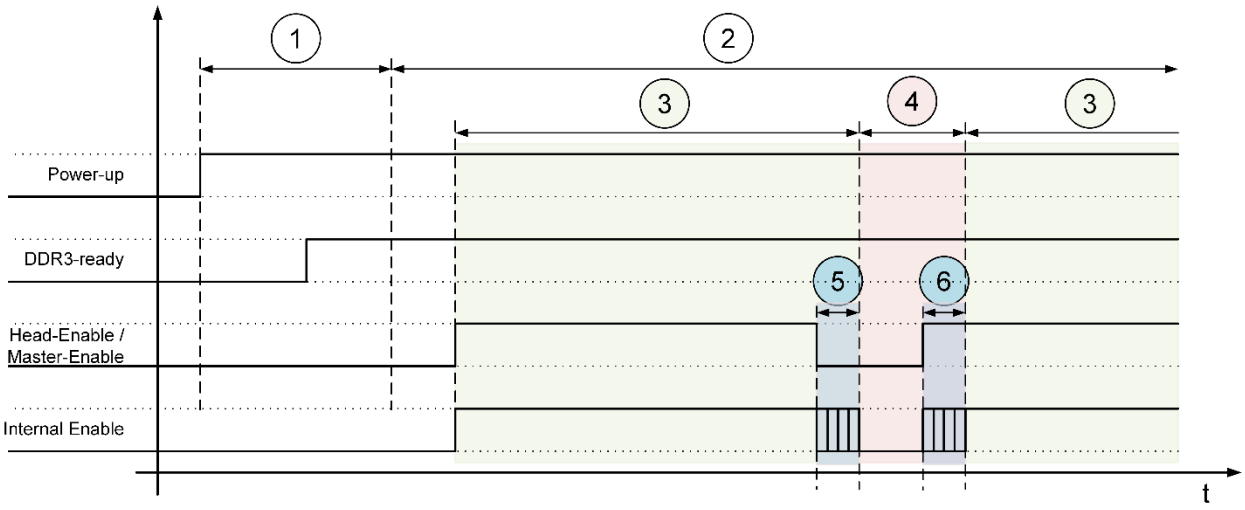


Figure 2: Operating phases after power-up

(1) **Configuration phase:** These registers have to be configured before entering the printing phase (3) and are typically static, meaning they do not change for a specific application during running or idle time. Write configuration registers at address

a) 0x00000 – 0x0009C Setup general registers
b) 0x01000 – 0x01104 Setup head0 registers
c) 0x02000 – 0x02104 Setup head1 registers
d) 0x03000 – 0x03104 Setup head2 registers
e) 0x04000 – 0x04104 Setup head3 registers
f) 0x05000 – 0x0500C Setup ethernet0 registers
g) 0x06000 – 0x0600C Setup ethernet1 registers
h) 0x07000 – 0x07020 Setup UDP-flag start / end ranges per head
i) 0x28000 – 0x28010 Setup encoder registers

Note: Before entering the control phase (2) the DDR3-memory initialization phase has to be finished. This can be checked by reading bit 0 (DDR3 ready) of register at address 0x2C00C until it is '1'.

(2) **Running phase:** The configuration registers mentioned above usually remain untouched in this phase. Control registers can be written and read during running (3) and idle (4) phases except during the short delay times when head-enable or master-enable goes low (5) or high (6). Write and read control registers at address

a) 0x20000 – 0x200D8 Write image-information registers
b) 0x08000 – 0x1EC14 Read UDP-flags registers to check if image-data is available in the DDR3-memory space or has to be retransmitted
c) 0x2C000 – 0x2C33C Read and check status registers
d) 0x2D000 – 0x2D0AC Read and check error registers
e) 0x2E000 – 0x2E09C Read and check info registers

(3) **Printing phase:** Encoder firepulses are enabled and each time a firepulse or a PG reaches the single_head_ctrl a waveform (via DAC) is started with corresponding loading of image- or dummy-lines followed by loading of the print-heads with drop-data. The waveform length can have a maximum length of 4095 cycles or 58.5 µs corresponding to a printing speed of 17.1 kHz or 0.36 m/s. A shorter waveform length corresponds to a higher maximum print speed.

(4) **Idle phase:** During this phase, encoder firepulses are masked and no printing or loading of print-heads is initiated.

(5) **Disable delay:** In case the head-enable or master-enable is deasserted (change from '1' to '0') during a running waveform sequence, the internal disable is delayed until it has finished the currently ongoing waveform sequence. This is done to prevent any running state machine (FSM) of hanging due to an unacknowledged request from another FSM. Therefore, maximum expected delay corresponds to the currently programmed waveform length. As the longest configurable waveform length is 4096, the maximum delay is less than 59 µs.
As soon as the internal delayed disable is active, the corresponding head UDP-flags memory is cleared to mark the DDR3-memory space as free again. The time needed for this process can be found in chapter 3.5.1. Additionally the clearing of the head-memory, which consists of 608 image lines, is initiated. The time needed for this process can be found in chapter 3.5.2.
Note: This can be checked by reading bit 16 (clearing UDP-flags) of register at address 0x2C000 until it is '0'.

(6) **Enable delay:** In case the head-enable or master-enable is asserted again (change from '0' to '1') and the clearing process of the UDP-flags memory has not finished yet the internal enable is delayed accordingly. In contrast, the clearing process of the head-memory is not finished but is interrupted. This means that the head-memory may possibly not be cleared completely for the case where the head-enable or master-enable is asserted before the clearing process has finished.

1.3 Image-data management

Image data is transferred from the host to the FPGA as UDP packet. The UDP packet has a configurable image-payload size of 1'440 byte, or 2'880 byte, or 5'760 byte, or 8'640 byte. The FPGA internal data word size is defined as 256 bit or 32 byte. The payload sizes are multiples of the internal data word size as can be seen in Table 1 below:

Configuration	UDP payload size	Number of int. words
0	1'440 byte	45 words
1	2'880 byte	90 words
2	5'760 byte	180 words
3	8'640 byte	270 words

Table 1: Configurable UDP image-payload size

To transmit image-data in UDP packets there are some requirements that have to be fulfilled.

1) Each image-line is transferred in blocks with a size, which is a multiple of 256 bit (32 byte), the remaining bits are padded.

2) Image-data from one image cannot be shared with image-data of another image within the same UDP block. After the last image line, the remaining space to the end of the UDP block has to be padded.

Example: Let's assume we have an image width (line size) of 999 pixels each 1 bit. The image length is defined as 99 pixels (number of lines).
Line size: 999 bit / 8 bit = 124 byte and 7 bit -> 125 byte
Image size: 99 x 128 byte = 12'375 byte

With padding, this results in:
Line size: 125 byte -> next multiple of 32 byte = 128 byte
Image size: 99 x 128 byte = 12'672 byte

Configured UDP-image-payload size:	1'440 byte
Number of used UDP packets:	12'672 byte / 1'440 byte = 8.8 -> 9 UDP packets
Image padding bytes:	9 x 1'440 byte – 12'672 byte = 288 byte
Image size:	99 x 999 = 98'901 bit
Image size in UDP packets:	9 x 1'440 byte x 8 bit = 103'680 bit
Used to total ratio:	98'901 bit / 103'680 bit = 95.4 %

Figure 3 below shows the needed UDP packets and the splitting of the image-data for the example above.

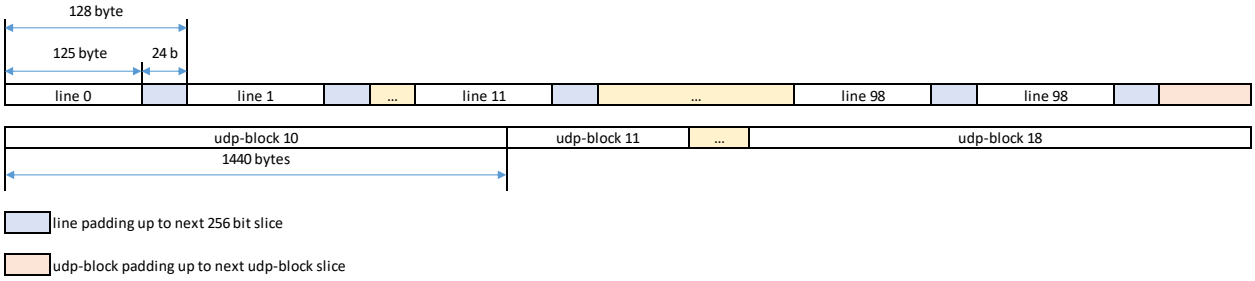


Figure 3: Padding of image-lines and images in UDP-packets

In order to store the image-data at the correct address in the external DDR3-memory the UDP packets have four leading byte in the payload that indicate the UDP-packet number. The four bytes themselves are not stored in the DDR3-memory but are only used to calculate the DDR3 address.
The block data_aquisition_ctrl takes the received UDP-block number and with the configured UDP packet size (0 to 3) in register at address 0x07020 it is able to calculate the address. Additionally it takes the UDP-block number and sets the corresponding UDP-flag in the UDP-flags memory. Using this mechanism the host is able to check if all image data was received and stored correctly in the DDR3-memory. In case a UDP-block-flag is missing in the UDP-flags memory, it has the possibility to re-send the missing UDP packet before it is used.

Table 2 shows the number of UDP-blocks available in the DDR3-memory space depending on the configured UDP image payload size.

Configuration	UDP payload size	Number of UDP-blocks available in DDR3-memory
0	1'440 byte	745'472 UDP-blocks (186'368 per head)
1	2'880 byte	372'736 UDP-blocks (93'184 per head)
2	5'760 byte	186'638 UDP-blocks (46'659 per head)
3	8'640 byte	124'273 UDP-blocks (31'068 per head)

Table 2: Number of UDP-blocks available in DDR3-memory space depending on configured UDP image-payload size

The host can split the total available memory space and define a start and an end UDP-block number for each print head by writing the corresponding registers at address 0x07000 to 0x0701C. The memory space can be split evenly but it do not has to. It can be defined from one head range to the next range seamlessly or with a gap. If it is split evenly, each of the four print heads gets a quarter of the whole memory space.
Accordingly, the available UDP-block flags (flags set by the data_acquisition_ctrl and optionally cleared by the data_print_ctrl block) in the UDP-flags memory must correspond to the maximum number of available UDP-block numbers for the DDR3-memory space.
The host can determine the current fill level per print head by reading the registers at address 0x2E080 to 0x2E09C, which show how many UDP-block flags are currently set to '1'. The get a percentage number just divide these values by the total number of UDP-flags memory space assigned to each print head.

1.4 Image-information management

A set of image-information is transferred from the host to the FPGA (9 registers per print head at address 0x20040 to 0x200D8), which is then stored internally in a 128 entries deep image-information FIFO per head by incrementing the image-in-index (register per print head at address 0x20000 to 0x2000C). The image-information set consists of following parameters:

Image parameter	#Bit	Description
image bit per pixel	2	0: 1 bit / pixel, 1: 2 bit / pixel
image start UDP-block number	20	UDP-block number -> DDR3 start address
image width in pixel	12	Image width in pixel
image width in byte	10	Image width in byte
image length in pixel	18	Image length in pixel
image print x-offset	4	Offset: 0…15 pixel
print direction	1	0: forward, 1: backward
flip image left-right	1	0: no flip, 1: flip left right
UDP-block flag clear	1	0: no clear UDP-flag after print, 1: clear flag

Table 3: Image parameters in the image-information set

Note: In contrast to the image-data, which can be written in any order, the host has to deliver the image-information in image print order.

Beside the image-in-index, which is incremented by the host each time a new image-information has been provided, the two image-out-indices (index1 at bit 15:8 and index2 at bit 7:0) available on registers at address 0x2E000 to 0x2E00C are incremented only when the corresponding image-information of the current image has been processed completely. The image-print-done counter, which is incremented only when the entire image until the last image-line has been printed or loaded to the print heads.

Important issues:

- The host has to take care that on any time during printing phase enough image-information sets are available in the image-information FIFO. Otherwise the image-line error counter 2 (see also chapter 2.4) is incremented
- The host has to take care that the image-in-index never reachs the same value as any of the image-out-indices. Otherwise this results into an image-information FIFO overflow which is not secured by the FPGA logic
- When head-enable or master-enable goes low ('1' to '0') the **image-in-index** and both image-out-indices are reset automatically to zero

1.5 Image print start management

The image print start or PG position is transferred through the encoder telegram. When a PG is detected the current positon, added to the configurable firepulse offset value defined in the head configuration registers, is written into a 128 entries deep image-PG FIFO and the pg-in-index is incremented. At the FIFO output, the current value is compared to the current position from the encoder telegram. If they match an image-print-start pulse is generated and the pg-out-index is incremented.
Note: The image PG is used sequentially and according to the order of the received image-information parameters.

1.6 Relation between print-go, image-out-index, and image-print-done

The different **image counters** are incremented sequentially at different points in the data path. The time relationship between the counters is shown below:

Nr	Head0	Head1	Head2	Head3	Counter	Source
1	0x2C2EC (7:0)	0x2C2EC (15:8)	0x2C2EC (23:16)	0x2C2EC (31:24)	print-go	encoder-in
2	0x2E040	0x2E044	0x2E048	0x2E04C	print-go	pg-in-index
3	0x2E050	0x2E054	0x2E058	0x2E05C	print-go	pg-out-index
4	0x2C290 0x2C2A0	0x2C294 0x2C2A4	0x2C298 0x2C2A8	0x2C29C 0x2C2AC	print-go ok / print-go late	encoder-out 1
5	0x2C2D0	0x2C2D4	0x2C2D8	0x2C2DC	print-go	encoder-out 2
6	0x2C010 (7:0)	0x2C010 (15:8)	0x2C010 (23:16)	0x2C010 (31:24)	print-go	head_req_fp_data
7	0x2E000 (15:8)	0x2E004 (15:8)	0x2E008 (15:8)	0x2E00C (15:8)	image-out-index1 (wrap-around after 127)	img_line_fifo_wr_ctrl
8	0x2E000 (7:0)	0x2E004 (7:0)	0x2E008 (7:0)	0x2E00C (7:0)	image-out-index2 (wrap-around after 127)	main_head_ctrl
9	0x2C1C0	0x2C1C4	0x2C1C8	0x2C1CC	print-done	rd_head_mem

Table 4: Different "image-counters" among the data path

The increment delay times between the counters are as follow:

Delay	Description
From 1 to 2	Almost no delay
From 2 to 3	Depends on head "offset-firepulse" parameter
From 3 to 4	Almost no delay
From 4 to 5	Almost no delay
From 5 to 6	Almost no delay
From 6 to 7	Almost no delay (note: 6 can also be ahead of 5 on very small images)
From 7 to 8	Between 14 and max 255 encoder-firepulses (= image-lines)
From 8 to 9	608 encoder-firepulses (= image-lines)

Table 5: Time delays among the "image-counters"

2 Block single-head-control

Within the block data_print_ctrl there are four single_head_ctrl blocks for each individual print head.

2.1 Block diagram

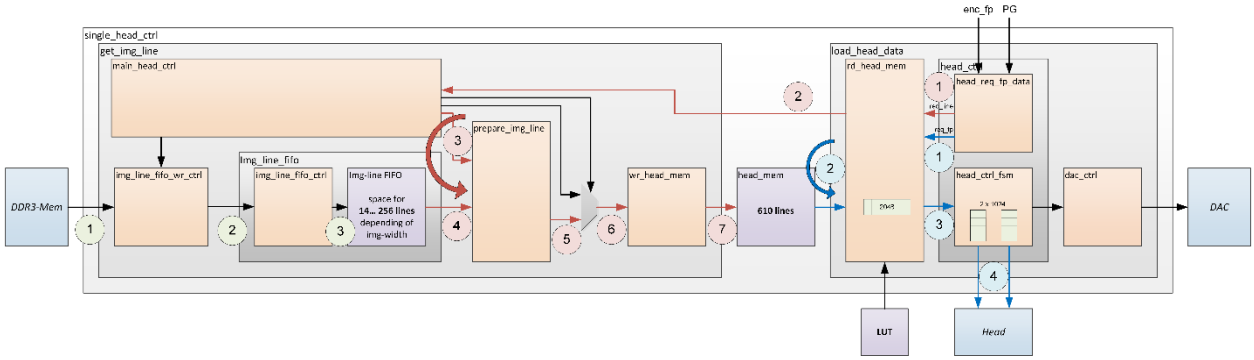


Figure 4: single-head-control block diagram

2.2 Dataflow

The dataflow in signle-head-cotrol block diagram in Figure 4 basically shows three different paths (green, red, and blue). These three paths are described in detail below:

1. Black arrows and green numbers 1 to 3: As soon as both image-information (from internal image-information FIFO) and image-data (from external DDR3 memory), represented by UDP-flags (from internal UDP-flags memory) marking the presence of corresponding image data, are available, image-lines are loaded from the external DDR3-memory into the internal image-line FIFO. The FIFO has a capacity of 14 to 256 image-lines depending on the image width. As soon as new space is available in the FIFO the next line of the current image or if the current image is finished the first line of the next image is loaded (of course only if the corresponding image-information and image-data is available). During read of image-lines according to the image-information, the corresponding UDP-flags are cleared (marking the DDR3 memory space as free again) or not (remaining in the DDR3-memory space).

2. Red arrows and red numbers 1 to 7: On an encoder-firepluse (all 21.167 µm) the waveform sequence on the DAC is started. As a first part of the waveform cycle, an image-line needs to be loaded into the head-memory. As long as no image printed (no PG received) a dummy image-line (a zero line) is loaded into the head-memory (not from the image-line FIFO, since it contains only real image-lines) and the oldest line in the head-memory is overwritten. As mentioned before the head-memory has a capacity of 608 lines. As soon as a PG is received image-lines from the image-line FIFO are loaded. From image to image print three valid cases are distinguished:
   - With a gap: Dummy lines are loaded in the mean time
   - Without gap or seamless: No dummy lines are loaded at all
   - With overlap: The current image is stopped (skipped) and the next image is started immediately with the first line. In this case, the remaining image-lines of the current image in the image-line FIFO need to be flushed and the corresponding UDP-block-flags in the UDP-flags memory, marking the occupied space in the DDR3-memory needs to be cleared (but only if defined so in the image-information parameters).
3. Blue arrows and blue numbers 1 to 4: The second part after having loaded an image-line into the head-memory consists of loading the drop data for the jets to the print head. According to the head configuration, 1 to 3 drops (resulting in different drop sizes) are loaded to the print head per waveform cycle.

2.3 Dataflow from encoder firepulse to print heads

The three data paths described above can also be seen in the simulator waveforms figures below.

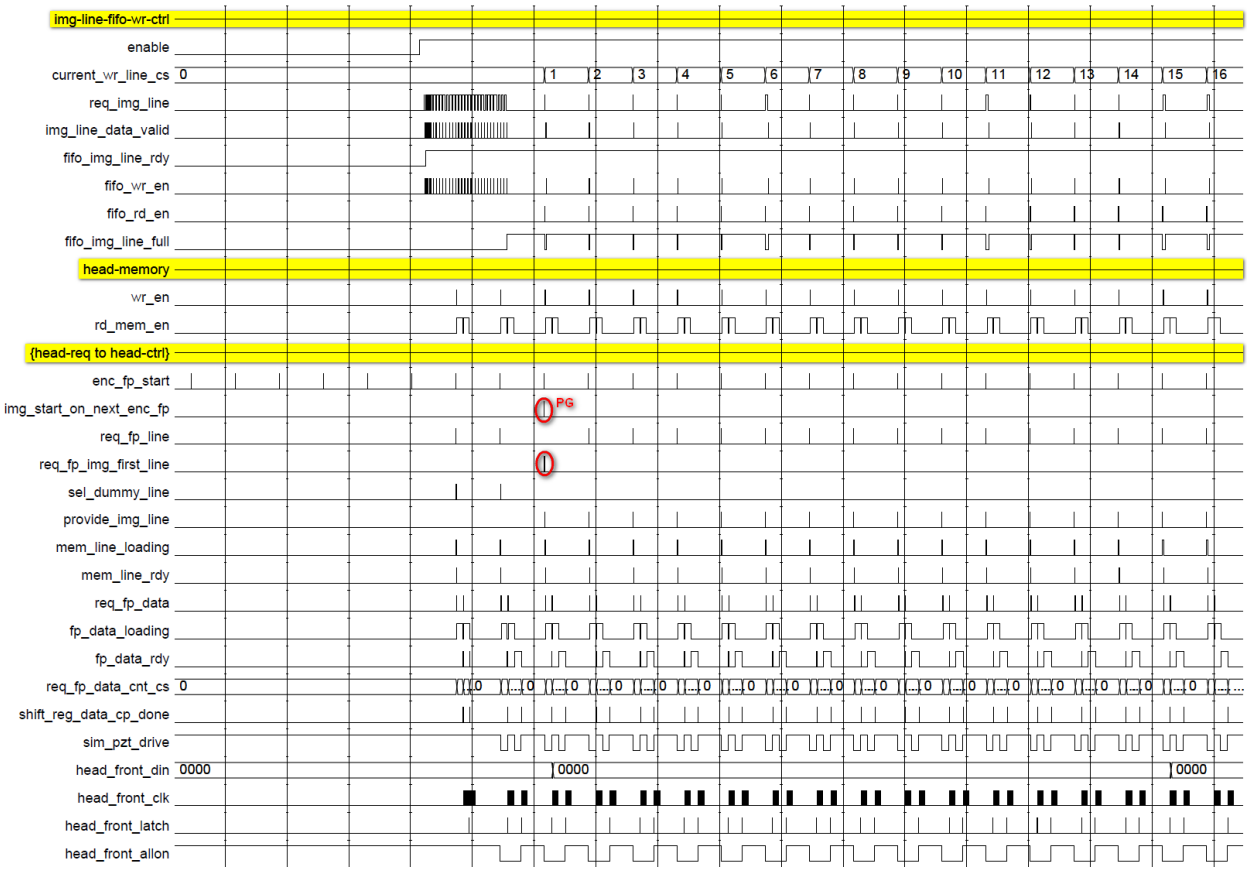


Figure 5: Encoder firepulses and PG sequences

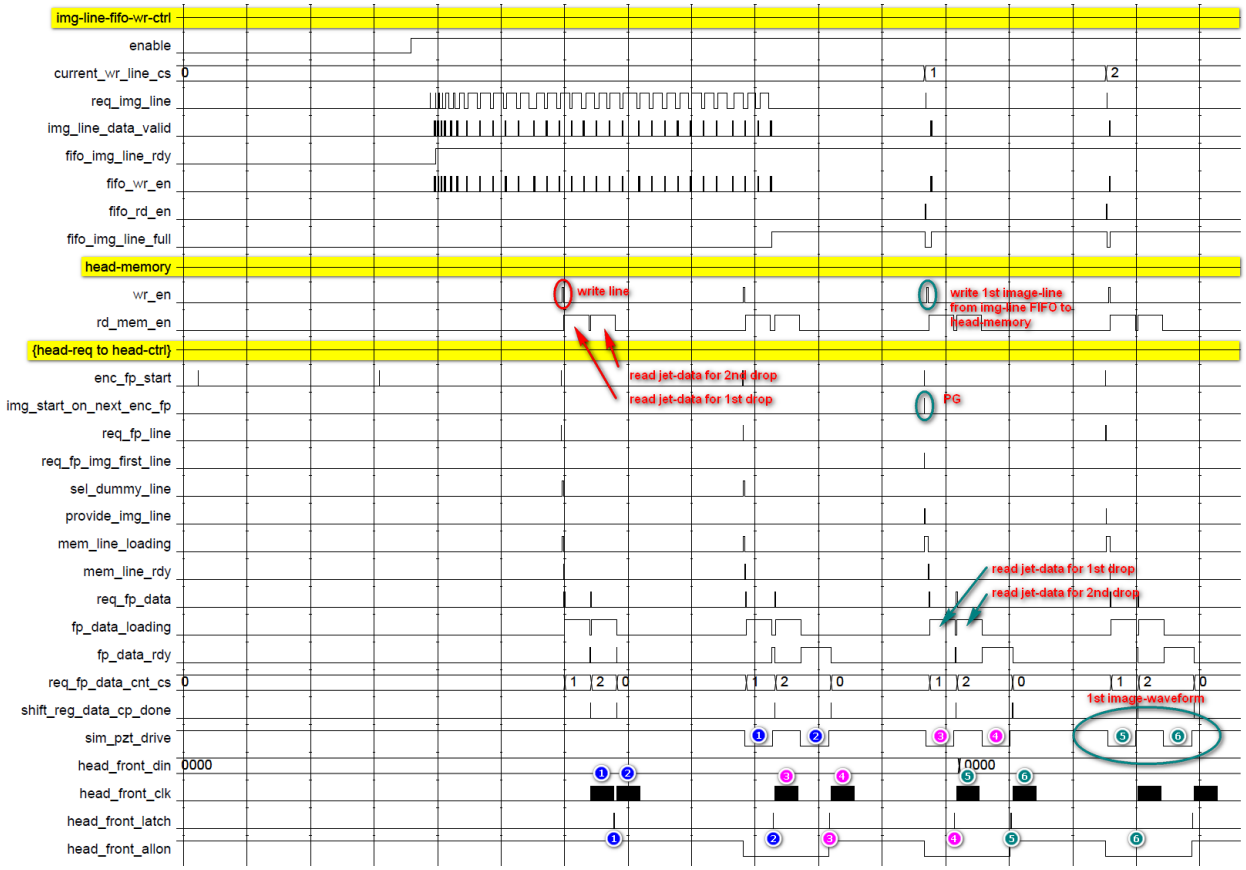


Figure 6: Encoder firepulses and PG sequences (zommed)

1) As soon as enable goes high and image-information is available and UDP-flags mark corresponding image-data ready in the DDR3-memory, the img_line_fifo_wr_ctrl loads image-data from the DDR3-memory to the image-line FIFO (yellow marked upper part of the diagrams) until it is full (signal fifo_img_line_full).
2) On each encoder firepulse (signal enc_fp_start) and no corresponding print-go (PG or signal img_start_on_next_enc_fp), a dummy-line (signal sel_dummy_line) is loaded into the head-memory (always overwriting the oldest line in the memory). Then the waveform is started (signal sim_pzt_drive) and the print head drop data is read from the head-memory and loaded to the print heads (numbered blue and magenta in the lower part of the Figure 5 and Figure 6).
3) When an encoder firepulse along with a PG pulse occurs, the first image-line (signal provide_img_line) is loaded from the image-line FIFO into the head-memory and accordingly the print head drop data is read from the head-memory and loaded into the print heads. This process is repeated for each further image-line as long as the image has not been completely processed. After the last image line and no further PG the head-memory is again filled with dummy lines on each encoder firepulse.

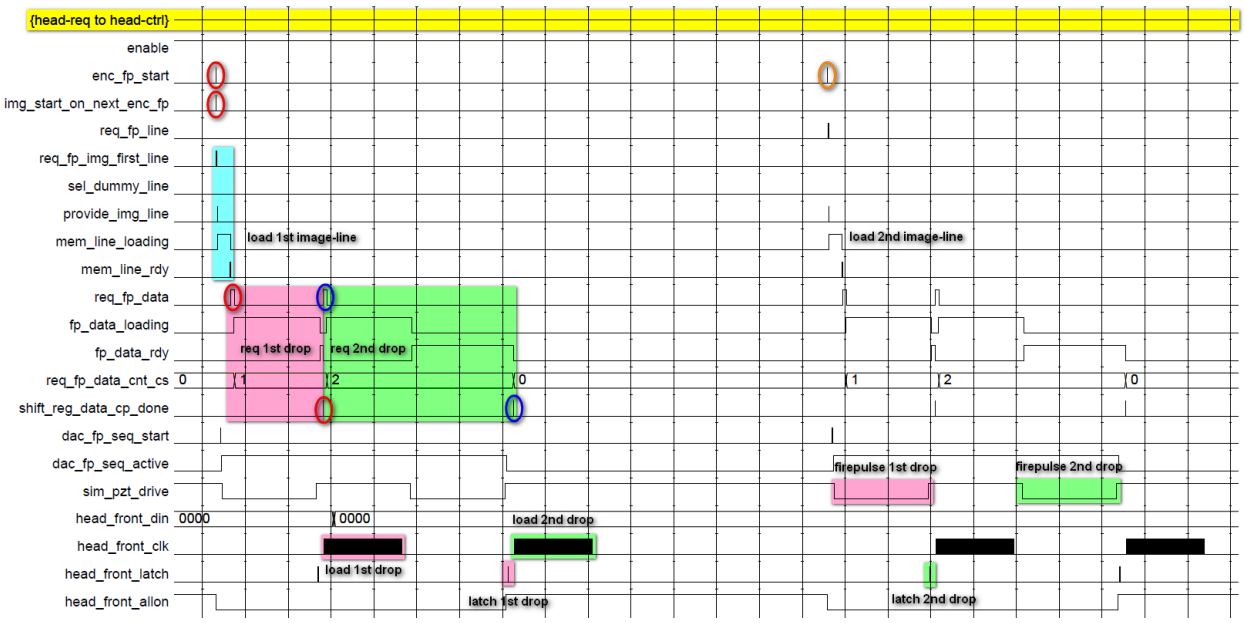


Figure 7: Encoder firepulses and PG sequences (more zoomed)

In the Figure 7 above can be seen that the process described above – loading the image-line from the image-line FIFO to the head-memory (marked light blue) and reading drop data from the head-memory and load them to the print head (marked pink and green) – has been moved to the earliest point in time possible in order to maximize the print speed.

2.4 Overview of some important error registers

Error counters for head0:

Error counter	Address	Range	Source	Description
Write image-line error	0x2D01C	(31:16)	Main_head_ctrl	On PG with image skip but no further img-information available
Image-line error 1	0x2D0A0	(7:0)	Main_head_ctrl	1st image-line missing in image-line FIFO
Image-line error 2	0x2D0A4	(7:0)	Main_head_ctrl	1st image-line missing due to missing img-info
Image-line error 4	0x2D0AC	(7:0)	Main_head_ctrl	Image-line missing in image-line FIFO (not 1st line)
Fifo image-line error	0x2D01C	(15:0)	Img_line_fifo	Image-line FIFO over- or underflow
Head-mem not ready error	0x2D020	(31:16)	Prepare_img_line	New Line-request when is still busy
Latch missed error	0x2D024	(31:16)	Head_ctrl_fsm	Latch could not be applied due to waveform firepulse already active
Enc fp missed error	0x2D060	(15:0)	Head_ctrl_fsm	Image-line skipped

Table 6: Error counters for print head 0

Error counters for head1:

Error counter	Address	Range	Source	Description
Write image-line error	0x2D028	(31:16)	Main_head_ctrl	On PG with image skip but no further img-information available
Image-line error 1	0x2D0A0	(15:8)	Main_head_ctrl	1st image-line missing in image-line FIFO
Image-line error 2	0x2D0A4	(15:8)	Main_head_ctrl	1st image-line missing due to missing img-info
Image-line error 4	0x2D0AC	(15:8)	Main_head_ctrl	Image-line missing in image-line FIFO (not 1st line)
Fifo image-line error	0x2D028	(15:0)	Img_line_fifo	Image-line FIFO over- or underflow
Head-mem not ready error	0x2D02C	(31:16)	Prepare_img_line	New Line-request when is still busy
Latch missed error	0x2D030	(31:16)	Head_ctrl_fsm	Latch could not be applied due to waveform firepulse already active
Enc fp missed error	0x2D068	(15:0)	Head_ctrl_fsm	Image-line skipped

Table 7: Error counters for print head 1

Error counters for head2:

Error counter	Address	Range	Source	Description
Write image-line error	0x2D034	(31:16)	Main_head_ctrl	On PG with image skip but no further img-information available
Image-line error 1	0x2D0A0	(23:16)	Main_head_ctrl	1st image-line missing in image-line FIFO
Image-line error 2	0x2D0A4	(23:16)	Main_head_ctrl	1st image-line missing due to missing img-info
Image-line error 4	0x2D0AC	(23:16)	Main_head_ctrl	Image-line missing in image-line FIFO (not 1st line)
Fifo image-line error	0x2D034	(15:0)	Img_line_fifo	Image-line FIFO over- or underflow
Head-mem not ready error	0x2D038	(31:16)	Prepare_img_line	New Line-request when is still busy
Latch missed error	0x2D03C	(31:16)	Head_ctrl_fsm	Latch could not be applied due to waveform firepulse already active
Enc fp missed error	0x2D070	(15:0)	Head_ctrl_fsm	Image-line skipped

Table 8: Error counters for print head 2

Error counters for head3:

Error counter	Address	Range	Source	Description
Write image-line error	0x2D040	(31:16)	Main_head_ctrl	On PG with image skip but no further img-information available
Image-line error 1	0x2D0A0	(31:24)	Main_head_ctrl	1st image-line missing in image-line FIFO
Image-line error 2	0x2D0A4	(31:24)	Main_head_ctrl	1st image-line missing due to missing img-info
Image-line error 4	0x2D0AC	(31:24)	Main_head_ctrl	Image-line missing in image-line FIFO (not 1st line)
Fifo image-line error	0x2D040	(15:0)	Img_line_fifo	Image-line FIFO over- or underflow
Head-mem not ready error	0x2D044	(31:16)	Prepare_img_line	New Line-request when is still busy
Latch missed error	0x2D048	(31:16)	Head_ctrl_fsm	Latch could not be applied due to waveform firepulse already active
Enc fp missed error	0x2D078	(15:0)	Head_ctrl_fsm	Image-line skipped

Table 9: Error counters for print head 3

3 Requirements

3.1 Configuring the waveform

The configuration of the waveform affects the registers at address:

- 0x01040 – 0x01088 Setup waveform of head0
- 0x02040 – 0x02088 Setup waveform of head1
- 0x03040 – 0x03088 Setup waveform of head2
- 0x04040 – 0x04088 Setup waveform of head3

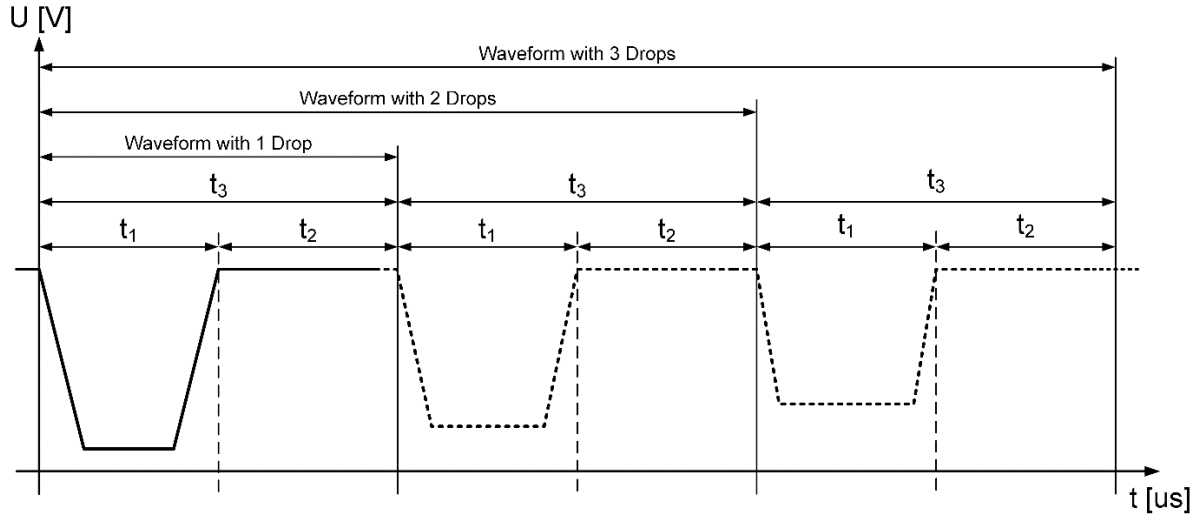


Figure 8: Waveform timings

Minimal timing requirements for 1 – 3 drops waveform:

t_{1min} = 0 ns

t_{2min} = 300 ns

t_{3min} = 2'200 ns

Resulting to a minimal timing for the waveform:

$t_{Waveform,1drop}$ = 2.2 µs ($f_{print,max}$ = 454.5 kHz)

$t_{Waveform,2drops}$ = 4.4 µs ($f_{print,max}$ = 227.2 kHz)

$t_{Waveform,3drops}$ = 6.6 µs ($f_{print,max}$ = 151.5 kHz)

Violating these requirements could result in missed latch pulses which increments a counter(31:16) at address 0x2D024 or 0x2D030 or 0x2D03C or 0x2D048, depending on the involved print head.

3.2 Calculate the maximum allowed print speed

Depending on the configured waveform length, the maximal allowed print speed is calculated as follow:

$f_{print,max}$ = 1 / (<waveform length> x $T_{clk\text{-}DAC}$ + 300 ns)

Example:

waveform length (head0 config-reg addr = 0x01044) = 466

$T_{clk\text{-}DAC}$ = 1 / 70 MHz = 14.286 ns
$f_{print,max}$ = 1 / (466 x 14.286 ns + 300 ns) = 143.737 kHz

3.3 Calculate the print speed using the status registers

The status registers for the current, minimal, and maximal print speed are located at address:

Print head	Current speed addr	Min speed addr	Max speed addr
Head0	0x02C210	0x02C214	0x02C218
Head1	0x02C21C	0x02C220	0x02C224
Head2	0x02C228	0x02C22C	0x02C230
Head3	0x02C234	0x02C238	0x02C23C

Table 10: print speed registers

With the values read in the registers above the print speed can be calculated as follow:
Print speed [Hz]: 140 MHz / <register value>
Print speed [m/s]: <print-speed in Hz> x 21.167 um
Print speed [m/min]: <print speed in m/s> x 60

Example:
Register value: 4'802
Print speed: 140 MHz / 4'802 = 29.15 kHz = 0.62 m/s = 37.02 m/min

3.4 Printing faster than allowed

In case the print speed goes slightly above the calculated maximum print speed (see chapter 3.2) a corresponding counter is incremented at register address 0x2E090 or 0x2E094 or 0x2E098 or 0x2E09C.
If the higher speed persists or an even higher print speed is applied and therefore two image-lines should be printed at the same time the older image-line is skipped and only the newer line is used. In this case another counter(15:0) is incremented at address 0x2D060 or 0x2D068 or 0x2D070 or 0x2D078.

3.5 Enable and disable using head-enable and master-enable

As mentioned above enabling and disabling is internally delayed in order to start and stop in a synchronous way. During the idle time internal processes are activated, which are described more in detail below.

3.5.1 Clear UDP-flags memory

The single head-enables are logically AND connected with the master-enable. Only when the single head-enable and the master-enable are asserted the firepulses coming from the encoder-block are not masked and printing is executed. In case the master-enable or the single head-enable is diasserted, the internal head-enable is synchronized in a way that the running wave-form is not interrupted as long as the cycle has not finished. This makes sure the stop will not cause a state machine waiting for an answer from another state machine already in the idle state, which could cause a hanging state machine.
Additionally, in case of a single head disable, the UDP-flags are cleared representing the completely configured DDR3-memory space of the corresponding print head. This procedure takes

some time to finish so the reassert of the head-enable is delayed internally in case it was asserted too early. In case the master-enable is diasserted all the UDP-flags are cleared representing the entire DDR3-memory space. Of course, this procedure takes even more time and in this case, a reassert of the master-enable is delayed internally if it was enabled too early.

The time needed to clear the complete UDP-flags memory representing the DDR3-memory space is dependent on the configured UDP image-payload size:

UDP-frame size	#UDP-flags	#clear cycles	Clear time [Clk = 140 MHz]
0 (1'440 byte)	745'472	23'296	166.4 µs
1 (2'880 byte)	372'736	11'648	83.2 µs
2 (5'760 byte)	186'638	5'833	41.7 µs
3 (8'640 byte)	124'274	3'884	27.7 µs

Table 11: Clear time for the UDP-flags memory

3.5.2 Clear head-memory

A falling edge on the head-enable initiates also a clear of the head-memory, which contains 608 image lines with the drop data used for the next print head firepluse. The difference to the UDP-block flag clear procedure is that in this case a reassert of the head-enable will interrupt the clear process and possibly leave some of the image-lines in the head-memory not cleared.

The time needed to clear the complete head-memory space is:
608 lines x 2'170 pixel x 2 bit / 256 bit/address x (1 / 140 MHz) = 74 µs

Note: When disabling the master-enable the memory clear time for UDP-flags memory takes longer than the clear time for the head-memory.

Appendix

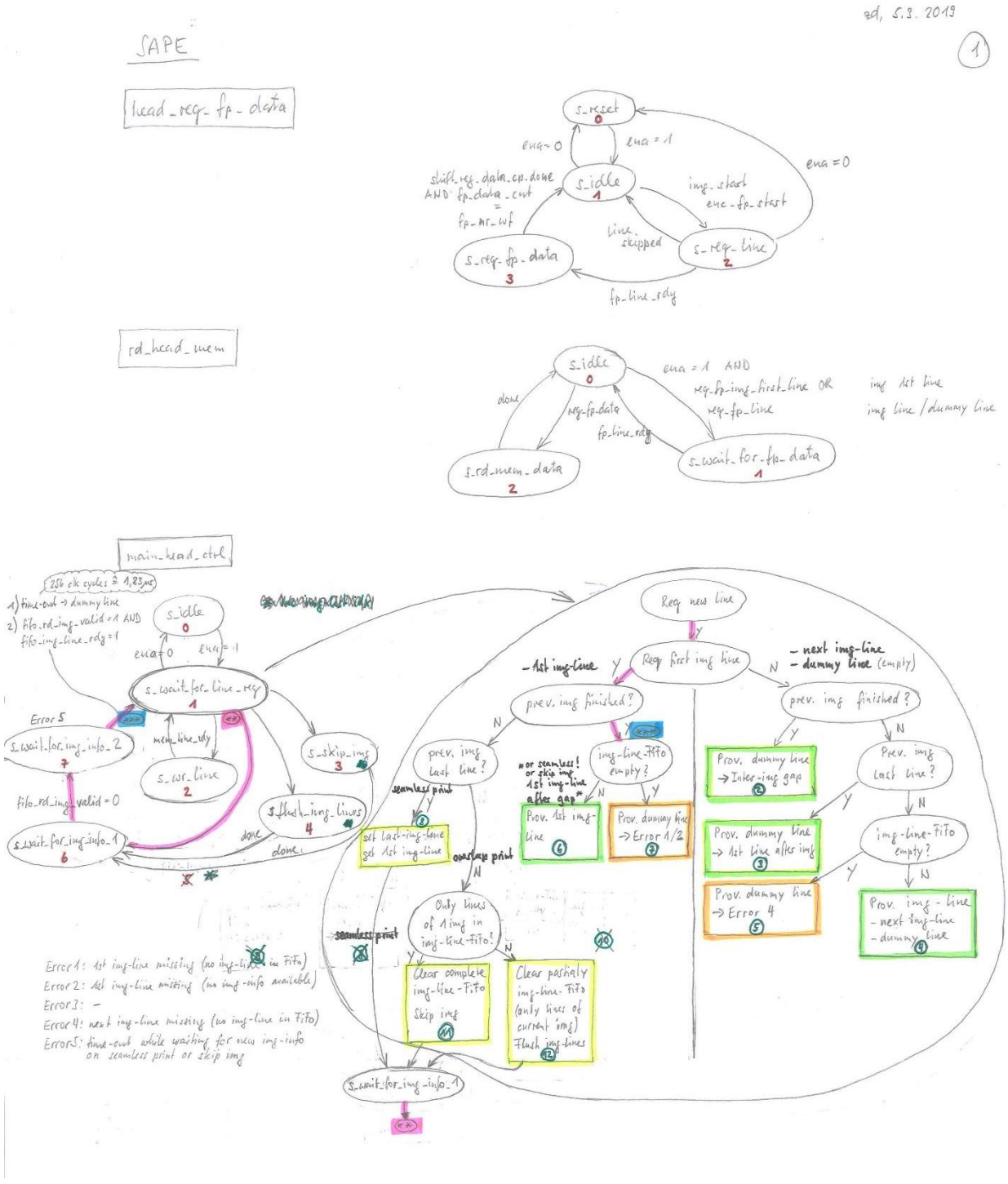


Figure 9: SAPE FSMs Part 1

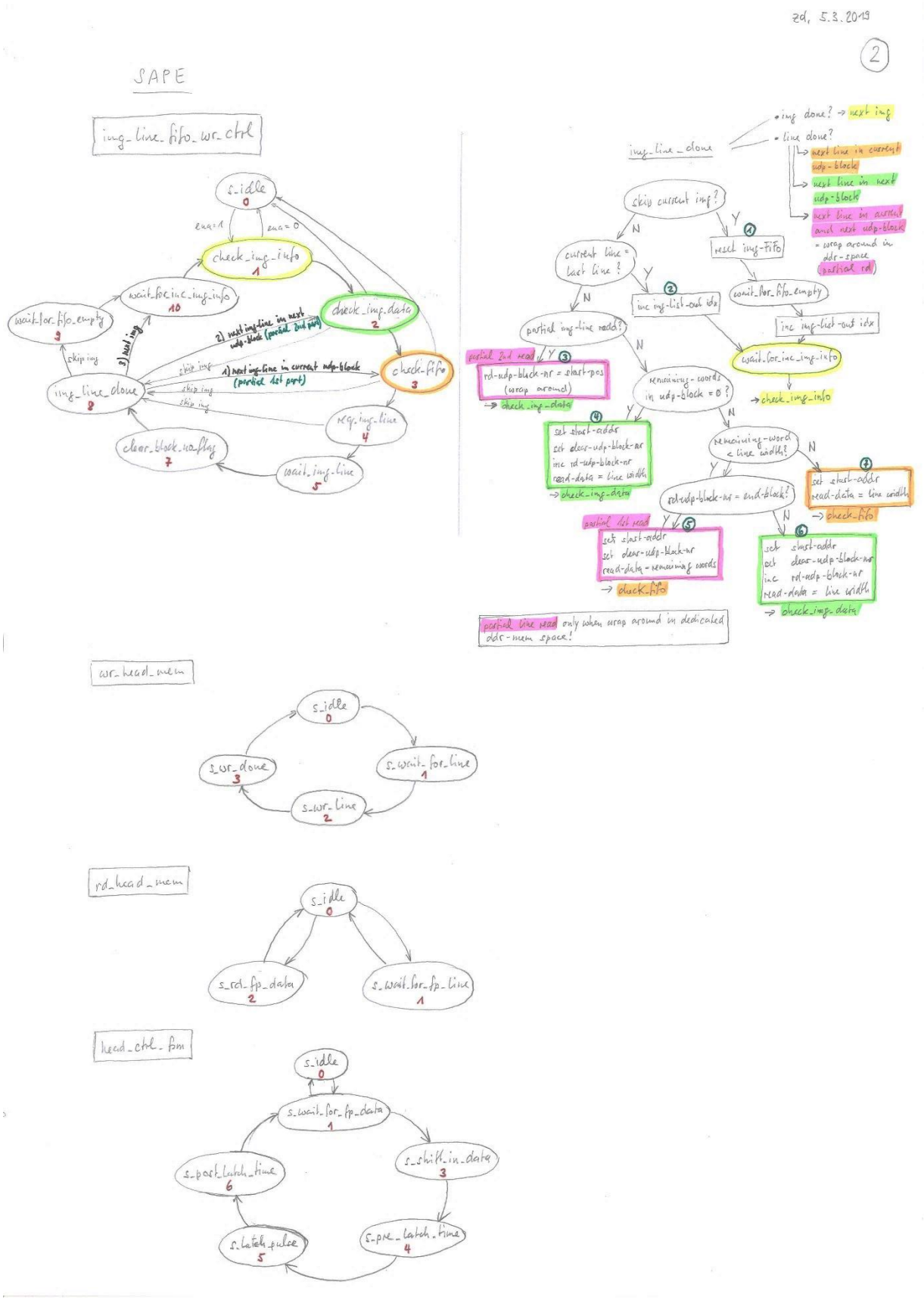


Figure 10: SAPE FSMs Part2