PAUL SCHERRER INSTITUT

PSI46 Pixel Chip - External Specification

0	18.09.2004	Revised Version of ¼ µm chip specification	Kurt Gabathuler
ED	**Date**	**Change Note**	**Originator**

Table of Contents

1 Introduction

The CMS pixel chip is organized in 26 double columns of 2x80 pixels. Each double column has in its perifery 12 8-bit time stamp buffers and 32 data buffers (both circular, see illustration 1). The physical size of the chip with its connection pads is given in illustration 2.

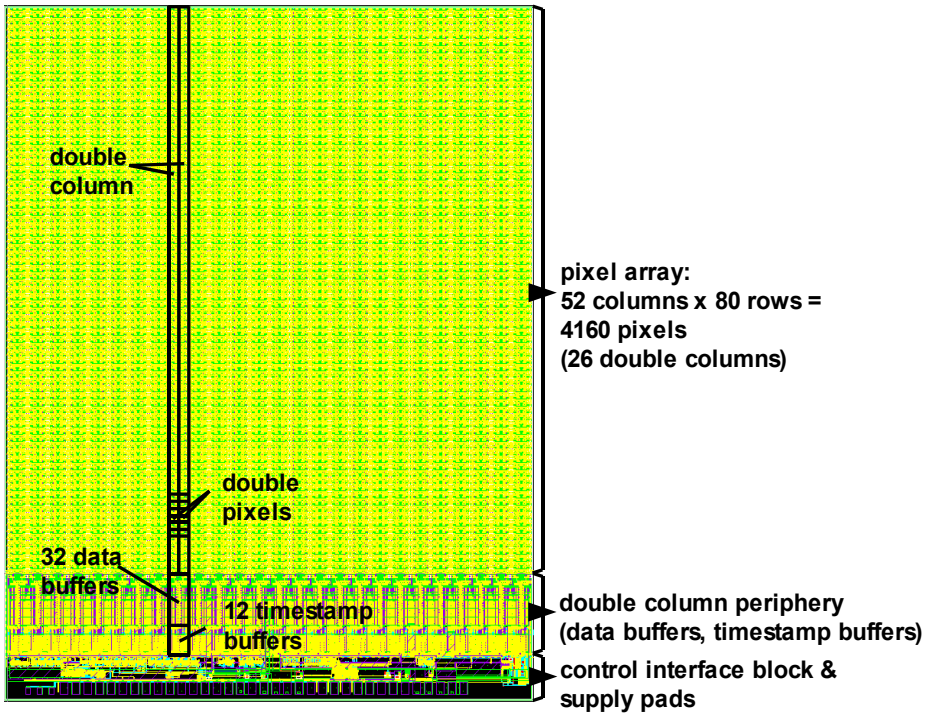


Illustration 1: Arrangement of pixel chip.

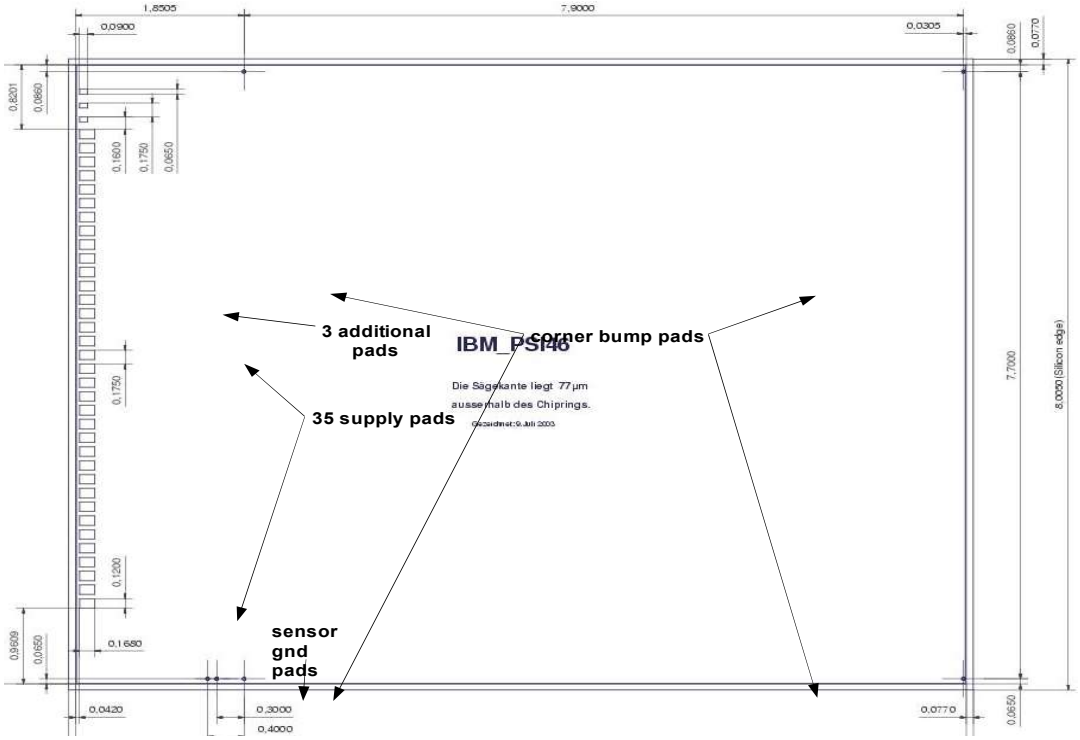


Illustration 2: Physical size of pixel chip.

For data recording, triggering and readout purposes each chip has four counters, the contents of which are available in each double column perifery:

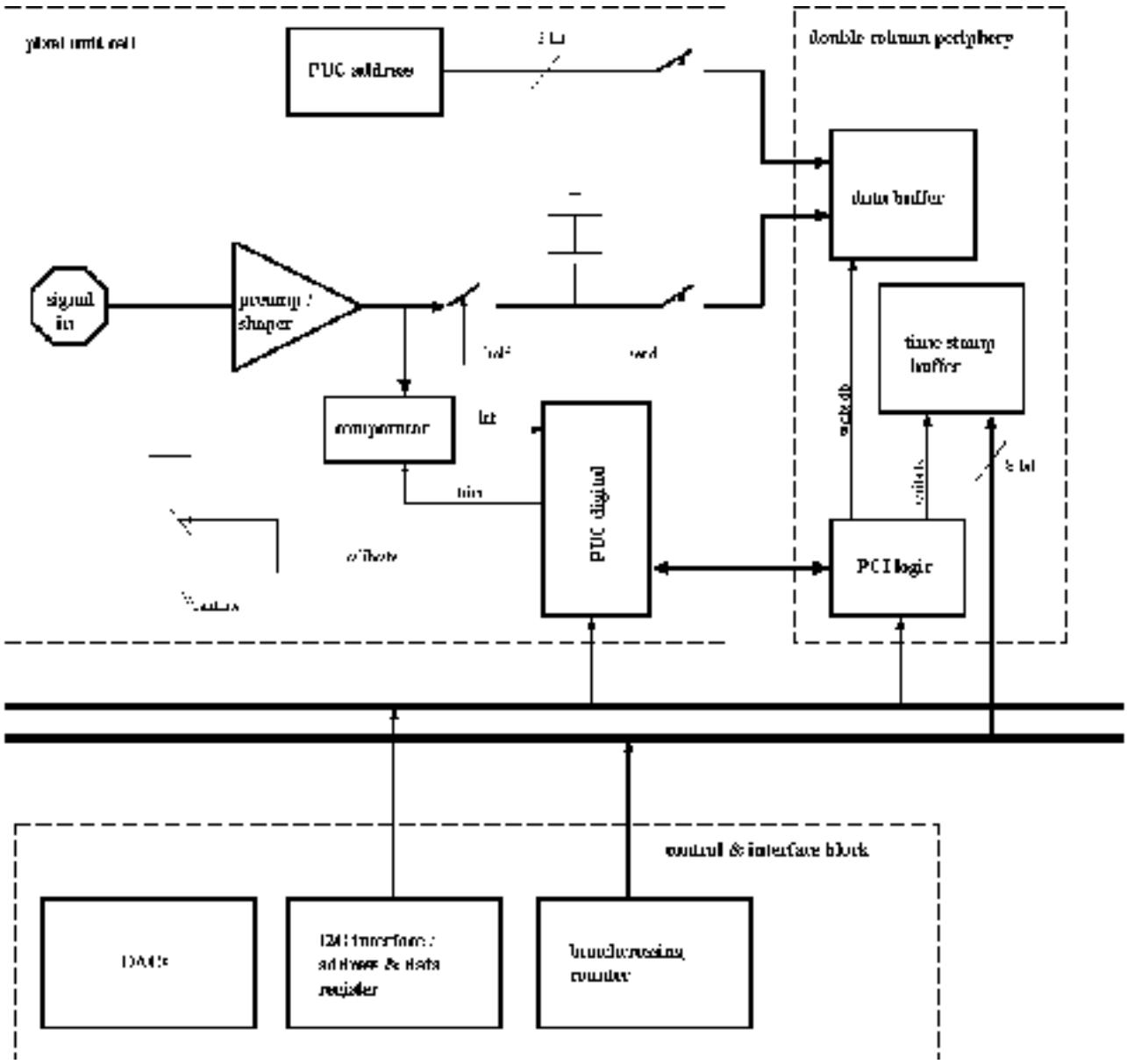
1. Bunch crossing counter, 8 bits. Its content is written as time stamp into a time stamp buffer.
2. Bunch crossing counter with trigger delay, 8 bits. It runs typically 128 counts (selectable, WBC in table 2) behind the bunch crossing counter and is used to validate hits belonging to a trigger.
3. Trigger counter (4 bits) and
4. Token counter (4 bits). They ensure that for each readout token the correct double columns are read out.

For pixel hits to be read out the following consecutive actions must take place:

- The accumulated charge in any of the pixels of a double column must exceed a (programmable) threshold (trim-input to comparator, see illustration 3). Then the corresponding time stamp (bunch crossing number) is written into the time stamp buffer presently pointed at, and the analog signals and the pixel addresses of all hit pixels are written into the next free data buffers (one data buffer per hit pixel). This hit-recording into the time stamp and data buffers runs autonomously and asynchronously in each double column of the chip, independently of the bunch crossing clock. The recorded hit information must be kept in the buffers during the latency time of the first level trigger (3.2µs or 128 bunch crossings).
- Hits in a column are validated by an external level 1 trigger, by comparing their corresponding time stamp with a counter running behind the bunch crossing counter by the trigger delay (illustration 4), otherwise the hits are cleared. If hits in a column are validated, the value of a 4-bit trigger counter is latched into the column perifery. The column is frozen and cannot record further hits until reset after readout of the triggered hits. Other (untriggered) columns remain of course active.
- All frozen columns with the latched value of the trigger counter equal to the present value of the token counter are set to the readout mode. Directly afterwards the trigger counter is incremented (illustration 5). The readout process starts when the token bit enters the chip. After a three bit chip header is sent, each consecutive double column which is in the readout mode is read out and reset. Just before the token leaves the chip the token counter is incremented.

A new token is sent for each trigger and only data belonging to that trigger will be sent onto the readout bus, even if more triggers arrive between a trigger and the corresponding token (see illustration 4).

Illustration 6 shows a flow diagram of the readout process.



llustration 3: Recording of hits in data buffer and time stamp buffer. PUC=Pixel Unit Cell, PCI=Pixel-Column Interface. The details of the handshake mechanism are omitted.

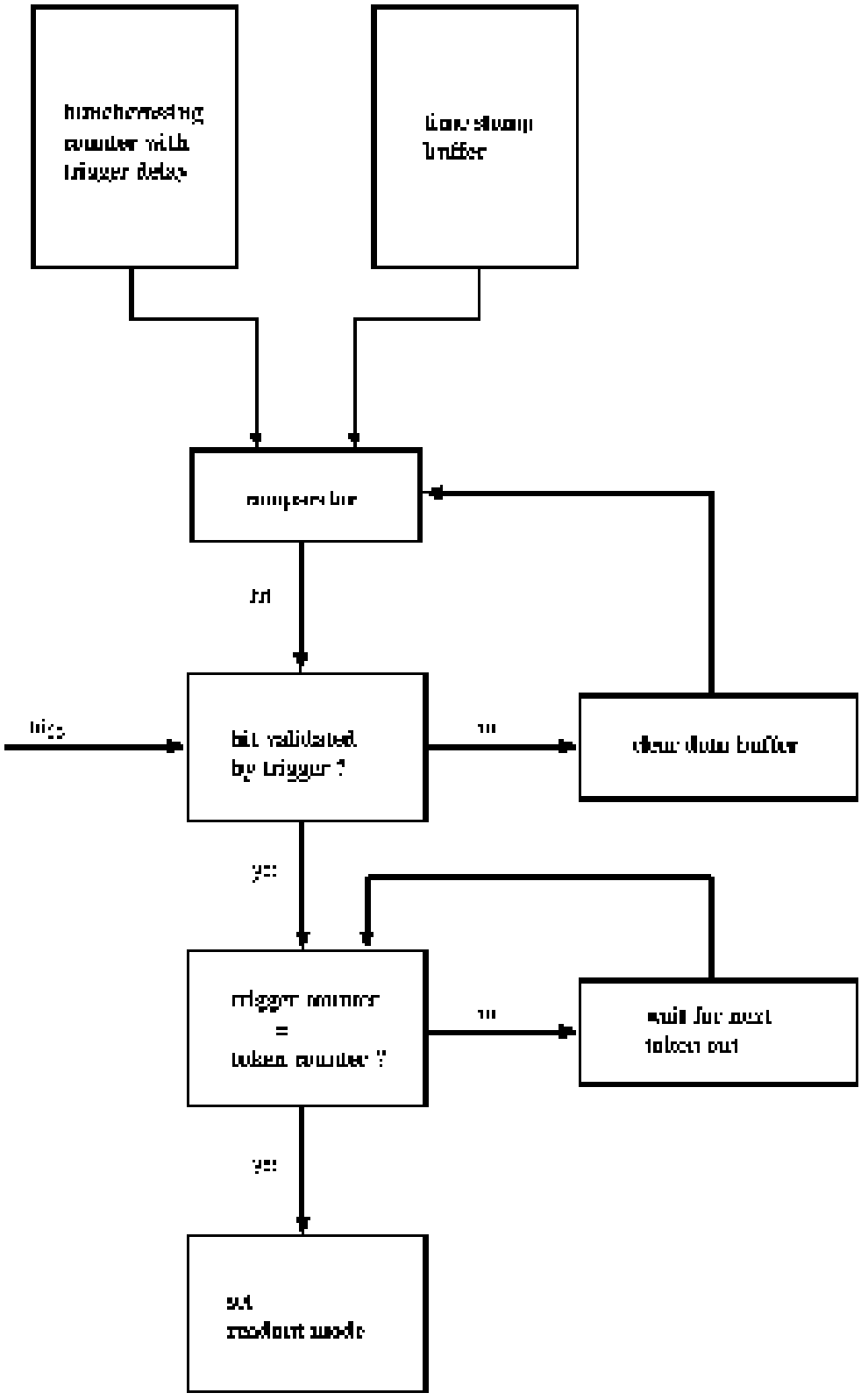


Illustration 4: Trigger validation of a recorded hit

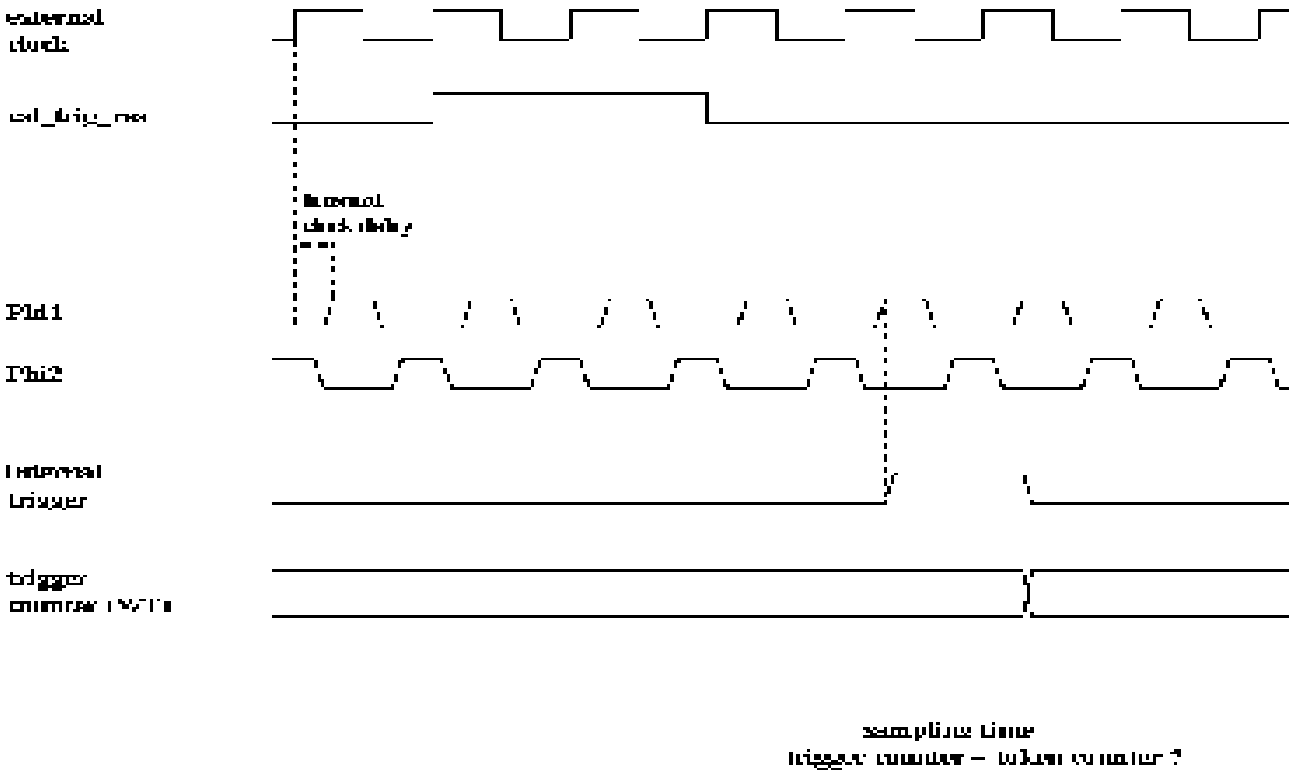


Illustration 5: Timing diagram of the trigger validation

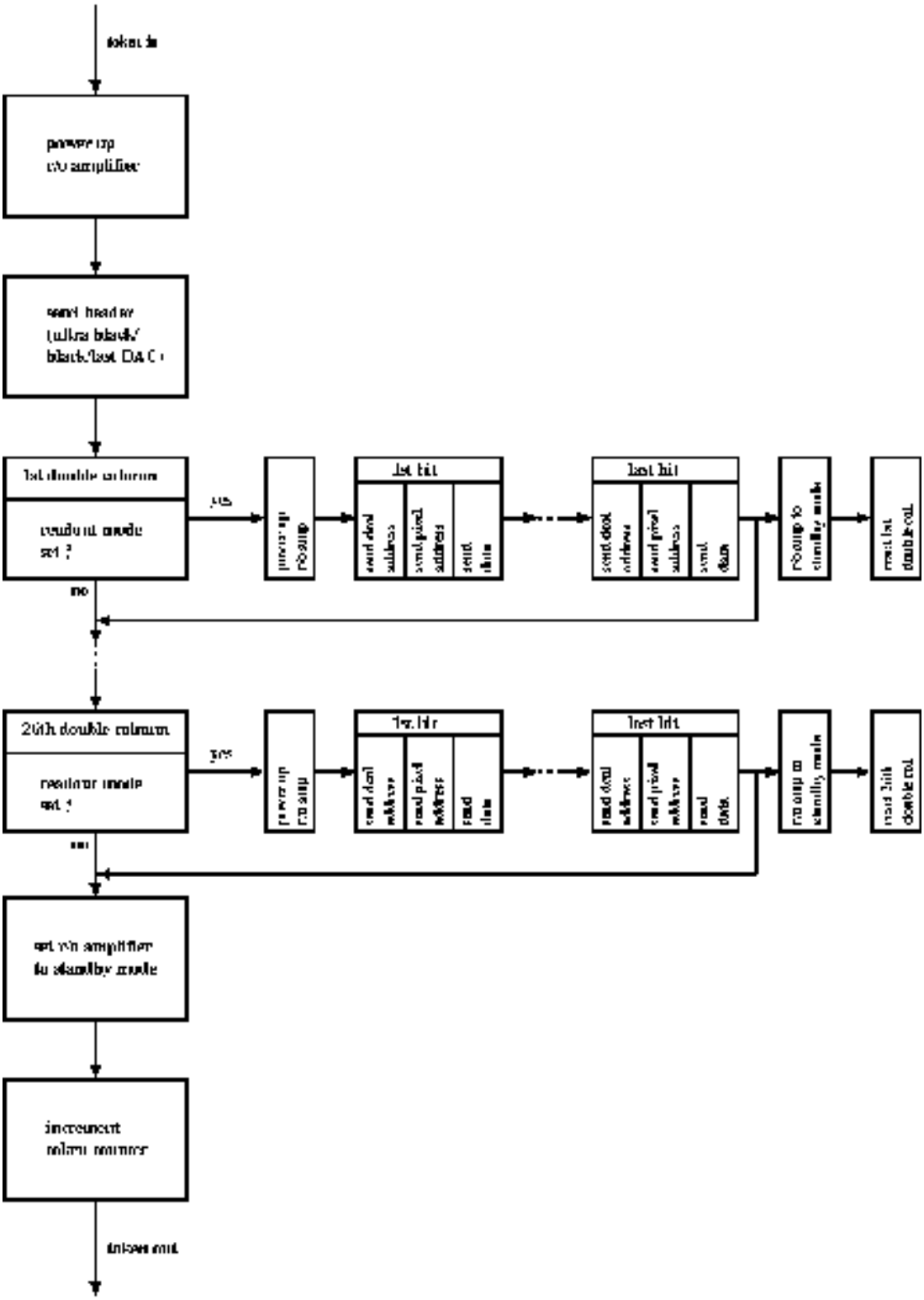


Illustration 6: Readout procedure

2 Pad Layout

Illustration 7 shows the wire bond pads 1-35 and 72 spy-pads. The pads a,b,c have been used in earlier chip versions. They have not been removed in the latest version, but should not be connected anymore. The spy-pads are not specified here since they are only useful for the chip designers. The wire bond pads are described in illustration 7 and table 1.

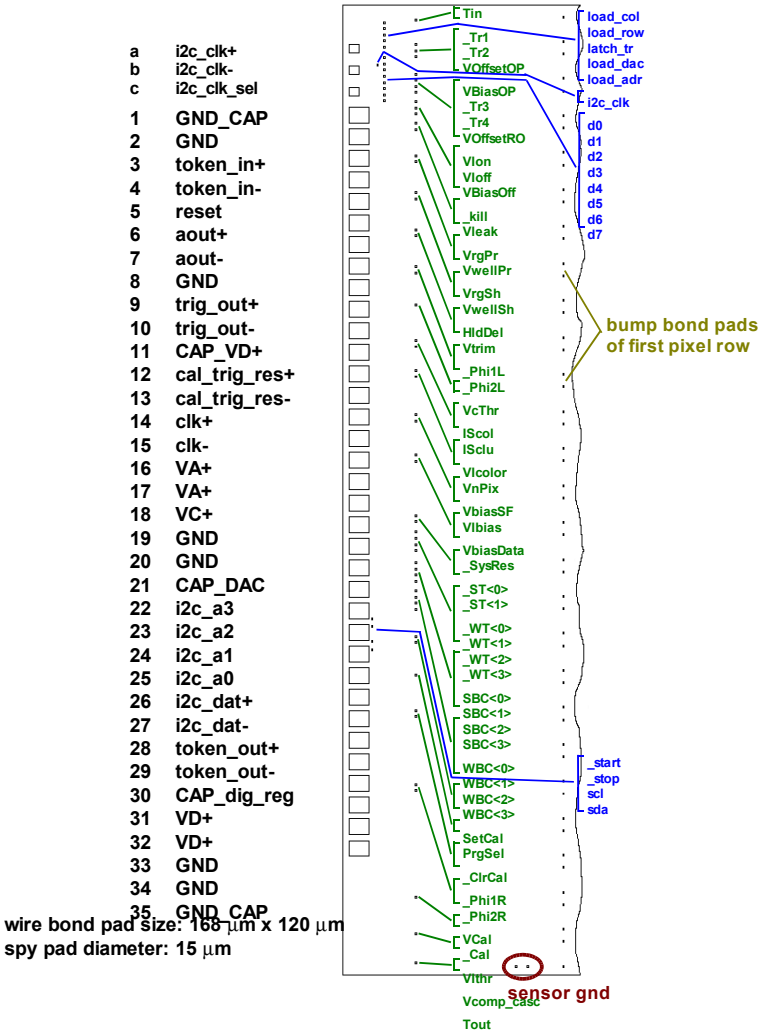


Illustration 7: Wire bond pads and spy pads.

Pin#	Name	Description
1	GND_CAP	Output to external filter capacitances, connected chip internally to ground
2	GND	Ground input
3	token_in +	Readout token input +
4	token_in -	Readout token input -
5	reset	Hard chip reset, resets all double columns, I²C, DAC's
6	aout +	Analogue output +
7	aout -	Analogue output -
8	GND	Ground input, serves as shield between pad 7 and 9
9	trig_out +	Chip multiplicity trigger output + , presently unused
10	trig_out -	Chip multiplicity trigger output - , presently unused
11	CAP_VD +	Output to ext.capacity filtering VD (unregulated), chip internally connected to pad 18
12	cal_trig_res +	Combined input signal: calibrate/trigger/reset +
13	cal_trig_res -	Combined input signal: calibrate/trigger/reset -
14	clk +	40 MHz clock input +
15	clk -	40 MHz clock input -
16	VA +	Analog Voltage +1.5 V input
17	VA +	Analog Voltage +1.5 V input
18	VC +	Input to supply the comparators with +2.5 V (same as VD +). See also pad 11.
19	GND	Ground input
20	GND	Ground input
21	CAP_DAC	Output to ext. capacity filtering the internally regulated voltage supplying the DAC's
22	i2c_a3	Chip address bit 3
23	i2c_a2	Chip address bit 2
24	i2c_a1	Chip address bit 1
25	i2c_a0	Chip address bit 0
26	i2c_dat +	I²C data input SDA +
27	i2c_dat -	I²C data input SDA -
28	token_out +	Readout token input +
29	token_out -	Readout token input -
30	CAP_dig_reg	Output to external capacitance filtering VD regulated
31	VD +	Digital voltage +2.5 V input
32	VD +	Digital voltage +2.5 V input
33	GND	Ground input
34	GND	Ground input
35	GND_CAP	Output to external filter capacitances, connected chip internally to ground

Table 1: List of wire bond pads. Pads 1, 11, 21, 30 and 35 are wire-bonded to external filter capacitances sitting very closely to the chip (base-plate in the case of the barrel).

3 DAC's and Registers

Table 2 lists 26 DAC's and 2 registers. Illustration 8 indicates the actions of some of the DAC's on the analogue signal chain.

Name	addr	unit	# bits	Min Value	Max Value	Recomm. DAC Value	Keyword
Voltage Regulators							
Vdd	1	mV	4	1700	2100	6	Voltage regulator
Vana	2	mV	8	800	1300	140	Voltage regulator
Vsf	3	mV	8	1000	2100	255	Voltage regulator
Vcomp	4	mV	4	1800	2100	15	Voltage regulator
Analog PUC							
Vleak	5	mV	8	-700	0	0	Sensor leakage current compensation
VrgPr	6	mV	4	0	500	0	Preamplifier feedback
VwllPr	7	mV	8	500	1300	35	Preamplifier feedback
VrgSh	8	mV	4	0	500	0	Shaper feedback
VwllSh	9	mV	8	500	1300	35	Shaper feedback
VHldDel	10	mV	8	-1500	-500	117	Hold delay
Vtrim	11	mV	8	-710	-400	29	Pixel trimming
VcThr	12	mV	8	-1500	-600	60	Comparator threshold
Pixel Readout							
VIbias_bus	13	µA	8	0	12	30	
VIbias_sf	14	µA	4	0	50	6	Source follower
Double Column Readout							
VOffsetOp	15	mV	8	1000	1500	90	
VbiasOp	16	µA	8	0	20	115	
VOffsetRO	17	mV	8	1000	1500	76	
VIon	18	µA	8	0	100	115	
Chip Readout							
VIbias_PH	19	µA	8	0	30	100	Pulse height
VIbias_DAC	20	µA	8	0	20	160	Pixel address range
VIbias_roc	21	µA	8	0	30	150	Adjust single ended output leve
Multiplicity Trigger							
VIColOr	22	µA	8	0	200	99	
Vnpix	23	µA	8	0	70	0	
VsumCol	24	µA	8	0	150	0	
Others							
Vcal	25	mV	8	0	260/1800	150	Calibrate pulse height, see also section 5.3.5
CalDel	26	nsec	8	55	205		See chapter 7
WBC	254	clocks	8	0	255		Trigger latency
Chip Control Register	253		8				See section 5.3.5

Table 2: List of DAC's. See also illustration 8.

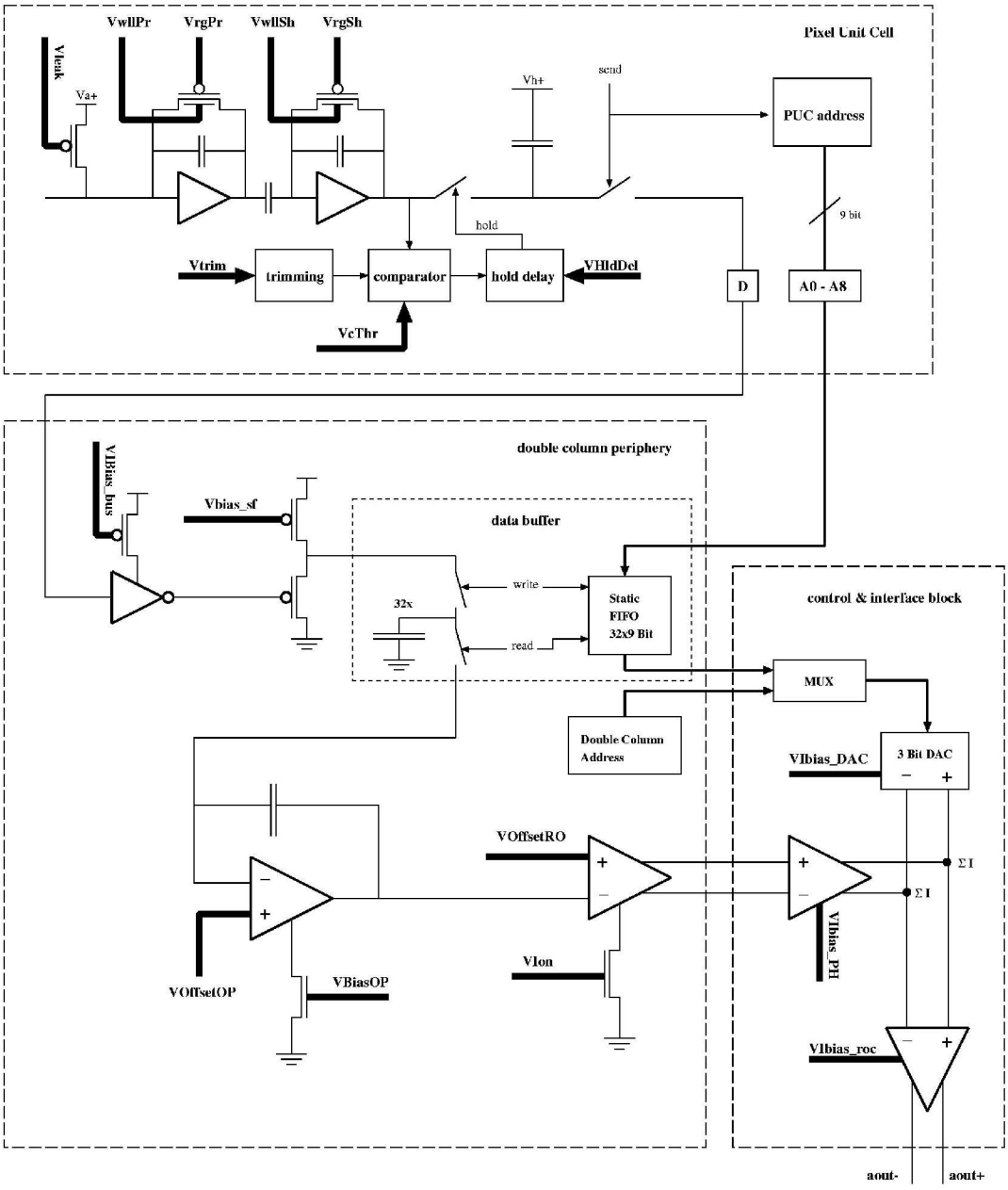


Illustration 8: Actions of some of the DAC's on the analogue signal chain.

4 Temperature Sensor

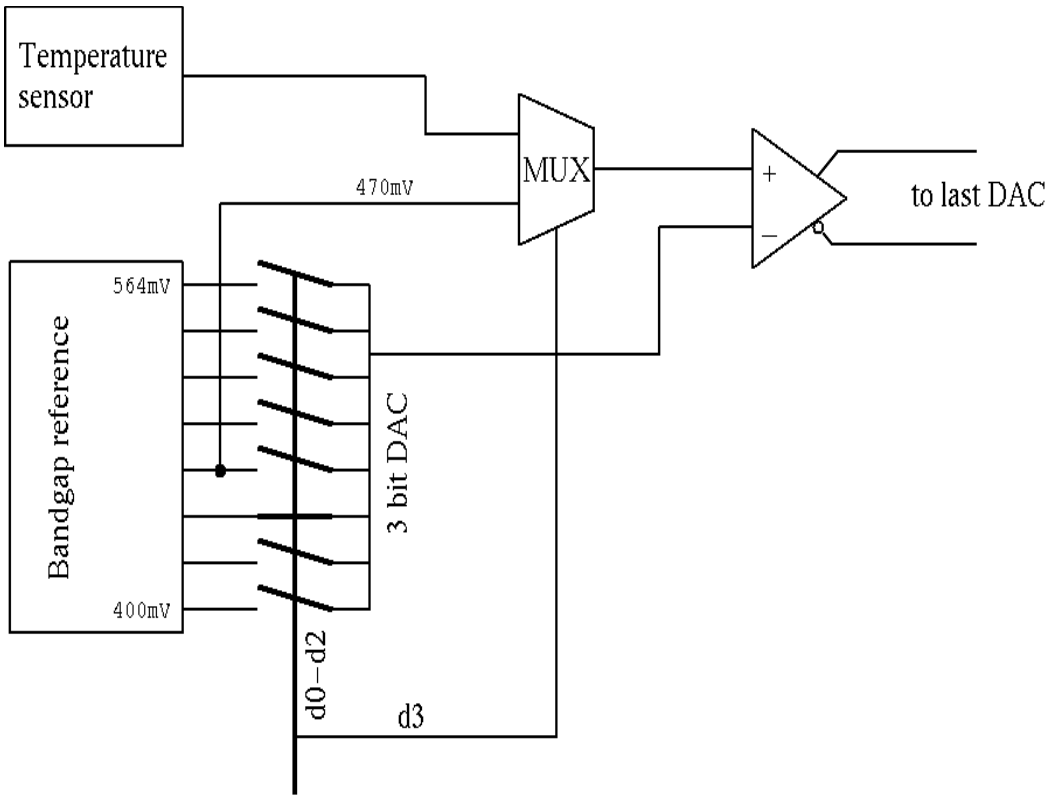
The CMS pixel readout chip has a built in temperature sensor. It consists of a non temperature compensated voltage reference which in fact is optimised to have a large temperature coefficient. This voltage is then compared to an almost temperature independent reference voltage, which is a programmable fraction of the output of the bandgap voltage reference. The amplified voltage difference can be read out in the 3d cycle (last DAC) of the chip readout. For this to happen DAC 27 (0x1B) had to be addressed last (see below).

The sensor is designed to measure temperatures in the range [-30ºC, +70ºC]. To reach a better accuracy, the gain has been chosen such that not the full range of temperatures can be measured with the same reference voltage. Instead one has to program the appropriate voltage using a 3 bit DAC. This defines 8 different windows of measurement. For each temperature the output voltage is in the valid range for at least 2 consecutive windows. A voltage limitation ensures that there is no output level below 0 (black level).

In order to calibrate the temperature reading, the temperature sensor output voltage can be replaced by a fixed, temperature independent voltage. This happens through a multiplexer controlled by the 4h data bit of the afore mentioned DAC.

Summary:

- Address of the DAC: 27 (0x1B)
- The first 3 data bits set the reference voltage in the range [399.5mV, 564mV] in steps of 23.5mV (unirradiated).
- Bit 4 switches between the temperature sensor output voltage (bit cleared) and a fixed voltage of 470mV (bit set).



5 Programming of the PSI46

The PSI46 can be programmed via a 'fast I²C' interface.

To allow high speed programming (>40MHz) some features of the standard I²C interface are removed:

- No readback of data.
- 10 bit data structure instead of 9 bit to prevent long periods where SDA is not changing.
- No acknowledge bit after data byte, the 4th and 10th bits are ignored.
- Only 1 master allowed.

5.1 Fast I²C external interface

Since the I²C interface does not require readback, no tristate or open drain pins are needed.

LVDS (Low Voltage differential signaling) input drivers are used for SDA.

SCL (separate I²C clock, which was still used in PSI43) is not available on PSI46, instead the chip clock is used. This also implies that the **PSI46 cannot be programmed via slow I/O** like the PSI43.

Fast I²C works similar to standard I²C interface, a change of SDA when the clock is '1' is a start (H->L) or a stop (L->H) condition. The state of SDA at the rising edge of CLK determines the data to be shifted in.

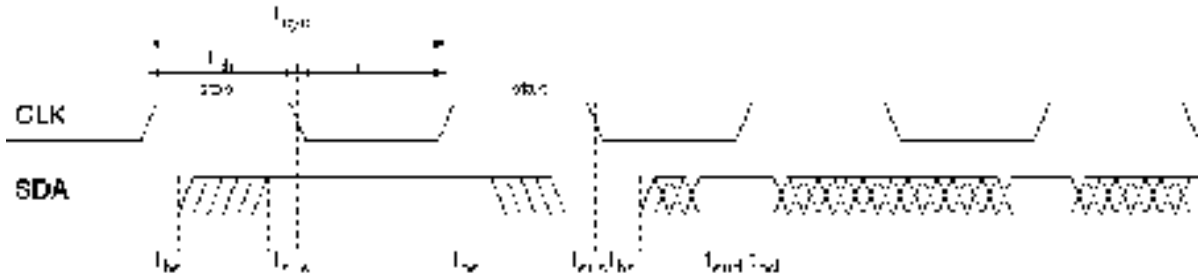


Illustration 9: fast I²C timing diagram

parameter	description	value
T_{cyc}	Min cycle time	25 ns
T_{ch}	Min clock high	10 ns
T_{cl}	Min clock low	10 ns
T_{hs}	Hold time SDA start or stop after CLK	1 ns
T_{sus}	Setup time SDA start or stop before CLK	3 ns
T_{hd}	Hold time SDA data after CLK	1 ns
T_{sud}	Setup time SDA data before CLK	3 ns

Table 3: Fast I²C AC spec.

5.2 Data format

All data sent to the PSI46 via I²C in a 10 bit format. The 4th and 10th bit are inverted to the previous bits to prevent too many consecutive 0s or 1s.

All data are sent in bytes, the first bit of the digital data stream is the MSB. Usage of start and stop bits is equivalent to standard I²C.

Between commands referring to the same device no stop bit is required, a single start bit to indicate a new command is sufficient.

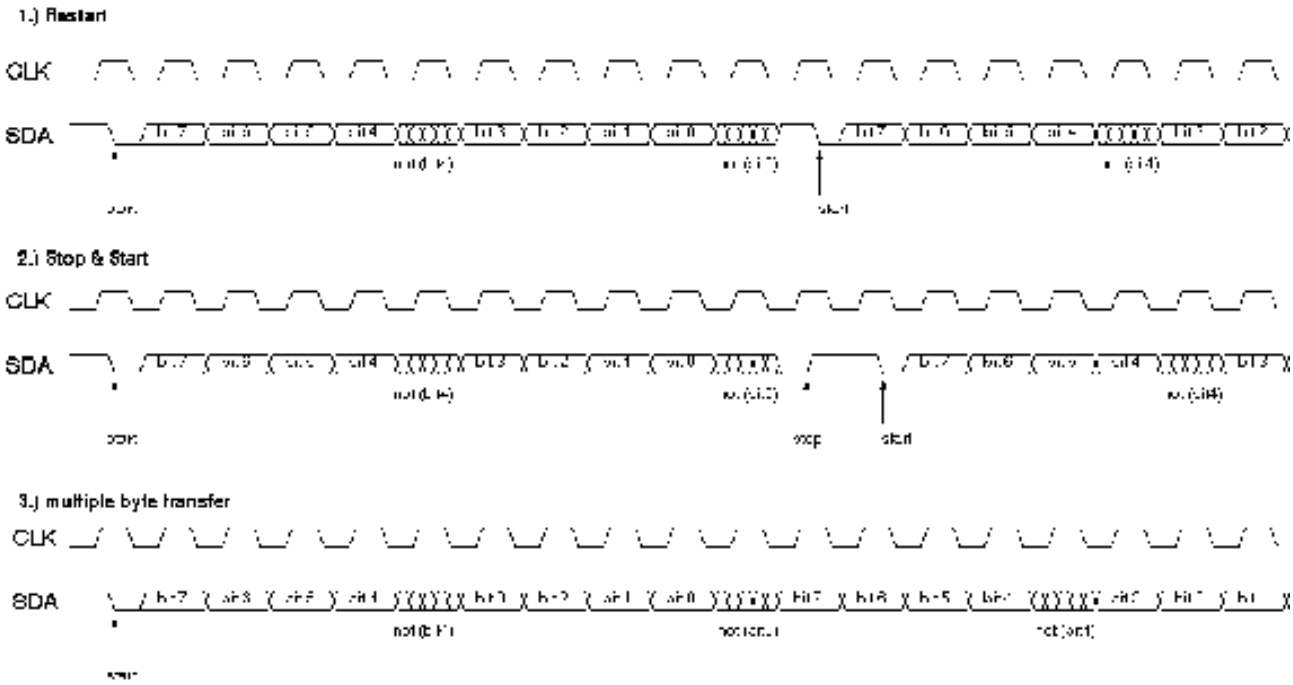


Illustration 10: fast I²C byte structure

- A change of SDA while CLK is high depicts either a start (L->H) or a stop (H->L) command.
- Data are always latched at the rising edge of CLK.
- The 4th and 10th bit has to be sent (usually the inverted previous bit) but is ignored by the receiver.
- No data *output* is generated.

5.3 I²C commands

In the following section a byte is always 8bit, the two synchronisation bits are not mentioned.

5.3.1 Commands and data structure

The first byte transmitted after a start is the command byte.

The following commands are available for I²C:

Command	Byte 1	Byte 2	Byte 3	Comment
Prog_DAC	DAC address	DAC value		DAC programming, see Table 2
ClrCal				Removes calibrate mode from all pixels
Prog_Pix	ColAdr	RowAdr	Data	Pixel programming, see 5.3.3.
Cal_Pix	ColAdr	RowAdr	Data	Calibrate signal into pixels, see

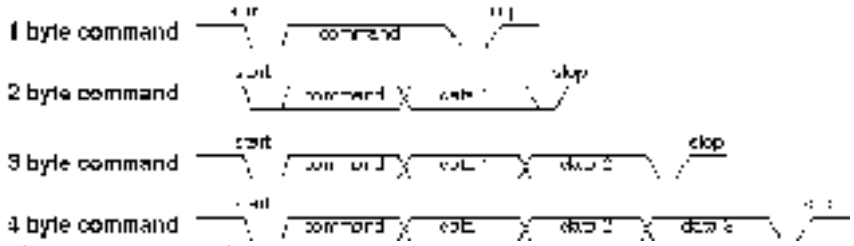


Illustration 11: Command structure

5.3.2 Command decoding

There are only four commands required:

	7	6	5	4	3	2	1	0
Prog_DAC	Chip address				1	0	0	0
Prog_Pix	Chip address				0	1	0	0
Cal_Pix	Chip address				0	0	1	0
ClrCal	Chip address				0	0	0	1

The Chip address is defined by wire bonding the pads 22-25 (see illustration 7). An address bit with no external connection is zero. To set bit 1 the address pad should be wire bonded to ground. When all address pads are left unconnected, the chip address is therefore 0 and the Prog_DAC command would be 0x08.

5.3.3 Programming of pixel addresses

A pixel is identified by row (0-79) and column (0-51). The column and row addresses have to be programmed using the gray code (e.g row0 = 0; row1 = 1; row2=3; row3=2;...., see the gray code table in Appendix 1).

- Row addresses
  are simply gray coded
- Column addresses
  Programming column addresses involves (single) column numbers (0-51), unlike in the readout, where double column numbers (0-25) are used. However, coding a (single) column address uses the gray coded double column address to which the column belongs; the distinction between left and right column of a

double column is done with the LSB. Starting from a given column number col_num, its address is obtained as follows:
(gray[col_num>>1] << 1) | (col_num & 0x01)
where [col_num >> 1] equals the address of the double column. This is put into gray code according to the table in appendix 1, shifted left to go back to column numbers, and the LSB is set or not, depending on whether col_num belongs to a right (odd) or left (even) column.

Example: Column address (col_num) = 27 (0x1B = b0011011)

0x1B (b00011011) -> DCol 0x0D (b00001101) // change to DCol address by shifting right
0x0D (b00001101) -> gray 0x0B (b00001011) // convert to graycode
0x0B (b00001011) -> 0x16 (b00010110) // shift left to switch from DCol to column addressing
0x16 (b00010110) | (0x1B & 0x01) -> 0x17 // since it was the right (odd) column, the LSB is set.
In other words, the DCol is gray coded, the left column adds a '0', the right column adds a '1' at the end.

- Notes for calibration of pixels
  The calibration is set for every crossing of programmed columns and rows, i.e. when you choose e.g. Pixel 0/0 and Pixel 2/2 for calibrate mode you implicitly also calibrated Pixel 0/2 and Pixel 2/0.

5.3.4 Structure of the data bytes for Prog_Pix and Cal_Pix commands

Prog_Pix

7	6	5	4	3	2	1	0
E				T3	T2	T1	T0

E = Enable_pixel bit : '1' = enable, '0'= mask
T3...T0 = trim bits. For T3T2T1T0=1111 the pixel is least sensitive.

Cal_Pix

7	6	5	4	3	2	1	0
						Cb	Cd

Cb = calibrate via sensor bumps
Cd = calibrate directly via calibrate capacitor

5.3.5 Control Registers

- Chip Control Register
  can be programmed like a DAC
  Addr = 0xFD
  bit0: readout speed
  if bit0 is set, the readout is in half_speed_mode (20MHz)
  cleared: full_speed_mode
  bit1: Disable chip
  if bit1 is set, all pixels and double_columns are disabled
  bit2: range of calibrate signal
  if bit2 is set: 0-1800mV, otherwise 0-280mV

  Example: 0x08 0xFD 0x01
  |____|____|_____ 0x08 : Program DAC at chip #0
  |____|_____ 0xFD : Chip Control register
  |_____ 0x01 : Half Speed Mode

- Double Column Control Register
  It enables / disables double columns.
  It is programmed with the command Prog_Pix (see 5.3.1.).
  As column address use the gray-coded address of the left (even) column belonging to the double column.
  As row address use the special “row address” 0x40 (graycode of 127).
  If data is 0x80, this double column is enabled, if data is 0, it is disabled.
  Example: 0x34 0x0C 0x40 0x80
  |____|____|____|_____ 0x34 : Program Pixel at chip #3
  |____|____|_____ 0x0C : DCol 4 (Column 8,9: DCol Ctrl at column 8, gray(8) = 12)
  |____|_____ 0x40 : Special “row address” of DCol control register
  |_____ 0x80 : enable this DCol

- WBC
  The WBC register is programmed like a DAC. It must correspond to the latency time of the trigger in numbers of bunch crossings.
  Address: 0xFE
  The new WBC is only valid after the next reset.
  Example: 0x08 0xFE 0x10
  |____|____|_____ 0x08 : Program DAC at chip #0
  |____|_____ 0xFE : WBC register
  |_____ 0x10 : latency = 16 (400 ns).

5.3.6 Multiple Pixel programming

If download times have to be minimized, a multiple programming of pixels within the same double column is possible. The structure of these commands are:

Prog_Pix - ColAdr - RowAdr - Data - RowAdr - Data - RowAdr - Data - RowAdr - Data ...

or:

Cal_Pix - ColAdr - RowAdr - Data - RowAdr - Data - RowAdr - Data - RowAdr - Data ...

Multiple programming is not available for ClrCal and Prog_DAC commands.

5.3.7 Examples

The following example shows all commands, all bytes are given in hex,
the 4th and 10th bit of the data stream as well as start and stop bits are not printed.

a = chip address.

- 0xa8 0x01 0x08 — DAC #1 = 8 (vdd = ½ of full scale of a 4bit DAC)
- 0xa8 0x02 0x80 — DAC #2 = 128 (Vana = ½ of full scale of a 16bit DAC)
- 0xa8 0xFE 0x12 — WBC = 18
- 0xa8 0xFD 0x00 — Chip Control register: enable & full speed
- 0xa4 0x00 0x40 0x80 — enable dcol 0
- 0xa4 0x02 0x40 0x80 — enable dcol 1 (dcol=1 -> col=2 -> 2=3(gray)
  -> 3 & 0xFE = 2 (left column of dcol)
- 0xa4 0x06 0x40 0x80 — enable dcol2 (dcol=2 -> col=4 -> 4=6(gray))

- 0xa4 0x02 0x00 0x0F — mask pixel 0 in dcol 1 (right side) = column 3
  3 = 2 (gray), 0x0F: MSB not set -> Pixel masked.
- 0xa4 0x02 0x12 0x8F — trim Pixel 28 in column 3
  28 = 0x12(gray).
  0x8F : MSB set -> enable Pixel. all 4 LSBs set -> Maximum trimming (least sensitive setting).
- 0xa1 — Clear Cal.
- 0xa2 0x02 0x12 0x01 — Calibrate Pixel 28 in column 3
  Calibrate via the direct calibration capacitance.
- 0xa2 0x02 0x06 0x01 0x07 0x01 0x05 0x02 0x04 0x02
  Also Calibrate in column 3 :
  Pixel 4 (gray=6) and Pixel 5 (gray=7) with direct cal,
  Pixel 6 (gray=5) and Pixel 7 (gray=4) with sensor cal.

6 Readout

6.1 Analog readout

The PSI46 has an analog serial readout. The output is a differential open collector stage. The outputpads (aout+/aout-) should be connected via pull-down resistors to the negative supply voltage (GNDD). Multiple daisy chained readout chips are connected to a common bus with a single pull-down for each of the two differential signals. The value of the external resistors determines the size of the signal swing. Table 4 gives approximate output levels into 200 Ohm pull-down resistors (for VIBias_roc=15uA).

Level	**aout+**	**aout-**	**differential**
UltraBlack	60 mV	140 mV	-80 mV
Black	100 mV	100 mV	0 mV
Data/Addresses	80 mV..150 mV	60mV .. 120 mV	-40 .. 90 mV

Table 4: Approximate output levels

The total current of the output stage is determined by VIBias_roc and will be approximately 35 times higher. The gain of the output stage is fixed and not affected by VIBias_roc. Higher values will add a DC offset to the output while very low values will result in an insufficient output range.

The output amplifier of the readout chips that do not have the readout token are not tri-stated but switched to a reduced current. This will add a small (few mV/chip) DC offset to both differential lines.

The following internal bias voltage affect the readout levels.

- VoffsetOP adds a dc offset to the pixel data. The gain is fixed.
- VIbias_DAC scales the values of the addresses and UB
- VoffsetRO adds a DC offset to the pixel data

6.2 Readout sequence

A readout is started when a token is sensed during the rising edge of CLK. A new token must be sent for each trigger and only data belonging to that trigger will be sent onto the readout bus, even if more triggers arrive between a trigger and the corresponding token (see illustration 4).

Tokens are validated a few ns after the rising edge of the clock and the readout starts immediately (illustration 7).

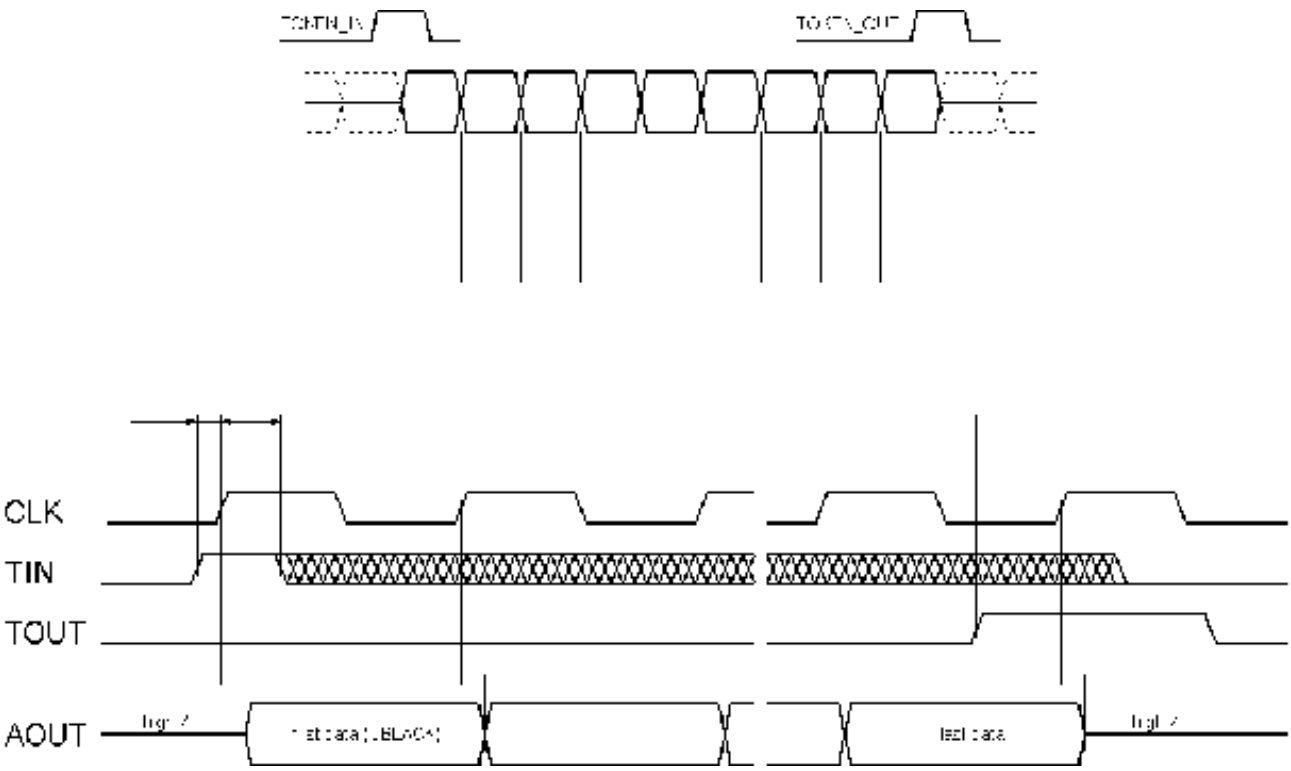


Illustration 12: PSI46 readout sequence

The readout chip will always send UltraBlack, Black and "last DAC", where "last DAC" represents the value of the most recently programmed DAC. If any of the columns of the readout chip has data for the current token/trigger it will be sent onto the readout bus in the following format which is repeated for each pixel with a hit (see illustration 4):

Cycle	Signal	description
1	C1	Double column address MSB (6 levels)
2	C0	Double column address LSB (6 levels)
3	A2	Pixel address MSB (6 levels)
4	A1	Pixel address NMSB (6 levels)
5	A0	Pixel address LSB (6 levels)
6	D	Pulse height (analog)

The readout order of pixels in a double column is given by the column drain mechanism. It starts in the left column with the pixel closest to the periphery, goes up the left column, comes back down the right column and ends with the last pixel in the right double column. The analog levels A0, A1,A2 of each pixel are given in appendix 2. Note that A2 never assumes the level 5 in appendix 2. The analog levels C0, C1 of each double column are also given in appendix 2.

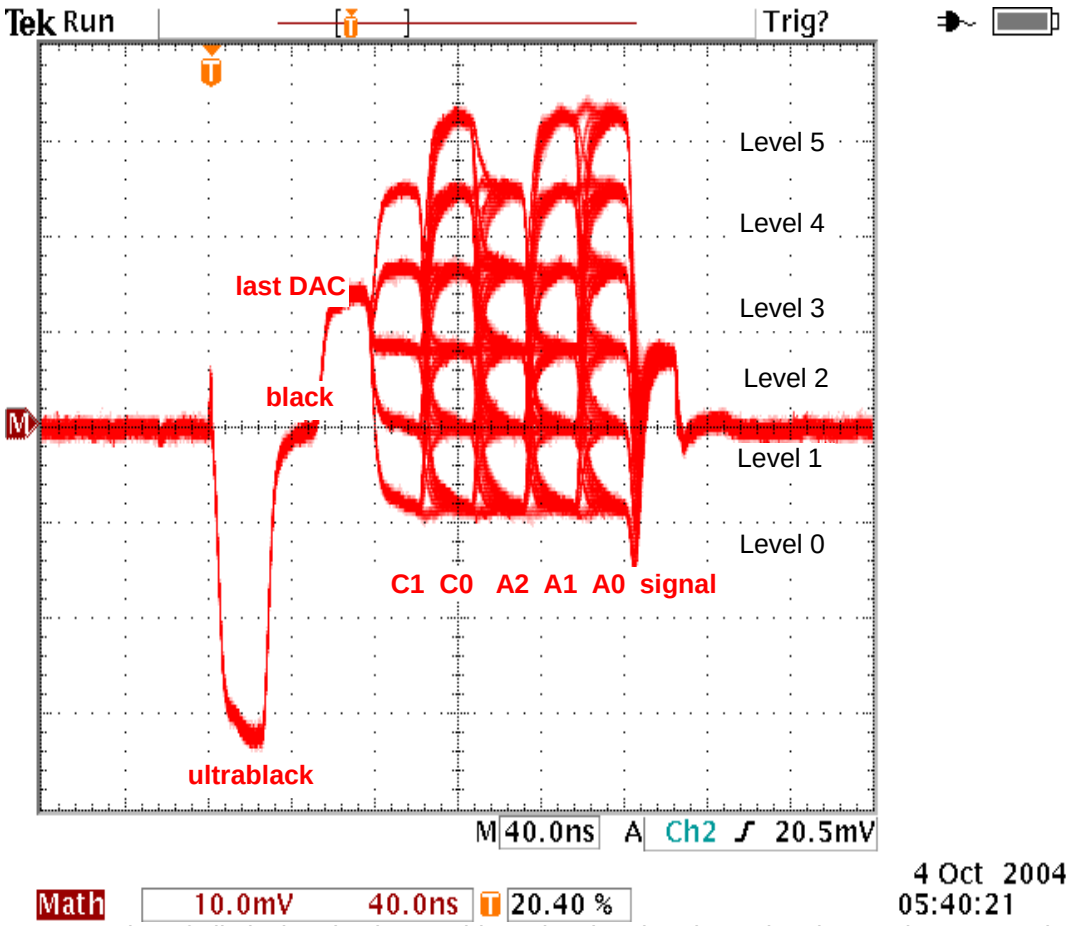


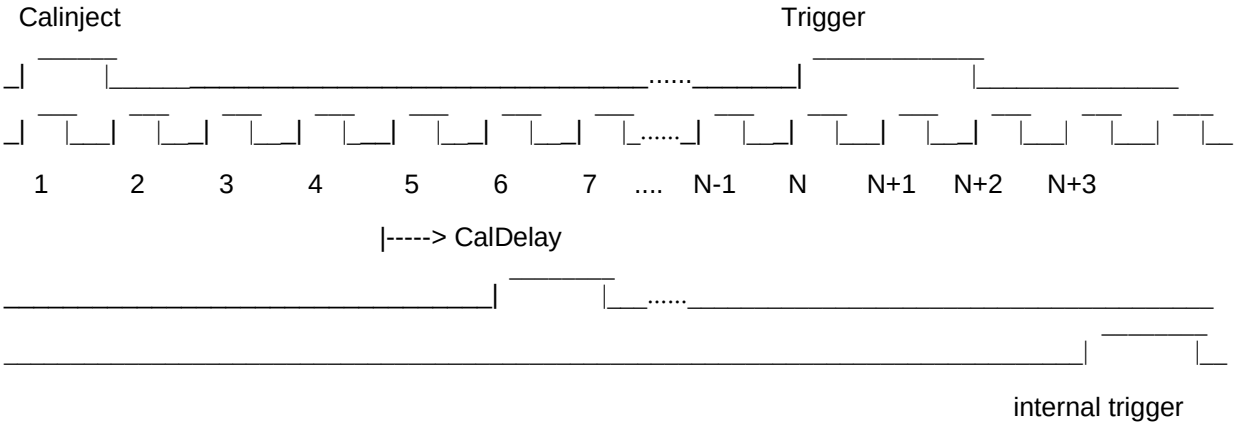
Illustration 13: Overlay of all pixel and column address levels. The picture has been taken at 40 Mhz readout speed with a PSI46 (version 1). The DC-levels of PSI46 (version 2) will be according to Table 4. The address bits are preceeded by ultrablack/black/last DAC value.

7 Calibration

The internally generated Calibration pulse appears 5 clock cycles plus an adjustable delay after the first clock registering the CalTrigReset signal. The additional delay is controlled by the CalDel DAC. It is approximately

CalDelay ~ 0.32 ns (256 - DAC) + 30 ns.

The bunch crossing number will be latched on the next rising edge of the clock. With the trigger signal starting at clock cycle N, the internal trigger is generated on N+3.



The calibration signal is accepted by the trigger if N+3 = 5 + CalDelay/TBC + WBC, where

TBC is the time between collisions and WBC is the trigger latency setting to be downloaded via I²C.

WBC = N -2 - CalDelay/TBC

The CalDelay setting is unreliable for very small delays (large DAC values, >150).

Appendix 1: Gray code table

dec	hex	bin	gray(dec)	gray(hex)	gray(bin)
0	0x00	b00000000	0	0x00	b00000000
1	0x01	b00000001	1	0x01	b00000001
2	0x02	b00000010	3	0x03	b00000011
3	0x03	b00000011	2	0x02	b00000010
4	0x04	b00000100	6	0x06	b00000110
5	0x05	b00000101	7	0x07	b00000111
6	0x06	b00000110	5	0x05	b00000101
7	0x07	b00000111	4	0x04	b00000100
8	0x08	b00001000	12	0x0C	b00001100
9	0x09	b00001001	13	0x0D	b00001101
10	0x0A	b00001010	15	0x0F	b00001111
11	0x0B	b00001011	14	0x0E	b00001110
12	0x0C	b00001100	10	0x0A	b00001010
13	0x0D	b00001101	11	0x0B	b00001011
14	0x0E	b00001110	9	0x09	b00001001
15	0x0F	b00001111	8	0x08	b00001000
16	0x10	b00010000	24	0x18	b00011000
17	0x11	b00010001	25	0x19	b00011001
18	0x12	b00010010	27	0x1B	b00011011
19	0x13	b00010011	26	0x1A	b00011010
20	0x14	b00010100	30	0x1E	b00011110
21	0x15	b00010101	31	0x1F	b00011111
22	0x16	b00010110	29	0x1D	b00011101
23	0x17	b00010111	28	0x1C	b00011100
24	0x18	b00011000	20	0x14	b00010100
25	0x19	b00011001	21	0x15	b00010101
26	0x1A	b00011010	23	0x17	b00010111
27	0x1B	b00011011	22	0x16	b00010110
28	0x1C	b00011100	18	0x12	b00010010
29	0x1D	b00011101	19	0x13	b00010011
30	0x1E	b00011110	17	0x11	b00010001
31	0x1F	b00011111	16	0x10	b00010000
32	0x20	b00100000	48	0x30	b00110000
33	0x21	b00100001	49	0x31	b00110001
34	0x22	b00100010	51	0x33	b00110011
35	0x23	b00100011	50	0x32	b00110010
36	0x24	b00100100	54	0x36	b00110110
37	0x25	b00100101	55	0x37	b00110111
38	0x26	b00100110	53	0x35	b00110101
39	0x27	b00100111	52	0x34	b00110100

dec	hex	bin	gray(dec)	gray(hex)	gray(bin)
40	0x28	b00101000	60	0x3C	b00111100
41	0x29	b00101001	61	0x3D	b00111101
42	0x2A	b00101010	63	0x3F	b00111111
43	0x2B	b00101011	62	0x3E	b00111110
44	0x2C	b00101100	58	0x3A	b00111010
45	0x2D	b00101101	59	0x3B	b00111011
46	0x2E	b00101110	57	0x39	b00111001
47	0x2F	b00101111	56	0x38	b00111000
48	0x30	b00110000	40	0x28	b00101000
49	0x31	b00110001	41	0x29	b00101001
50	0x32	b00110010	43	0x2B	b00101011
51	0x33	b00110011	42	0x2A	b00101010
52	0x34	b00110100	46	0x2E	b00101110
53	0x35	b00110101	47	0x2F	b00101111
54	0x36	b00110110	45	0x2D	b00101101
55	0x37	b00110111	44	0x2C	b00101100
56	0x38	b00111000	36	0x24	b00100100
57	0x39	b00111001	37	0x25	b00100101
58	0x3A	b00111010	39	0x27	b00100111
59	0x3B	b00111011	38	0x26	b00100110
60	0x3C	b00111100	34	0x22	b00100010
61	0x3D	b00111101	35	0x23	b00100011
62	0x3E	b00111110	33	0x21	b00100001
63	0x3F	b00111111	32	0x20	b00100000
64	0x40	b01000000	96	0x60	b01100000
65	0x41	b01000001	97	0x61	b01100001
66	0x42	b01000010	99	0x63	b01100011
67	0x43	b01000011	98	0x62	b01100010
68	0x44	b01000100	102	0x66	b01100110
69	0x45	b01000101	103	0x67	b01100111
70	0x46	b01000110	101	0x65	b01100101
71	0x47	b01000111	100	0x64	b01100100
72	0x48	b01001000	108	0x6C	b01101100
73	0x49	b01001001	109	0x6D	b01101101
74	0x4A	b01001010	111	0x6F	b01101111
75	0x4B	b01001011	110	0x6E	b01101110
76	0x4C	b01001100	106	0x6A	b01101010
77	0x4D	b01001101	107	0x6B	b01101011
78	0x4E	b01001110	105	0x69	b01101001
79	0x4F	b01001111	104	0x68	b01101000

Appendix 2: Pixel and double Column Address Analog Levels

Pixel	Address level		Pixel	Address level	
	Left	Right		Left	Right
0	4 2 4	4 2 5	40	2 1 2	2 1 3
1	4 2 2	4 2 3	41	2 1 0	2 1 1
2	4 2 0	4 2 1	42	2 0 4	2 0 5
3	4 1 4	4 1 5	43	2 0 2	2 0 3
4	4 1 2	4 1 3	44	2 0 0	2 0 1
5	4 1 0	4 1 1	45	1 5 4	1 5 5
6	4 0 4	4 0 5	46	1 5 2	1 5 3
7	4 0 2	4 0 3	47	1 5 0	1 5 1
8	4 0 0	4 0 1	48	1 4 4	1 4 5
9	3 5 4	3 5 5	49	1 4 2	1 4 3
10	3 5 2	3 5 3	50	1 4 0	1 4 1
11	3 5 0	3 5 1	51	1 3 4	1 3 5
12	3 4 4	3 4 5	52	1 3 2	1 3 3
13	3 4 2	3 4 3	53	1 3 0	1 3 1
14	3 4 0	3 4 1	54	1 2 4	1 2 5
15	3 3 4	3 3 5	55	1 2 2	1 2 3
16	3 3 2	3 3 3	56	1 2 0	1 2 1
17	3 3 0	3 3 1	57	1 1 4	1 1 5
18	3 2 4	3 2 5	58	1 1 2	1 1 3
19	3 2 2	3 2 3	59	1 1 0	1 1 1
20	3 2 0	3 2 1	60	1 0 4	1 0 5
21	3 1 4	3 1 5	61	1 0 2	1 0 3
22	3 1 2	3 1 3	62	1 0 0	1 0 1
23	3 1 0	3 1 1	63	0 5 4	0 5 5
24	3 0 4	3 0 5	64	0 5 2	0 5 3
25	3 0 2	3 0 3	65	0 5 0	0 5 1
26	3 0 0	3 0 1	66	0 4 4	0 4 5
27	2 5 4	2 5 5	67	0 4 2	0 4 3
28	2 5 2	2 5 3	68	0 4 0	0 4 1
29	2 5 0	2 5 1	69	0 3 4	0 3 5
30	2 4 4	2 4 5	70	0 3 2	0 3 3
31	2 4 2	2 4 3	71	0 3 0	0 3 1
32	2 4 0	2 4 1	72	0 2 4	0 2 5
33	2 3 4	2 3 5	73	0 2 2	0 2 3
34	2 3 2	2 3 3	74	0 2 0	0 2 1
35	2 3 0	2 3 1	75	0 1 4	0 1 5
36	2 2 4	2 2 5	76	0 1 2	0 1 3
37	2 2 2	2 2 3	77	0 1 0	0 1 1
38	2 2 0	2 2 1	78	0 0 4	0 0 5
39	2 1 4	2 1 5	79	0 0 2	0 0 3

DCOL	Address Levels
0	0 0
1	0 1
2	0 2
3	0 3
4	0 4
5	0 5
6	1 0
7	1 1
8	1 2
9	1 3
10	1 4
11	1 5
12	2 0
13	2 1
14	2 2
15	2 3
16	2 4
17	2 5
18	3 0
19	3 1
20	3 2
21	3 3
22	3 4
23	3 5
24	4 0
25	4 1