PS08-beta

Single-chip Solution for Weight Scales

Preliminary Datasheet

Beta-version
01. October 2007

acam-messelectronic gmbh
solutions in time

Inhaltsverzeichnis

Important Note:
This datasheet is a preliminary version based on first engineering samples. It is not intended to be complete or correct in any detail. There might be major changes for the final version.

1 System Overview

1.1 Introduction

The PS08 is an ultra-low-power SoC (System-on-Chip) especially designed for weight scales. It combines the performance of a 24-Bit microprocessor with the great advantages of the PICOSTRAIN measuring principle. Due to this the current consumption of the total system, including the load cell itself, can be reduced to a so far unknown level. The PS08 is perfectly suited for battery driven or solar cell driven weight scales. The precision of the device allows to build even battery driven calibrated scales.

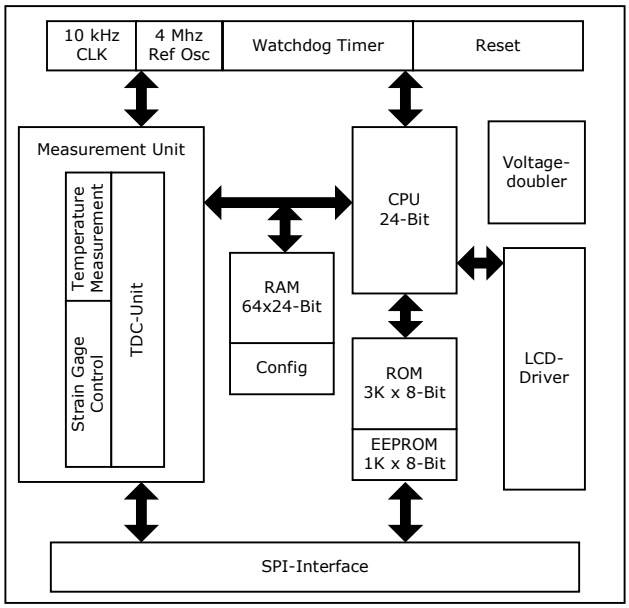


Figure 1 System Architecture

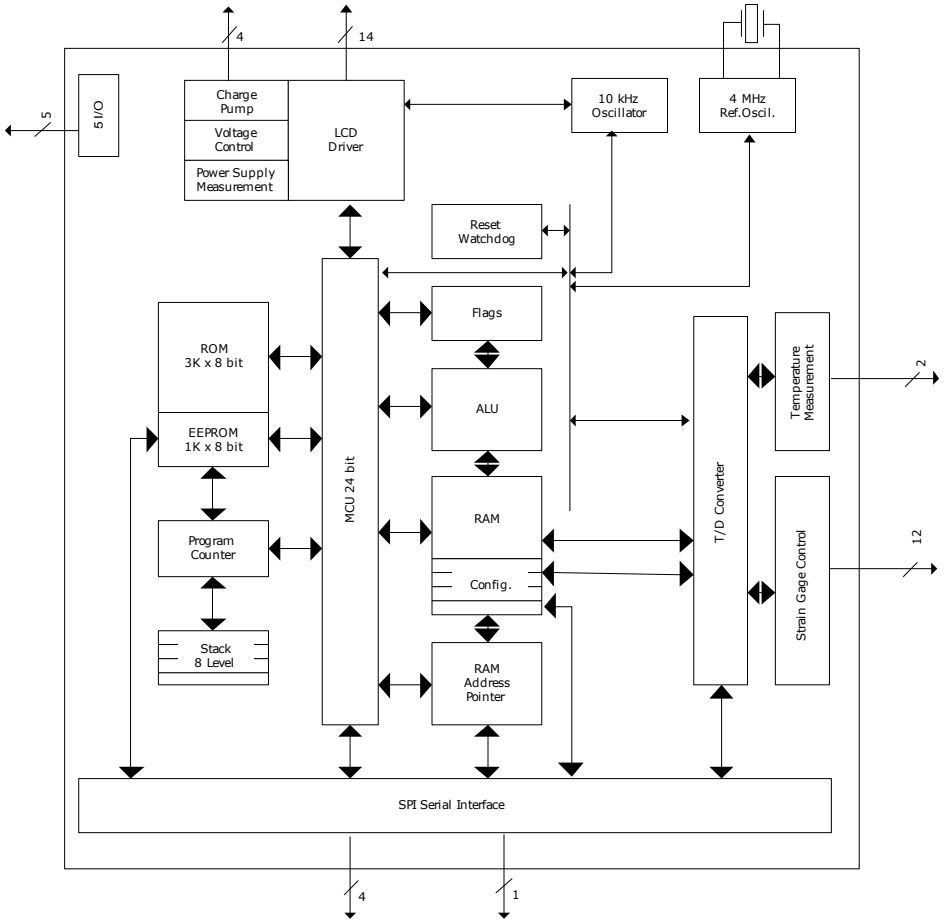
1.2 Features

- PICOSTRAIN Front-End with up to 19 Bit resolution (@2mV/V)
- Microprocessor with 24-Bit data structure
- 1 k x 8-Bit EEprom progam memory
- 3 k ROM powerful program code like 48 Bit multiplication and division or binary to 7-segment conversion
- 8-layer hardware stack
- Embedded very low current 10 kHz oscillator
- Driver for external 4 MHz ceramic oscillator
- Standby current $<1\mu A$
- 5 programmable I/O-ports
- 4 x 14, 3 x 15, 2 x 16 LCD driver
- Embedded charge pump for driving the LCD at 3.0 V and 2.5 V
- Embedded bandgap voltage reference for low battery detection
- Ports for temperature measurement with low-cost carbon resistors
- Watchdog timer
- Serial SPI interface
- Supply voltage 2.2 to 3.6 V at 130 dB PSRR
- System operational current down to 50 μA
- As dice (90 μm pitch) or packaged (QFN64)

1.3 Applications

- Solar scales
- Body scales
- Kitchen scales
- Postal scales
- Package scales
- Calibrated scales

1.4 Architecture



2 Characteristics and Specifications

2.1 Pin Assignment

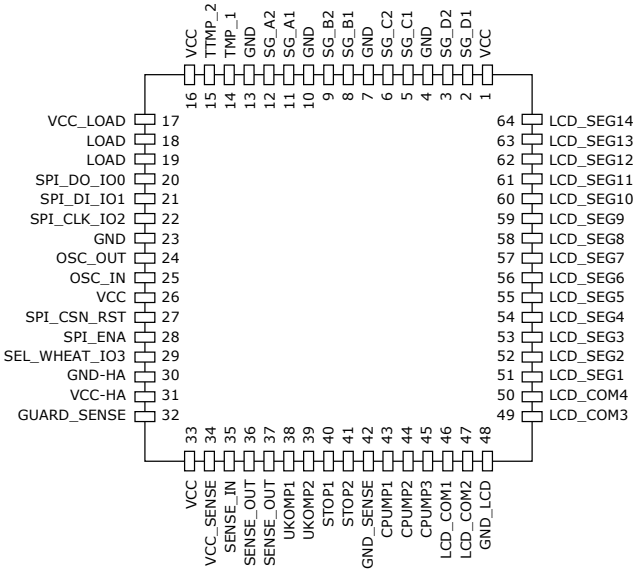


Abbildung 1 Pin Assignment

2.2 Pin Description

No.	Name	Description	Value	If not used
1	Vcc	Supply voltage digital part , I/O, 4MHz-osc.		
2	SG_D1	Port 1 halfbridge D		
3	SG_D2	Port 2 halfbridge D		
4	GND	Ground for digital part, I/O, Load, LCD, osc.		
5	SG_C1	Port 1 halfbridge C		
6	SG_C2	Port2 halfbridge C		
7	GND	Ground for digital part, I/O, Load, LCD, osc.		
8	SG_B1	Port 1 halfbridge B		
9	SG_B2	Port 2 halfbridge B		
10	GND	Ground for digital part, I/O, Load, LCD, osc.		
11	SG_A1	Port 1 halfbridge A		
12	SG_A2	Port 2 halfbridge A		
13	GND	Ground for digital part, I/O, Load, LCD, osc.		
14	TMP_1	Port 1 temperature measurement		
15	TMP_2	Port 2 temparature measurement		
16	Vcc	Supply voltage digital part , I/O, 4MHz-osc.		
17	Vcc_load	Power supply load output pins 1 and 2		
18	Load	Load output to measuring capacitor		
19	Load	Load output to measuring capacitor		
20	SPI_SO_IO0	Output serial SPI interface or IO0		
21	SPI_SI_IO1	Input serial SPI interface or IO1		
22	SPI_SCK_IO2	Clock serial SPI interface or IO2		
23	GND	Ground for digital part, I/O, Load, LCD, osc.		
24	OSC_OUT	Output to 4MHz ceramic resonator		
25	OSC_IN	Input to 4MHz ceramic resonator		
26	Vcc_OSC	4MHz Oscillator supply voltage		
27	SPI_CSN_RST	Slave select or RST input (High active)		
28	SPI_ENA	Serial SPI interface enable		
29	SEL_WHEAT_IO 3	Select for Wheatestone Mux in the comparator or IO3		
30	GND_HA			
31	Vcc_HA			
32	GUARDE_SENS E	Guard ring sensing part – GND		
33	Vcc	Supply voltage digital part , I/O, 4MHz-osc.		
34	Vcc_SENSE			
35	SENSE_IN	Input internal CMOS comparator		
36	SENSE_OUT	Output internal CMOS comparator		
37	UCOMP1	External comparator circuit connection		

38	UCOMP2	External comparator circuit connection		
39	STOP1	Stop input measuring signal		
40	STOP2	Stop input measuring signal		
41	GND_SENSE	GND sense electronics		
42	VCC_LCD	Supply voltage LCD, 10kHz osc., bandgap		
43	CP1	LCD voltage doubling and stabilization		
44	CP2	LCD voltage doubling and stabilization		
45	CP3	LCD voltage doubling and stabilization		
46	LCD_com1	LCD line driver for 1/2, 1/3, 1/4 duty		
47	LCD_com2	LCD line driver for 1/2, 1/3, 1/4 duty		
48	GND_LCD			
49	LCD_com3	LCD line driver for 1/3, 1/4 duty, row driver for 1/2 duty		
50	LCD_com4	LCD line driver for 1/4 duty, row driver for 1/2 , 1 /3 duty		
51	LCD_seg1	LCD row driver		
52	LCD_seg2	LCD row driver		
53	LCD_seg3	LCD row driver		
54	LCD_seg4	LCD row driver		
55	LCD_seg5	LCD row driver		
56	LCD_seg6	LCD row driver		
57	LCD_seg7	LCD row driver		
58	LCD_seg8	LCD row driver		
59	LCD_seg9	LCD row driver		
60	LCD_seg10	LCD row driver		
61	LCD_seg11	LCD row driver		
62	LCD_seg12	LCD row driver		
63	LCD_seg13	LCD row driver		
64	LCD_seg14	LCD row driver		

2.3 Absolute Maximum Ratings

Symbol	Parameter	Conditions	Min	Max	Unit
Vcc Vcc_load Vcc_osc Vcc_HA Vcc_Sense Vcc_LCD	Supply voltage	Vcc vs. GND	-0.5	5.0	V
Vin	DC input voltage		-0.5	Vcc + 0.5	V
Tstg	Storage Temperature	Plastic package	-55	150	°C

2.4 Normal Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
Vcc Vcc_load Vcc_osc Vcc_HA Vcc_Sense Vcc_LCD	Supply voltage	Vcc vs. GND (* without EEPROM programming and voltage measurement)	2.2 (1.4*)	3.6	V
Vin	DC input voltage		0.0	Vcc	V
Vout	Output voltage		0.0	Vdd	V
Tj	Junction temperature		-55	150	
Tstg	Storage temperature	Plastic package	-55	150	°C

2.4.1 Electrical Characterization

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vil	Input low voltage	CMOS			0.3Vcc	V
Vih	Input high voltage	CMOS	0.7Vcc			
Vhyst	Input hysteresys	Vcc = 3.6 V Vcc = 3.0 V Vcc = 2.7 V Vcc = 2.2 V Vcc = 1.8 V		400 280 225 150 80		mV
Voh	Output high voltage		0.8			V
Vol	Output low voltage				0.2Vcc	V
Vlbat	Low battery voltage detect		2.2		2.9	V
LCD_com LCD_seg	LCD driver Voltage stabilized	lcd_vlt = 0 lcd_vlt = 1 lcd_vlt = 2		2.0 2.5 3.0		V

2.5 Converter Precision

The following tab les shows the measurement capability at different supply voltages for the PS∅8.

Table 1 Performance at Vcc = 2,6 V

Measuring	ENOB SG Ration		No. off eff. LSB's @ 2mV/V		Noise in nV	
Rate in Hz	Fast settle	SINC8	Fast settle	SINC8	Fast settle	SINC8
500	23.4	24.9	14.4	15.9	234	83
250	23.8	25.3	14.8	16.3	182	64
100	24.6	26.1	15.6	17.1	104	37
50	25.4	26.9	16.4	17.9	61	21
20	26.0	27.5	17.1	18.6	38	13
10	26.5	28.0	17.5	19.0	28	10
5	26.9	28.4	17.9	19.4	21	7

SG Ratio: Resolution refered to resistance reatio of the SG resolution referred to 1 mV/V max. output (full scale)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
INL	Integral Non-linearity strain Offset Offset drift Gain drift Over 0°C ... 70°C	 Initial, uncalibrated Total system, 1 kΩ DMS, 3V Full-bridge Half-bridge Total System. 1 KΩ DMS, 5V		 < 1		% of FS µV nV/K nV/K ppm/K
PSSR1[1]	Power Supply Rejection Ratio Vcc			140		dB
PSSR2[1]	Power Suppöy Rejection Ratio VCC					dB

2.6 Current Consumption

The following table shows the total current consumption of the scale including all currenrs.

Divisions **	Update Rate	Double Tara*	Operating Current @ 3V		Scale type	Operating hours
2,000	3 Hz	yes	1 kOhm	15 µA	Solar	
2,000	5 Hz	yes	1 kOhm 350 Ohm	60 µA 90 µA	Postal, Body, Kitchen , Pocket	3,000 hours (1xCR2032)
5,000	5 Hz	yes	1 kOhm 350 Ohm	120 µA 220 µA	High-end postal, Kitchen, Pocket	1,500 hours (1xCR2032)
10,000	5 Hz	yes	1 kOhm	300 µA	High-end pocket,	2,000 hours

			350 Ohm	700 µA	Counting	(1xCR2430)
80,000	5 Hz	no	1 kOhm 350 Ohm	1.9 mA 4.5 mA	Counting	1,500hours 2 x AA

* With double Tara there is no overload of the loadcell if tara is set at max. load and additional max. load is put on the scale. In other words the sensor output is 1mV/V only at maximum load (no = 2 mV/V @ max load).
* * Divisiions are peak-peak values with 5 Sigma (e. g. 80.000 Divisions are 400.000 divisions of effective resolution)

2.7 Timings

At Vcc= 3,3 V ± 0,3V, ambient temperature –40°C ... +85°C unless otherwise specified

2.7.1 Oscillators

Table 3 Oscillator timing

Symbol	Parameter	Min	Typ	Max	Units
Clk10kHz	10 kHz reference oscillator		10		kHz
to10st	Oscillator start-up				µs
ClkHS	High-speed reference oscillator		4		MHz
toHSst	Oscillator start-up time with ceramic resonator				µs

2.7.2 SPI-Interface

Table 4 Serial interface timing

Symb	Parameter	Min	Typ	Max	Unit
		Max @ V_{io} =			
		2.0V	2.5V	3.3V	
f_{clk}	Serial clock frequency				MHz
		Min @ V_{io} =[1]			
		2.0V	2.5V	3.3V	
t_{pwh}	Serial clock, pulse width high				ns
t_{pwl}	Serial clock, pulse width low				ns
t_{sussn}	SSN enable to valid latch clock				ns
t_{pwssn}	SSN pulse width between write cycles				ns
t_{hssn}	SSN hold time after SCLK falling				
t_{sud}	Data set-up time prior to SCLK falling				ns
t_{hd}	Data hold time before SCLK falling				ns
		Max @ V_{io} =			
		1.8V	2.5V	3.3V	
t_{vd}	Data valid after SCLK rising				ns

Serial Interface (SPI compatible, Clock Phase Bit =1, Clock Polarity Bit =0):

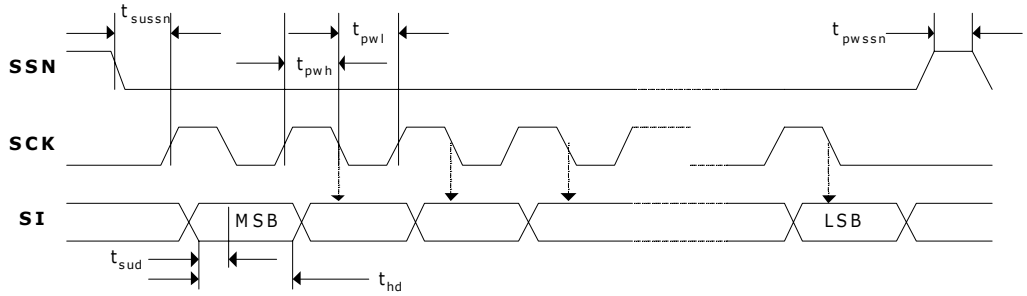


Figure 1 Write

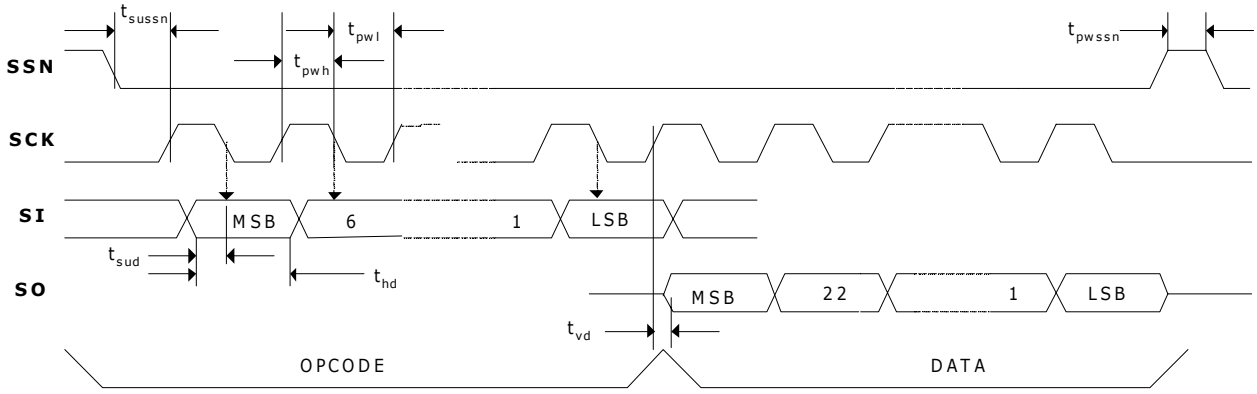
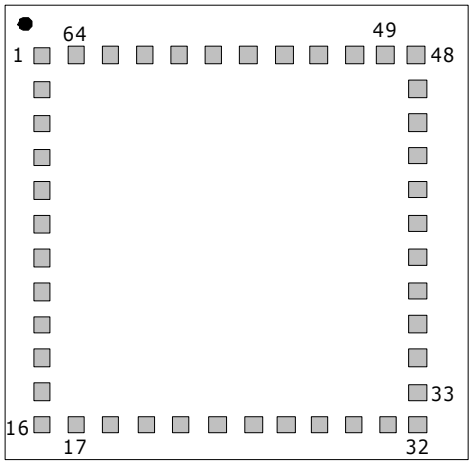


Figure 2 Read

2.8 Package Information

2.8.1 PAD Assignment



2.8.2 Bonding PAD Location

Table 6 Pad location

Pad#	Name	X [µm]	Y [µm]	Position	Pad#	Name	X	Y	Position
1	Vcc			right	33	Vcc			left
2	P14			right	34	Vcc_SENSE			left
3	P24			right	35	SENSE_IN			left
4	GND			right	36	SENSE_OUT			left
5	P13			right	37	UKOMP1			left
6	P23			right	38	UKOMP2			left
7	GND			right	39	STOP1			left
8	P12			right	40	STOP2			left
9	P22			right	41	GND_SENSE			left
10	GND			right	42	VCC_LCD			left
11	P11			right	43	CPUMP1			left
12	P21			right	44	CPUMP2			left
13	GND			right	45	CPUMP3			left
14	PSEP1			right	46	LCD_COM1			left
15	PSEP2			right	47	LCD_COM2			left
16	Vcc_Load			right	88	GND_LCD			left
17	LOAD			up	49	LCD_COM3			down
18	LOAD			up	50	LCD_COM4			down
19	SPI_DO_IO0			up	51	LCD_SEG1			down
20	SPI_DI_IO1			up	52	LCD_SEG2			down
21	SPI_CLK_IO2			up	53	LCD_SEG3			down
22	GND			up	54	LCD_SEG4			down
23	OSC_OUT			up	55	LCD_SEG5			down

24	OSC_IN			up	56	LCD_SEG6			down
25	Vcc_OSC			up	57	LCD_SEG7			down
26	SPI_CSN_RST			up	58	LCD_SEG8			down
27	SPI_ENA			up	59	LCD_SEG9			down
28	SEL_WHEAT_IO3			up	60	LCD_SEG10			down
29	GND_HA			up	61	LCD_SEG11			down
30	Vcc_HA			up	62	LCD_SEG12			down
31	Vcc			up	63	LCD_SEG13			down
32	GUARD_SENSE			up	64	LCD_SEG14			down

PAD#: 64 Pads: Die Size: XXX x XXX mm^2

2.8.3 QFN64 Package Outline

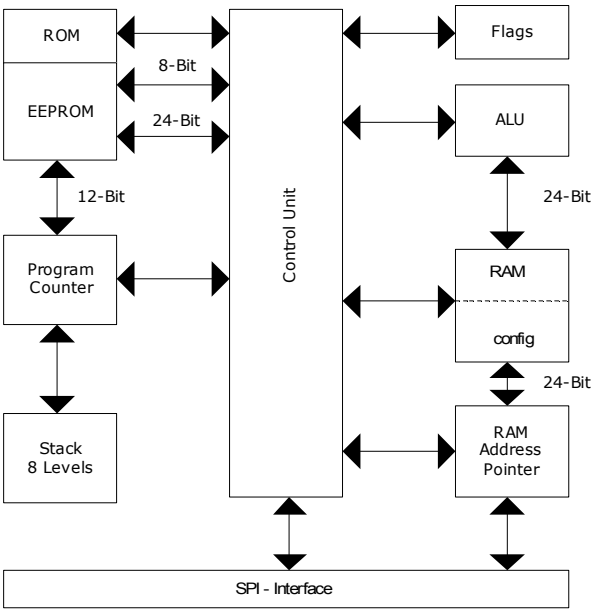
t.b.d.

2.8.4 QFN64 Recommended Pad Layout

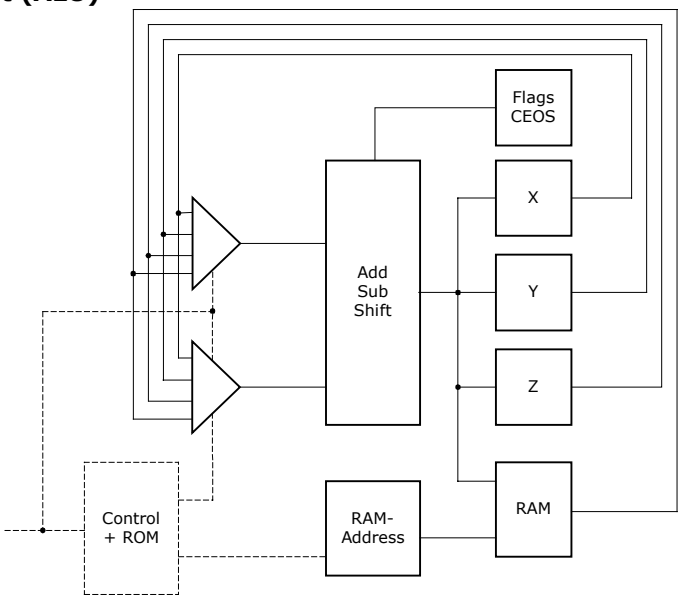
t.b.d.

3 Central Processing Unit (CPU)

3.1 Block Diagram



3.2 Arithmetic Logic Unit (ALU)



3.2.1 Accumulators

The ALU has three 24-Bit accumulators, X, Y and Z. The RAM is addressed by the RAM addresss pointer and the addressed RAM cell is used as forth accumulator. By the ram adress pointer a single ram adress is mapped into the ALU . So in total there are 4 accumulators. All transfer operations (move, swap) and arithmetic–operations (shift, add, mult...) can be applied to all accumulators.

3.2.2 Flags

The processor controls 4 flags for each operation. Not-Equal and Sign flags are set with each write access to one of the accumulators (incl. RAM). Additionally, the Carry and Overflow flags are set in case of a calculation (Add/Sub/shiftR). It is possible to query each flag by a jump instruction.

- Carry
  Shows the carry over inan addition or substraction. With shift operations (shiftL, rotR etc.) it shows the postponed bit.
- Not-equal zero
  This flag is set to zero in case a new result unequal zero is written into an accumulator (add,sub,move,swap etc.).
- Sign
  The Sign is set when a new result is written into an accumulator (add,sub,move,swap etc.) and the highest bit (MSB) is 1.
- Overflow
  Indicates an overflow during an addition or substraction of two numbers in the meaning of two's complement.

3.3 Memory Organization

3.3.1 ROM and EEPROM Organization

4095 ... 1024	Program Memory ROM
1023 ... 48	Program Memory EEPROM bank 1 EEPROM bank 2 Program entry
47...45 44...42 41...39	Read/Write EEprom 15 (e.g. calibration data) Read/Write EEprom 14 (e.g. calibration data) Autoconfig EEPROM Spec
38...35 5...3 2...0	Autoconfig EEprom 12 Autoconfig EEprom 1 Autoconfig EEprom 0

The ROM area is starting at address 1024. There are all computation routines needed for the PICOSTRAIN measuring method. There are also further helpful routines that are frequently needed in weight scale applications, e.g. decimal to 7-segment code conversion. These routines can be called by a program in the EEProm. The program can also jump back from the ROM to the EEProm when configured.
The EEProm is 1024 Bites big. The program memory occupies 976 bytes starting from address 48. Each jump from the ROM into the EEProm starts at address 48. The program in the EEProm may find out the reason for the jump by means of the status information in register 22.

The lower 48 bytes in the EEProm are reserved for an automatic configuration of the PS08 during a power-on reset. 3 succesive bytes are added to a 24 bit word. So there are 16

words of 24 bit that can be read by the program code.

The lower 33 bytes from 0 to 39 make 11 words of 24 bit and are used for configuration. During a power- on reset they are copied into RAM address 48 to 60.

EEprom cells 39 to 47 are not necessary for the standard configuration. The processor can write to and read from those cells during operation (putEpr and gerEpr). They can be used for saving calibration data.

3.3.2 RAM Organization

64	Config Reg Spec
63 48	Config Reg 15 ... Config Reg 0
47 ... 33	Not used
32 ... 26 25 24 23 22 21 20 19 18 17 16	System RAM ... System RAM UBATT CAL HB1+ Flags (p1-p2)/p2 HB0 = (A-B)+(...) / (A+B)+(...) HB4 = (G-H)/(G+H) HB3 = (E-F)/(E+F) HB2 = (C-D)/(C+D) HB1 = (A-B)/(A+B)
15 ... 0	User RAM 15 ... User RAM 0

3.3.3 RAM-Adresspointer

The RAM has its own address bus with 64 addresses. The with of 24 Bit corresponds to the registe width of the ALU. By means of the ram address pointer a single ram address is mapped into the ALU. It then acts as a fourth accumulator register. Changing the ram address pointer does not effect the content of the addressed ram. The RAM address pointer is modified by separate opcodes (ramadr, incramadr,...)

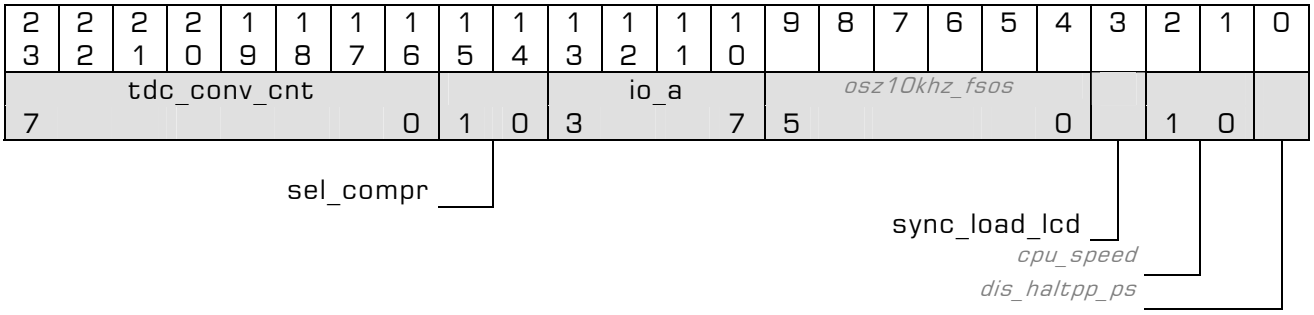
3.4 Register Set

3.4.1 Configuration Registers

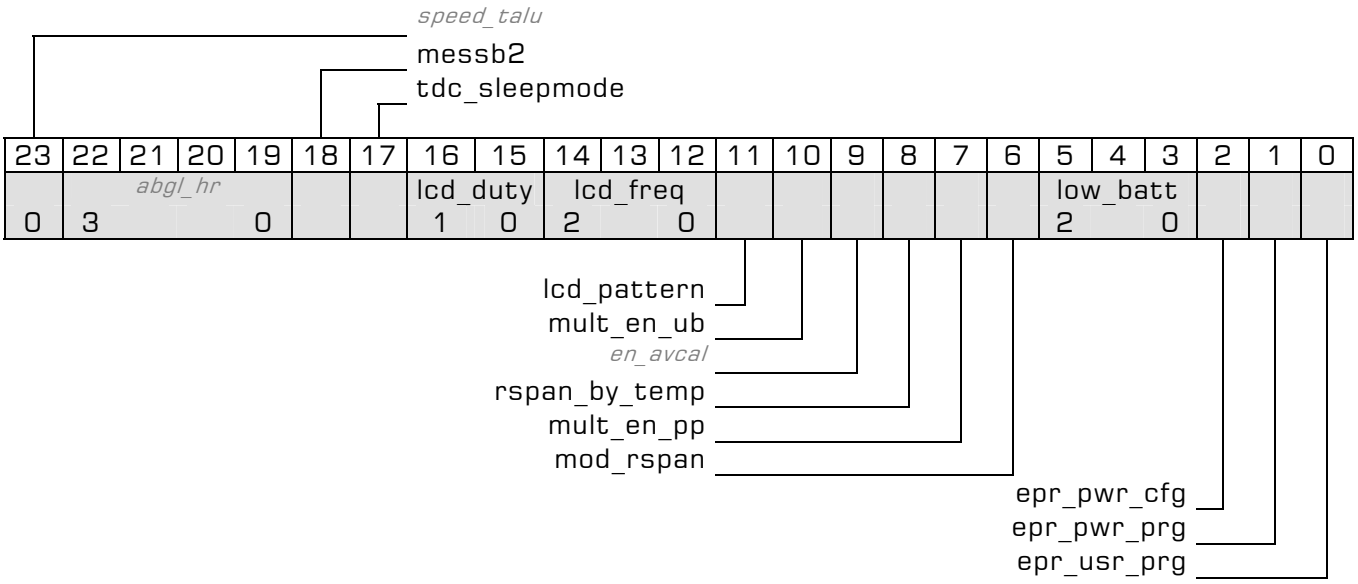
PS08 has 16 configuration registers of 24 Bit width, to be addressed in the RAM from address 48 to 63. The configuration registers control the whole chip including the strain measurement and the LCD controller. A 17th configuration register on address 64 is for testing purposes only.

It is possible to write into the configuration registers
- By the microporcessor during operation
- Through the SPI interface from an external processor
- During the Power-on reset transfering a basic configuration from the EEPROM

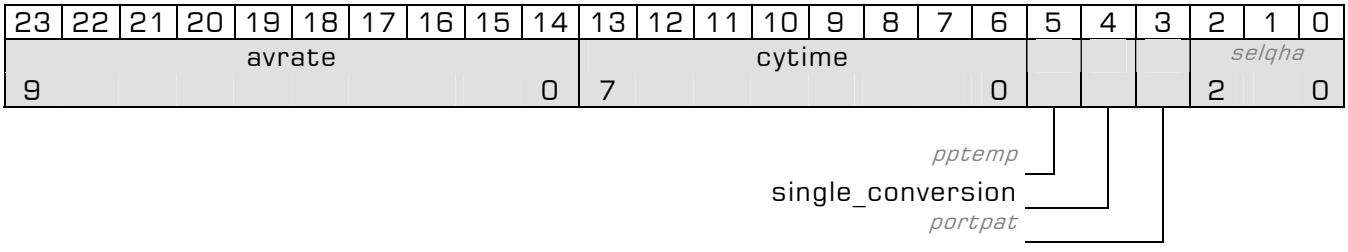
Configreg_00: RAM address 48 EEPROM address 0



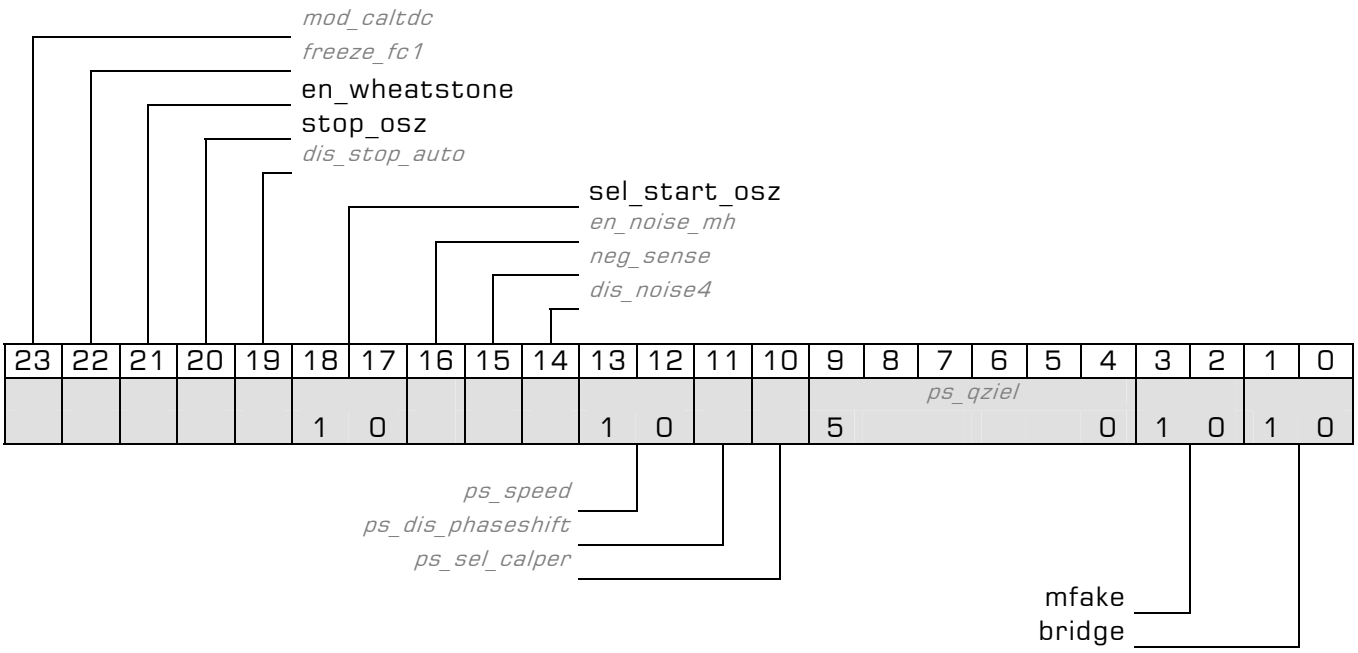
Configreg_01: RAM address 49 EEPROM address 3



Configreg_02: RAM address 50 EEPROM address 6 - 8



Configreg_03: RAM address 51 EEPROM address 9 - 11



Configreg_04: RAM address 52 EEPROM address 12 - 14

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb1																							
23																							0

Configreg_05: RAM address 53 EEPROM address 15 - 17

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb2																							
23																							0

Configreg_06: RAM address 54 EEPROM address 18 - 20

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb3																							
23																							0

Configreg_07: RAM address 55 EEPROM address 21 - 23

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb4																							
23																							0

Configreg_08: RAM address 56 EEPROM address 24 - 26

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

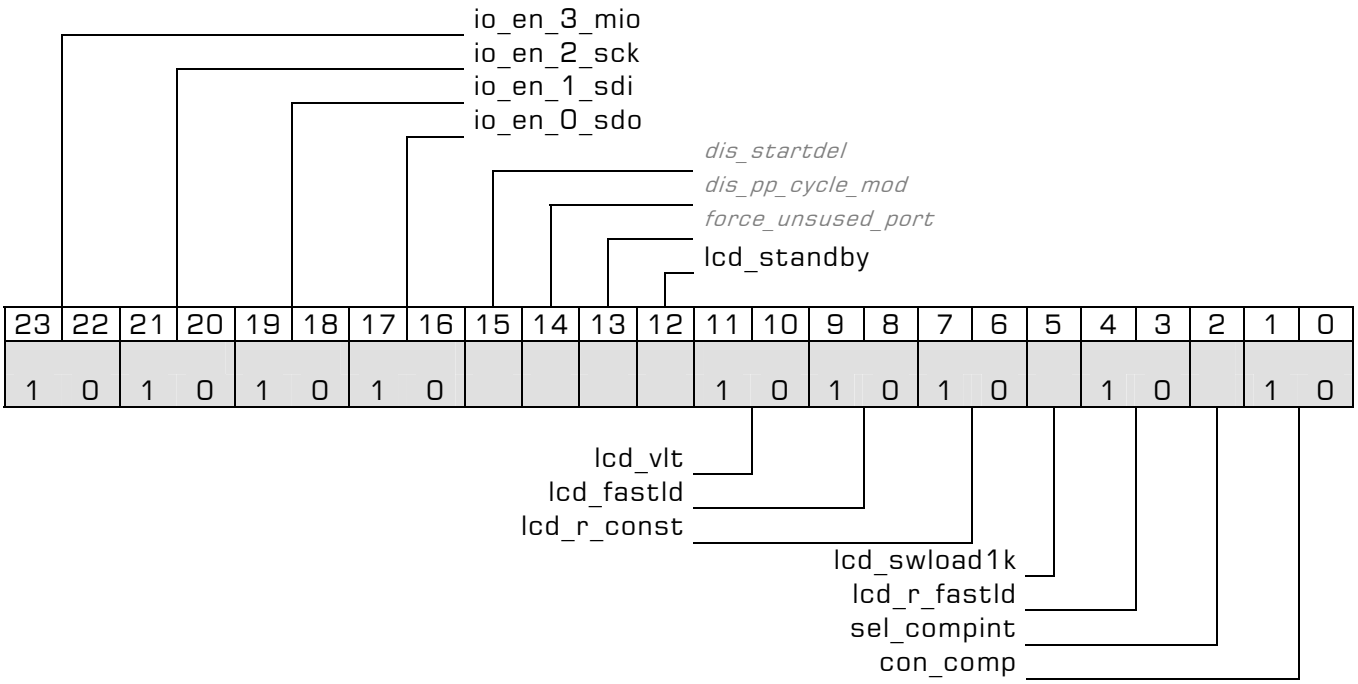
Configreg_09: RAM address 57 EEPROM address 27 - 29

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Tk0																							
23																							0

Configreg_10: RAM address 58 EEPROM address 30 - 32

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
calcor								Mult_Ub								Mult_PP							
7							0	7							0	7							0

Configreg_11: RAM address 59 EEPROM address 33 - 35



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

Configreg_12: RAM address 60 EEPROM address 36 - 38

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

Configreg_13: RAM address 61

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

Configreg_14: RAM address 62

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lcd_segment																							
47																							24

Configreg_15: RAM address 63

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1'bx																lcd_segment							
15															0	55							48

Configreg_SPEC: RAM address 64 EEPROM address 39 - 41

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
internal use																							

Configuration Value	Register	Recomm. Default	Hardware Default	Description
abgl_hr[3:0]	*1*	*5*	*5*	*High resolution trim*
avrate[9:0]	2	25	0	internal averaging rate
bridge[1:0]	3	0	0	0 = 1 half bridge 1 = 2 half bridges 2 = not reasnable 3 = 4 half bridges
calcor[7:0]	*10*	*0*	*0*	*Correction factor for TDC calibration value cal := cal + calcor/8 = cal + [-127 to +128]/8 = cal + [-15.875 to +16.000]*
comp2_sel]1:0]	*SPEC*	*0*	*0*	*Coice of second input stage 0 = Single stage 1 = Schmitt trigger HC14 2 = Improved Schmitt trigger*
con_comp[1:0]	11	0	0	comparator switch-off mode 00 = off 01 = on during mearuement 10 = on during loading 11 = always on
cpu_speed[1:0]	*0*	*3*	*3*	*CPU ring oscillator speed 00 = fast 01 = default 10 = slow 11 = very slow*
cytime[7:0]	2	25	25	Cycle time in multiples of the 10 kHz period (messb2 = 0) or 8 * 4 MHz-period (messb2 = 1)
dis_haltpp_ps	*0*	*0*	*0*	*Phase shift halt disable 0 = hold phase shift unit during gain measurement 1 = run phase shift unit during gain measurement*
dis_noise4	*3*	*0*	*0*	*Disable main noise unit*
dis_pp_cycle_mod	*11*	*0*	*0*	*Disable doppelte Cycle Time bei PP-Messung*

				Disable gain measurement with double cycle
dis_startdel	*11*	*0*	*0*	*Disable start delay*
dis_stop_auto	*3*	*0*	*0*	*Disable oscillator auto-off at single-converion*
en_avcal	*1*	*0*	*0*	*Enable TDC calibration value averaging by factor 16*
en_noisemh	*3*	*1*	*1*	*Enable noise at multihit*
en_wheatstone	3	0	0	Enable Wheastone mode
epr_pwr_cfg	1	0	0	1 = Configuration in the EEprom is used after a power-on reset
epr_pwr_prg	1	0	0	1 = After power-on reset the PowrOnResetcode in the EEprom is run, starting on address 48
epr_usr_prg	1	0	0	1 = After power-on the user code in the EEprom is started
force_unused_port	*11*	*0*	*0*	*force unused measurement ports to GND*
freeze_fc1	*3*	*1*	*1*	*Freeze finecount 1 (FC1) for each 2nd measurement to reduce noise*
io_a[3:0]	0	0	0	I/O's Output: output value, can be read back Input: read input value
io_en_0_sdo[1:0]	11	3	2	Port definition 00 = input 01 = input with pull-up 10 = input with pull-down 11 = output
io_en_1_sdi[1:0]	11	3	2	Port definition 00 = input 01 = input with pull-up 10 = input with pull-down 11 = output
io_en_2_sck[1:0]	11	3	2	Port definition 00 = input 01 = input with pull-up 10 = input with pull-down 11 = output
io_en_3_mio[1:0]	11	3	2	Port definition 00 = input 01 = input with pull-up 10 = input with pull-down 11 = output
lcd_duty[1:0]	1	0	0	LCD duty cycle definition

<table>
<tr><td></td><td></td><td></td><td></td><td>0 = off
1 = 1/2duty
2 = 1/3duty
3 = 1/4duty</td></tr>
<tr><td>lcd_fastld[1:0]</td><td>11</td><td>0</td><td>0</td><td>Configures the number of fastload periods
(10ms) with low-ohmic voltage divider</td></tr>
<tr><td>lcd_freq[2:0]</td><td>1</td><td>4</td><td>0</td><td>Switch LCD frequency (switch-on time of pixels)
<table>
<tr><td>Pixel</td><td>Time</td><td colspan="4">Multiplex mode</td></tr>
<tr><td></td><td></td><td>1/4</td><td>1/3</td><td>1/2</td><td>Hz</td></tr>
<tr><td>0</td><td>8.0ms</td><td>15</td><td>20</td><td>31</td><td>Hz</td></tr>
<tr><td>1</td><td>4.8ms</td><td>26</td><td>34</td><td>52</td><td>Hz</td></tr>
<tr><td>2</td><td>4.0ms</td><td>31</td><td>42</td><td>62</td><td>Hz</td></tr>
<tr><td>3</td><td>3.2ms</td><td>30</td><td>52</td><td>78</td><td>Hz</td></tr>
<tr><td>4</td><td>2.4ms</td><td>52</td><td>69</td><td>104</td><td>Hz</td></tr>
<tr><td>5</td><td>2.0ms</td><td>62</td><td>83</td><td>125</td><td>Hz</td></tr>
<tr><td>6</td><td>1.6ms</td><td>78</td><td>104</td><td>176</td><td>Hz</td></tr>
<tr><td>7</td><td>1.2ms</td><td>104</td><td>138</td><td>208</td><td>Hz</td></tr>
</table></td></tr>
<tr><td>lcd_pattern</td><td>1</td><td>0</td><td>0</td><td>0 = Standard
1 = Current saving LCD driving</td></tr>
<tr><td>lcd_pos[23:00]</td><td>12</td><td>o76543210</td><td>o76543210</td><td>Position of LCD segments</td></tr>
<tr><td>lcd_r_const[1:0]</td><td>11</td><td>1</td><td>3</td><td>Defines the cross resistance of the LCD voltage divider
0 = 15 k
1 = 200 k
2 = 800 k
3 = 1600 k</td></tr>
<tr><td>lcd_r_fastld[1:0]</td><td>11</td><td></td><td></td><td>Selects the resistor for fastly charging the LCD pixels
0 = 15 k
1 = 200 k
2 = 800 k
3 = 1600 k</td></tr>
<tr><td>lcd_segment[23:0]</td><td>13</td><td>h000000</td><td>h000000</td><td>Display segments digits 2 to 0</td></tr>
<tr><td>lcd_segment[47:24]</td><td>14</td><td>h000000</td><td>h000000</td><td>Display segments digits 5 to 3</td></tr>
<tr><td>lcd_segment[55:48]</td><td>15</td><td>h000000</td><td>h000000</td><td>Display segments special characters (1/3 and 1/4 duty)</td></tr>
</table>

lcd_standby	11	0	1	0 = LCD active 1 = LCD voltage supply in stand-by
lcd_swload1k	11	0	0	LCD driver's voltage doubler uses 1 kOhm instead of 200 Ohm
lcd_vlt[1:0]	11	0	0	LCD voltage 0 = 2 V 1 = 2.5 V 2 = 3 V 3 = 2 V without pump
low_batt[2:0]	1	0	0	Sets the voltage level for low battery detection and EEpromwrite 2.2 V, 2.3 V, 2.4 V to 2.9 V (2.2 V + 0.1 V * low_batt)
messb2	1	0	0	1 = Set TDC Measurement range 2
mfake[1:0]	3	0	0	Sets the number of fake measurements
mod_caltdc	*3*	*1*	*1*	*Sets TDC calibration period* *0 = each measurement* *1 = each 10th measurement (P11)*
mod_rspan	1	0	0	1 = Enable internal multiplication of gain compensation resistor Rspan
mult_en_pp	1	0	0	1 = Enable multiplications in gain correction
mult_en_ub	1	0	0	1 = Enable multiplications for supply voltage correction
Mult_Hb1[23:0]	4	h100000	h000000	Multiplication factor for HB1 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20
Mult_Hb2[23:0]	5	h100000	h000000	Multiplication factor for HB2 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20
Mult_Hb3[23:0]	6	h100000	h000000	Multiplication factor for HB3 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20
Mult_Hb4[23:0]	7	h100000	h000000	Multiplication factor for HB4 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20
Mult_PP[7:0]	10	h80	h00	Multiplication factor for gain correction g := g * [0 to 255]/2^7
Mult_TkG[23:0]	8	h000000	h000000	Amplification for Rspan correction Rs := Rs * [-2^23 to 2^23-1]/ 2^20
Mult_TkO[23:0]	9	h010000	h000000	Offset value for Rspan, directly substracted

Mult_Ub[7:0]	10	h00	h80	Multultiplication factor for gain compensation by means of voltage measurement hb = hb/(1 + ub*[-128 to 127]/2^21)
neg_sense	*3*	*0*	*0*	*Invert sense input*
osz10khz_fsos[6:0]	*0*	*0*	*0*	*Frequncy trim of 10 kHz oscillator*
portpat	*2*	*0*	*0*	*Switch port patterns*
pptemp	*2*	*0*	*0*	*Enable gain error and temperature measurement*
ps_dis_phaseshift	*3*	*0*	*0*	*Disable phase shifter*
ps_qziel[5:0]	*3*	*22*	*22*	*Target number of shifts per period for the phase shifter*
ps_sel_calper	*3*	*0*	*0*	*Calibration period for phase shifter* *0 = each measurement* *1 = each 16th measurement*
ps_speed[1:0]	*3*	*0*	*0*	*Selects the next measurement for aphase shift*
rspan_by_temp	1	0	0	Use temperature measurement instead of Rspan for temperature compensation
sel_compint	11	0	0	1 = Select internal comparator
sel_compr	0			Selects camparator working resistor 00 = 10k 01 = 10k 10 = 7k 11 = 4.1k
sel_start_osz[1:0]	3	0	0	Sets delay from start of 4 MHz oscillator to start of measurement 0 = off 1 = continuously on 2 = 800us 3 = 1500us
selqha[2:0]	*2*	*2*	*0*	*For acam internal use only*
single_conversion	2	0	0	1 = Switch on timer controlled single measurements
speed_talu	*1*	*0*	*0*	*For acam internal use only*
stop_osz	3	0	0	Stop the oszillator by command (e.g. if there is no interrupt after AutoOn)
synch_load_lcd	0	0	0	Synchronizes the start of a

				measurement with the refresh of the LCD display when measuring in timer mode
tdc_conv_cnt[7:0]	0	0	0	Single conversion timer based on 10 kHz/64 = 156.25 Hz
tdc_sleepmode	1	0	0	Scan mode without TDC or strain gage measurement, to be used for scanning buttons in case the scale is off, same as avrate=0

3.4.2 Result Registers

Content of the RAM result registers at the end of a measurement:

ram=16	: HB1=(A-B) / (A+B)	HB1 un-compensated
ram=17	: HB2=(C-D) / (C+D)	HB2 un-compensated
ram=18	: HB3=(E-F) / (E+F)	HB3 un-compensated
ram=19	: HB4=(G-H) / (G+H)	HB4 un-compensated
ram=20	: HB0=(A-B)+(..)/(A+B)+(..)	HB0 compensated sum
ram=21	: TMP=(p1-p2) / p2	Temperature
ram=22	: Status	
ram=23	: HB1+	Time measurement TDC at SG_A1, Pin11
ram=24	: CAL	Resolution TDC
ram=25	: UBATT	Measured supply voltage
ram=26-31	: NC	Free during EEprom program
x-Akku	: HB0	Value of ram=20
y-Akku	: Temp	Value of ram=21
z-Akku	: Flags	Value of ram=63
ramadr	: 0	Value of RAM address pointer

Descriptions:

A : Discharge time measurement at SG_A1, Pin 11
B : Discharge time measurement at SG_A2, Pin 12
C : Discharge time measurement at SG_B1, Pin 8
D : Discharge time measurement at SG_B2, Pin 9
E : Discharge time measurement at SG_C1, Pin 5
F : Discharge time measurement at SG_C2, Pin 6
G : Discharge time measurement at SG_D1, Pin 2
H : Discharge time measurement at SG_D2, Pin 3
P1: Discharge time measurement at TMP_1, Pin 14
P2: Discharge time measurement at TMP_2, Pin 15

Formats:

HB1: Result in 100 ppm, HB1/100 = result in ppm
HB2: Result in 100 ppm, HB2/100 = result in ppm
HB3: Result in 100 ppm, HB3/100 = result in ppm
HB4: Result in 100 ppm, HB4/100 = result in ppm
HB0: Result in 100 ppm, HB0/100 = result in ppm
TMP: Ergebnis(Tmp) = TMP/(1<<24)
Status: See below
HB1+: Result (HB1+)/ns = 250 * HB1+ /(1<<14) [4MHz clock]
CAL: Result (Cal)/ps = 250000 / CAL [4MHz clock]

UBATT: Result (UBATT)/V = 2.0+1.6*UBATT/64
HB1,HB2,HB3,HB4,HB0,TMP are given as two's complement. MSB = 1 indicates a negative value. To get the possitive value calculate h1000000000000-X.

3.4.3 Status Register

Bit	Bedeutung
Status[23]= flg_io3_mio	Pin29
Status[22]= flg_io2_sck	Pin22
Status[21]= flg_io1_sdi	Pin21
Status[20]= flg_io0_sdo	Pin20
Status[19]= flg_rstpwr	1 = Power-on reset caused jump into EEPROM
Status[18]= flg_rstssn	1 = Pushed button caused jump into EEPROM
Status[17]= flg_wdtalt	1 = Watchdog interrupt caused jump into EEPROM
Status[16]= flg_endavg	1 = End of measurement caused jump into EEPROM
Status[15]= flg_intav0	1 = Jump into EEPROM in sleep mode
Status[14]= flg_ub_low	1 = Low voltage
Status[13]= flg_errtdc	1 = TDC error
Status[12]= flg_pslock[1]	1 = Phase shifter locked
Status[11]= flg_pslock[0]	1 = Phase shifter locked
Status[10]= flg_errprt	1 = Error at strain gauge ports
Status[09]= flg_timout	1 = Timeout TDC
Status[08]= 1'bx	1 = not used
Status[07]= flg_io3_mio_r	1 = Rising edge at Pin29, 0 = no edge
Status[06]= flg_io2_sck_r	1 = Rising edge at Pin22, 0 = no edge
Status[05]= flg_io1_sdi_r	1 = Rising edge at Pin21, 0 = no edge
Status[04]= flg_io0_sdo_r	1 = Rising edge at Pin20, 0 = no edge
Status[03]= flg_io3_mio_f	1 = Falling edge at Pin29, 0 = no edge
Status[02]= flg_io2_sck_f	1 = Falling edge at Pin22, 0 = no edge
Status[01]= flg_io1_sdi_f	1 = Falling edge at Pin21, 0 = no edge
Status[00]= flg_io0_sdo_f	1 = Falling edge at Pin20, 0 = no edge

3.5 Instruction Set

The complete instruction set of the PS08 consists of 94 core instructions that have unique op-code decoded by the CPU. Further there is an emulated instruction, no2lcd, that is replaced automatically by the assembler and calls a subroutine in the ROM code.

The variety of the instruction set allows to write comprehensive programs that cope with the 1 K EEPROM.

Branch instructions:
There are 3 principles of jumping within the code:
- Jump. Absolute addressing with 12 Bit within the whole address space.
- Branch. Relative to the actual address with 8 Bit in the range of –128 to +127 bit addresses.
- Skip. jump ahead up to 3 op-codes (3 to 15 bytes).

The assembler puts together jump and branch into new code goto.
It is possible to jump into subroutines only by means of absolute jumps and without any condition.

Arithmetic operations:
The RAM is organized in 24 Bit words. All instructions are based on two's complement operations. A arithmetic command combines two accumulators and writes back the result into the the first mentioned accumulator. The ram address pointer shows the RAM address that is handled in the same way as an accumulator. Each operation on the accumulator

Affects the four flags. The flags refer to the last operation.

Simple Arithmetics	Complex Arithmetics	Shift & Rotate	RAM access
abs	div24	clrC	clear
add	divmod	rotl	decramadr
compare	mult24	rotR	incramadr
compl	mult	setC	move
decr		shiftL	ramadr
getflag		shiftR	swap
incr			
sign			
sub			

Logic	Bitwise	LCD display	EEPROM access
and	bitclr	dez2lcd	equal
eor	nitinv	newlcd	getepr
nor	bitset	no2lcd	putepr
invert			
nand			
nor			
or			

Unconditional jump		Miscellaneous
goto	skip	clk10kHz
gotoBitC	skipBitC	clrwdt
gotoBitS	skipBitS	initTDC
gotoCarC	skipCarC	newcyc
gotoCarS	skipCarS	nop
gotoEQ	skipEQ	stop
gotoNE	skipNE	
gotoNeg	skipNeg	
gotoOvrC	skipOvrC	
gotoOvrS	skipOvrS	
gotoPos	skipPos	
jsub		
jsubret		

abs	**Absolute value of register**
Syntax:	abs p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := | p1 |
Flags affected:	C O S Z
Bytes:	2
Cycles:	2
Description:	Absolute value of register
Category:	Simple arithmetics

add	**Addition**
Syntax:	add p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 + p2
Flags affected:	C O S Z
Bytes:	2 (p2 = ACCU) 4 (p2 = number)
Cycles:	2 (p2 = ACCU) 4 (p2 = number)
Description:	Addition of two registers or addition of a constant to a register
Category:	Simple arithmetics

and	**Logic AND**
Syntax:	and p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 AND p2
Flags affected:	S Z
Bytes:	2 (p2 = ACCU) 5 (p2 = number)
Cycles:	3 (p2 = ACCU) 6 (p2 = number)
Description:	Logic AND of 2 registersn or Logic AND of register and constant
Category:	Logic

bitclr	**Clear single bit**
Syntax:	bitclr p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = number 0 to 23
Calculus:	p1:=p1 and not (1<<p2)
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	Clear a single bit in the destination register
Category:	Bitwise

bitinv	**Invert single bit**
Syntax:	bitinv p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = number 0 to 23
Calculus:	p1:=p1 eor (1<<p2)
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	Invert a single bit in the destination register
Category:	Bitwise

bitset	**Set single bit**
Syntax:	bitset p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = number 0 to 23
Calculus:	p1:=p1 or (1<<p2)
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	Set a single bit in the destination register
Category:	Bitwise

clear	**Clear register**
Syntax:	clear p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := 0
Flags affected:	S Z
Bytes:	1
Cycles:	1
Description:	Clear addressed register to 0
Category:	RAM access

clk10khz	**Clock source 10 kHz**
Syntax:	clk10khz p1
Parameters:	p1 = number 0 or 1
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Change clock source of processor to 10 kHz. The clock of the processor is switched to the slower 10 kHz clock instead of the 40 MHz. The 10 kHz clock is still stable to variations in temperature and supply voltage. If p1 is set to 1 the 10 kHz clock is on, if p1 == 0 the 10 kHz clock is off.
Category:	Miscellaneous

clrC	**Clear flags**
Syntax:	clrC
Parameters:	-
Calculus:	-
Flags affected:	C O
Bytes:	1
Cycles:	1
Description:	Clear Carry and Overflow flags
Category:	Shift and Rotate

clrwdt	**Clear watchdog**
Syntax:	clrwdt
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	
Description:	Clear watchdog. This opcode is used to clear the watchdog at the end of a programme run. Apply this opcode right before 'stop'.
Category:	Miscellaneous

Compare	**Compare two values**
Syntax:	compare p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	---:=p2-p1 only the flags are changed but not the registers
Flags affected:	C O S Z
Bytes:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Description:	Compare of 2 registers by subtraction Compare of a contant with a register by subtraction The flags are changed according to the subtraction result, but not the registers contents themselves
Category:	Simple arithmetic

compl	**Complement**
Syntax:	compl p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := - p1 = not p1 + 1
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	two's complement of register
Category:	Simple arithmetic

decr	**Decrement**
Syntax:	decr p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := p1 – 1
Flags affected:	C O S Z
Bytes:	1
Cycles:	1
Description:	Decrement register
Category:	Simple arithmetic

decramadr	**Decrement RAM address pointer**
Syntax:	decramadr
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Decrement ramaddress pointer by one
Category:	Ram Access

dez2lcd	**Dezimal to segment code**
Syntax:	dez2lcd p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Converts decimal code in register p1 to 7 segment code. A decimal number from 0 to 9 of the addressed register p1 is converted to standard 7 segment code (digits a-h). This opcode may be used for advanced LCD conversions routines, where opcode no2lcd is not

	sufficient dez hgfe dcba 0 --> 0b00111111=0x3F 1 --> 0b00000110=0x06 2 --> 0b01011011=0x5B 3 --> 0b01001111=0x4F 4 --> 0b01100110=0x66 5 --> 0b01101101=0x6D 6 --> 0b01111101=0x7D 7 --> 0b00000111=0x07 8 --> 0b01111111=0x7F 9 --> 0b01101111=0x6F
Category:	LCD Display

div24	**Signed division 24 Bit**
Syntax:	div24 p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r]
Calculus:	p1 := (p1 << 24) / p2 (if |p1| < 2 * |p2|)
Flags affected:	S & Z of p1
Bytes:	2
Cycles:	20
Description:	Signed division of 2 registers 24 bits of the division of 2 registers, result is assigned to p1
Category:	Complex arithmetic

divmod	**Signed modulo division**
Syntax:	divmod p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r]
Calculus:	p1 := p1 / p2 and p2 := p1 % p2
Flags affected:	S Z
Bytes:	2
Cycles:	
Description:	Signed modulo division of 2 registers 24 higher bits of the division of 2 registers, result is assigned to p1 the rest is placed to p2
Category:	Complex arithmetic

eor	**Exclusive OR**
Syntax:	eor p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 xor p2 bitcombination 0 / 0 and 1 / 1 returns 0 bitcombination 0 / 1and 1 / 0 returns 1
Flags affected:	S Z
Bytes:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
Cycles:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
Description:	Logic XOR (eXclusive OR, antivalence) of the 2 given registers Logic XOR (eXclusive OR, antivalence) of register with constant
Category:	Logic

eorn	**Exclusive NOR**
Syntax:	eorn p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 xnor p2 bitcombination 0 / 0 and 1 / 1 return 1 bitcombination 0 / 1 and 1 / 0 return 0
Flags affected:	S Z
Bytes:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
Cycles:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
Description:	Logic XNOR (eXclusive NOR, equivalence) of the 2 given registers Logic XNOR (eXclusive NOR, equivalence) of register with constant
Category:	Logic

equal	**Write 3 Bytes to EEPROM**
Syntax:	equal p1
Parameters:	p1 = 24-Bit number
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	
Description:	Write 3 bytes (p1) to configuration register of EEPROM. The equal opcode is used to write 3 bytes of configuration data directly to an EEPROM register. Therefore the opcode is simply used 16 times in the beginning of the assembler listing, feeded with the configuration data given through p1. Like 'putepr' the configuration of the EEPROM is done in the lower area from byte 0..47, combined in 16x 24bit registers. From byte 48 upwards, the user code is written to the EEPROM. Use this opcode to provide your own configuration instead of the standard configuration.
Category:	EEPROM access

getepr	**Get EEPROM content**
Syntax:	getepr p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := EEPROM register content (addressed by ramaddress pointer)
Flags affected:	S Z
Bytes:	1
Cycles:	6
Description:	Get EEPROM into register. The addressed register p1 gets the EEPROM register content which is addressed by the ramaddress pointer. This opcode needs temporarily a place in the program counter stack (explanation see below).
Category:	EEprom Access

getflag	**Set S and Z flags**
Syntax:	getflag p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	signum := set if p1 < 0 notequalzero := set if p1 <> 0
Flags affected:	S Z
Bytes:	1
Cycles:	1
Description:	Set the signum and notequalzero flag according to the addressed register, content of the register is not affected
Category:	Simple arithmetic

goto	**Jump without condition**
Syntax:	goto p1
Parameters:	p1 = JUMPLABEL
Calculus:	PC:= p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump without condition. Program counter is set to target address. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Unconditional jump

gotoBitC	**Jump on bit clear**
Syntax:	gotoBitC p1, p2, p3
Parameters:	p1 = ACCU [x,y,z,r] p2 = NUMBER [0..23] p3 = JUMPLABEL
Calculus:	if (bit p2 of register p1 == 0) PC := p3
Flags affected:	-
Bytes:	3
Cycles:	4
Description:	Jump on bit clear. Program counter will be set to target address if selected bit in register p1 is clear. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Bitwise

gotoBitS	**Jump on bit set**
Syntax:	gotoBitS p1, p2, p3
Parameters:	p1 = ACCU [x,y,z,r] p2 = NUMBER [0..23] p3 = JUMPLABEL
Calculus:	if (bit p2 of register p1 == 1) PC := p3
Flags affected:	-
Bytes:	3
Cycles:	4
Description:	Jump on bit set. Program counter will be set to target address if selected bit in register p1 is set. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Bitwise

gotoCarC	**Jump on carry clear**
Syntax:	gotoCarC p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (carry == 0) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on carry clear. Program counter will be set to target address if carry is clear. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoCarS	**Jump on carry set**
Syntax:	gotoCarS p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (carry == 1) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on carry set. Program counter will be set to target address if carry is set. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoEQ	**Jump on equal zero**
Syntax:	gotoEQ p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (Z == 0) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on equal zero. Program counter will be set to target address if the foregoing result is equal to zero. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoNE	**Jump on not equal zero**
Syntax:	gotoNE p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (Z == 1) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on not equal zero. Program counter will be set to target address if the foregoing result is not equal to zero. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoNeg	**Jump on negative**
Syntax:	gotoNeg p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (S == 1) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on negative. Program counter will be set to target address if the foregoing result is negative. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoOvrC	**Jump on overflow clear**
Syntax:	gotoOvrC p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (O == 0) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on overflow clear. Program counter will be set to target address if the overflow flag of the foregoing operation is clear. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoOvrS	**Jump on overflow set**
Syntax:	gotoOvrS p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (O == 1) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on overflow set. Program counter will be set to target address if the overflow flag of the foregoing operation is set. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

gotoPos	**Jump on possitive**
Syntax:	gotoPos p1
Parameters:	p1 = JUMPLABEL
Calculus:	if (S == 0) PC := p1
Flags affected:	-
Bytes:	2 (relative jump) 3 (absolute jump)
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump on possitive. Program counter will be set to target address if the foregoing result is possitive. The target address is given by using a jumplabel. See examples section for how to introduce a jumplabel.
Category:	Goto on flag

incr	**Increment**
Syntax:	incr p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := p1 + 1
Flags affected:	C O S Z
Bytes:	1
Cycles:	1
Description:	Increment register
Category:	Simple arithmetic

incramadr	**Increment RAM address**
Syntax:	incramadr
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Increment RAM address pointer by 1
Category:	RAM access

initTDC	**Initialize TDC**
Syntax:	initTDC
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Initialisation reset of the TDC (time-to-digital converter). Should be sent after configuration of registers. The initreset preserves all configs .
Category:	Miscellaneous

invert	**Bitwise inversion**
Syntax:	invert p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := not p1
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	Bitwise inversion of register
Category:	Logic

jsub	**Unconditional jump**
Syntax:	jsub p1
Parameters:	p1 = JUMPLABEL
Calculus:	PC := p1
Flags affected:	C O S Z
Bytes:	3
Cycles:	4
Description:	Jump to subroutine without condition. The programm counter is loaded by the address given through the jumplabel. The subroutine is processed until the keyword 'jsubret' occurs. Then a jump back is performed and the next command after the jsub-call is executed. This opcode needs temporarily a place in the program counter stack (explanation see below).
Category:	Unconditional Jump

jsubret	**Return from subroutine**
Syntax:	jsubret
Parameters:	-
Calculus:	PC := PC from jsub-call
Flags affected:	-
Bytes:	1
Cycles:	3
Description:	Return from subroutine. A subroutine can be called via 'jsub' and exited by using jsubret. The program is continued at the next command following the jsub-call. You have to close a subroutine with jsubret - otherwise there will be no jump back.
Category:	Unconditional Jump

move	**Move**
Syntax:	move p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-bit number
Calculus:	p1 := p2
Flags affected:	S Z
Bytes:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Description:	Move content of p2 to p1 (p1=ACCU, p2=ACCU) Move constant to p1 (p1=ACCU, p2=NUMBER)
Category:	RAM access

mult24	**Signed 24-Bit multiplication**
Syntax:	mult24 p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r]
Calculus:	p1 := (p1 * p2) >> 24
Flags affected:	S & Z of p1
Bytes:	2
Cycles:	30
Description:	Signed multiplication of 2 registers like mult48, but only the 24 higher bits of the multiplication of 2 registers, result is stored in p1
Category:	Complex arithmetic

mult	**Signed 48-Bit multiplication**
Syntax:	mult48 p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r]
Calculus:	p1,p2 := p1 * p2
Flags affected:	S & Z of p1
Bytes:	2
Cycles:	30
Description:	Signed multiplication of 2 registers Higher 24 bits of the multiplication is placed to p1 Lower 24 bits of the multiplicatin is placed to p2
Category:	Complex arithmetic

nand	**Logic NAND**
Syntax:	nand p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p1 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 nand p2 returns only 0 in case of bitcombination 1 / 1
Flags affected:	S Z
Bytes:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
Cycles:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
Description:	Logic NAND (negated AND) of the 2 given registers Logic NAND (negated AND) of register with constant
Category:	Logic

newcyc	**Start TDC**
Syntax:	newcyc
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Start of TDC. This opcode can be used after configuration and initialisation of the PS08 to start a new measurement cycle. Normally this is done by the PS08 rom routines itself, but in case of custom-designed reset procedures this opcode can play a role.
Category:	Miscellaneous

newlcd	**Load new LCD data**
Syntax:	newlcd
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Load new LCD data. New segment data from register 61-63 is written to the LCD driver. Refreshes the display.
Category:	LCD display

no2lcd	**Covert 6 digits to LCD code**
Syntax:	no2lcd
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	subroutine call
Description:	Converts the 6 digits of the dezimal number in register x into 7-segment code. The positon of the decimal point is determined with the number (0..5) in register y. Leading zeros before the decimal point are cleared. The conversion result is directly written to the LCD register 61-62 Use this opcode in combination with 'newlcd'. In real a subroutine in the ROM code is called. The assembler converts this command to the corresponding jump command.
Category:	LCD display

nop	**No operation**
Syntax:	-
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Placeholder code or timing adjust (no function)
Category:	Miscellaneous

nor	**Logic NOR**
Syntax:	nor p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 nor p2 returns only 1 in case of bitcombination 0 / 0
Flags affected:	S Z
Bytes:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
Cycles:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
Description:	Logic NOR (negated OR) of the 2 given registers Logic NOR (negated OR) of register with constant
Category:	Logic

or	**Logic OR**
Syntax:	or p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r] or 24-Bit number
Calculus:	p1 := p1 or p2 returns only 0 in case of bitcombination 0 / 0
Flags affected:	S Z
Bytes:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
Cycles:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
Description:	Logic OR of the 2 given registers Logic OR of register with constant
Category:	Logic

putepr	**Put register to EEPROM**
Syntax:	putepr p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	EEPROM register (addressed by ramaddress pointer) := p1
Flags affected:	-
Bytes:	4
Cycles:	65536 = ~50ms
Description:	Put register into EEPROM. The content of the addressed register p1 is moved to the EEPROM (the EEPROM register address is set by the ramaddress pointer). Only EEPROM register 14 and 15 are accessible via 'putepr'. This opcode needs temporarily a place in the program counter stack (explanation see below). 'putepr' should not be combined with the skip-opcodes due to the long execution time of this opcode (approx.: 50ms)
Category:	EEPROM access

ramadr	**Set RAM address pointer**
Syntax:	ramadr p1
Parameters:	p1 = 5-Bit number
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Set pointer to ramaddress (range: 0..65)
Category:	RAM access

rotL	**Rotate left**
Syntax:	rotL p1(,p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none
Calculus:	p1 := p1<< 1+ carry; carry:=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) rotL p1 (in case rotL p1,p2)
Flags affected:	C O S Z (of the last step)
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
Description:	Rotate p1 left --> shift p1 register to the left, fill LSB with carry, MSB is placed in carry register Rotate p1 left p2 times with carry --> shift p1 register p2 times to the left, in each step fill LSB with the carry and place the MSB in the carry
Category:	Shift and rotate

rotR	**Rotate right**
Syntax:	rotR p1(,p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none
Calculus:	p1 := p1>> 1+ carry; carry: =MSB(x) (in case rotR p1, without p2) p1 := repeat (p2) rotR p1 (in case rotR p1,p2)
Flags affected:	C O S Z (of the last step)
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
Description:	Rotate p1 right --> shift p1 register to the right, fill MSB with carry, LSB is placed in carry register Rotate p1 right p2 times with carry --> shift p1 register p2 times to the right, in each step fill MSB with the carry and place the LSB in the carry
Category:	Shift and rotate

setC	**Set carry flag**
Syntax:	setC
Parameters:	-
Calculus:	-
Flags affected:	C O
Bytes:	1
Cycles:	1
Description:	Set carry flag and clear overflow flag
Category:	Shift and Rotate

shiftL	**Shift Left**
Syntax:	shiftL p1,(p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none
Calculus:	p1 := p1<< 1; carry :=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) shiftL p1 (in case rotL p1,p2)
Flags affected:	C O S Z
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1 + p2 (p1=ACCU, p2=NUMBER)
Description:	Shift p1 left --> shift p1 register to the left, fill LSB with 0, MSB is placed in carry register Shift p1 left p2 times --> shift p1 register p2 times to the left, in each step fill LSB with the 0 and place the MSB in the carry
Category:	Shift and rotate

shiftR	**Shift right**
Syntax:	shiftR p1,(p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none
Calculus:	p1 := p1>> 1; carry:=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) shiftL p1 (in case rotL p1,p2)
Flags affected:	C O S Z
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1 + p2 (p1=ACCU, p2=NUMBER)
Description:	Signed shift right of p1 --> shift p1 right, MSB is duplicated according to whether the number is positive or negative Signed shift p1 right p2 times --> shift p1 register p2 times to the right, MSB is duplicated according to whether the number is positive or negative
Category:	Shift and rotate

sign	**Sign**
Syntax:	sign p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := p1 / | p1 | p1 := 1 = 0x000001 if p1 >= 0 p1 := -1 = 0xFFFFFF if p1 < 0
Flags affected:	S Z
Bytes:	2
Cycles:	2
Description:	Sign of addressed register in complement of two notation. A positive value returns 1, a negative value returns -1 Zero is assumed to be positiv
Category:	Simple arithmetic

skip	**Skip**
Syntax:	skip p1
Parameters:	p1 = NUMBER [1,2,3]
Calculus:	PC := PC + bytes of next p1 lines
Flags affected:	
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 without conditions
Category:	Unconditional jump

skipBitC	**Conditional skip**
Syntax:	skipBitC p1,p2,p3
Parameters:	p1 = ACCU [x,y,z,r] p2 = NUMBER[0..23] p2 = NUMBER[1,2,3]
Calculus:	if (bit p2 of register p1 == 0) PC := PC + bytes of next p3 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p3 commands if bit p2 of register p1 is clear
Category:	Bitwise

skipBitS	**Conditional skip**
Syntax:	skipBitS p1,p2,p3
Parameters:	p1 = ACCU [x,y,z,r] p2 = NUMBER[0..23] p2 = NUMBER[1,2,3]
Calculus:	if (bit p2 of register p1 == 1) PC := PC + bytes of next p3 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p3 commands if bit p2 of register p1 is set
Category:	Bitwise

skipCarC	**Skip carry clear**
Syntax:	skipCarC p1
Parameters:	p1 = NUMBER [1,2,3]
Calculus:	if (carry == 0) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if carry clear
Category:	Skip on flag

skipCarS	**Skip carry set**
Syntax:	skipCarS p1
Parameters:	p1 = NUMBER [1,2,3]
Calculus:	if (carry == 1) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if carry set
Category:	Skip on flag

skipEQ	**Skip on zero**
Syntax:	skipEQ p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (notequalzero == 0) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if result of previous operation is equal to zero
Category:	Skip on flag

skipNE	**Skip on non-zero**
Syntax:	skipNE p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (notequalzero == 1) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if result of previous operation is not equal to zero
Category:	Skip on flag

skipNeg	**Skip on negative**
Syntax:	skipNeg p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (signum == 1) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if result of previous operation was smaller than 0
Category:	Skip on flag

skipOvrC	**Skip on overflow**
Syntax:	skipOvrC p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (overflow == 0) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if overflow is clear
Category:	Skip on flag

skipOvrS	**Skip on overflow**
Syntax:	skipOvrS p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (overflow == 1) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if overflow is set
Category:	Skip on flag

skipPos	**Skip on possitive**
Syntax:	skipPos p1
Parameters:	p1 = NUMBER[1,2,3]
Calculus:	if (signum == 0) PC := PC + bytes of next p1 lines
Flags affected:	-
Bytes:	1
Cycles:	1 + skipped commands
Description:	Skip p1 commands if result of previous operation was greater or equal to 0
Category:	Skip on flag

stop	**Stop**
Syntax:	stop
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	Stop of the converter The clock generator is stopped, the converter and the EEPROM go to standby. A restart of the converter can be achieved by an external event like 'watchdog timer', 'external switch' or 'new strain measurement results'. Usually this opcode is the last command in the assembler listing.
Category:	Miscellaneous

sub	**Substraction**
Syntax:	sub p1,p2
Parameters:	p1 = NUMBER[1,2,3] p2 = NUMBER[1,2,3] or 24-Bit number
Calculus:	p1:= p2 – p1
Flags affected:	C O S Z
Bytes:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Description:	Subtraction of 2 registers Subtraction of register from constant
Category:	Simple arithmetic

swap	**Swap**
Syntax:	swap p1,p2
Parameters:	p1 = ACCU [x,y,z,r] p2 = ACCU [x,y,z,r]
Calculus:	p1 := p2 and p2 := p1
Flags affected:	-
Bytes:	1
Cycles:	3
Description:	Swap of 2 registers The value of two registers is exchanged between each other
Category:	RAM Access

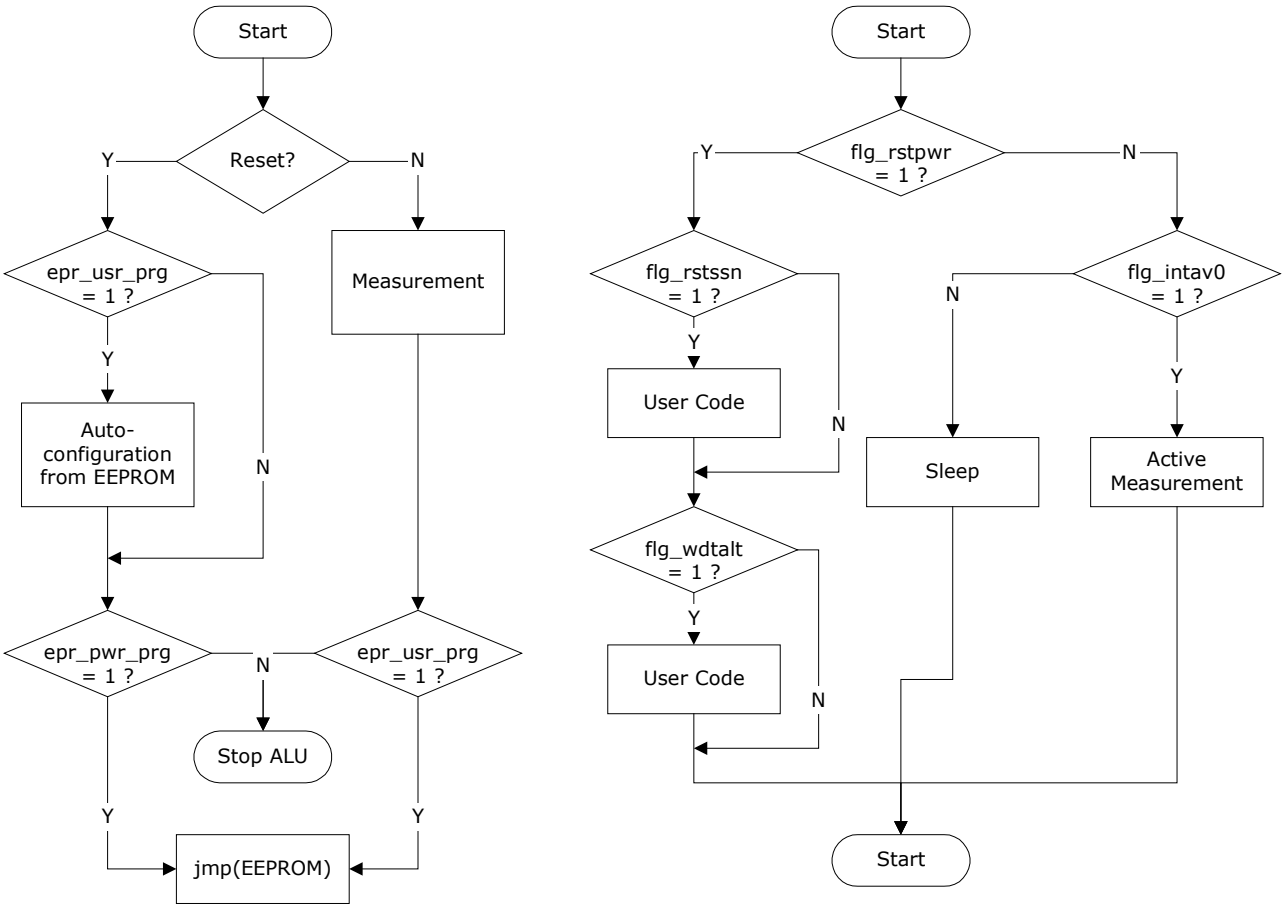
4 System Reset, Sleep Mode and Autoconfiguration

ALU activity is requested by a reset or the TDC (end of measurement). A reset has priority of a TDC request. The ALU jumps after activation into the ROM code starting with address 1024.
There a first check is done whether the ALU was activated after a reset or not.

In case of a reset the flag epr_usr_prg is checked to decide whether the autoconfiguration data from the EEPROM have to be copied into the RAM or not. In the following flag epr_pwr_prg is checked to decide whether EEPROM code (starting at address 48) shall be executed. if yes, the program jumps into the EEPROM, otherwise the µP is stoppped.

In case the ALU is started by sleep mode and not by a reset, the TDC unit starts a measurement. Afterwards the flag epr_usr_prg is checked to decide about a jump into the EEPROM (address 48).

In the EEPROM code a first check should be done for the flg_rstpwr flag to see whether the reason for the jump was a reset. If yes, a detailed check for the cause should be done with the consequential user code.
Otherwise a check of flag flg_intav0 will indicate an active strain measurement.



At the end the ALU is stopped. This implements a complete reset of the ALU including the start flags. Als the program stack is reset. ONly the RAM data remain unchanged.

4.1 Power On Reset
When applying the supply voltage to the chip a power-on reset is generated. The whole chip is resetted, only the RAM emains unchanged.

In case bit epr_pwr_prg is set to 1 (bit1, register 1) the EEPROM program code beginning at address 48 is started.

4.2 Watchdog Reset
A power-on reset can also be triggered by the watchdog timer. This happens in case the microprocessor is started four times without being reset by the opcode "clrwdt". Status bit flg_wdtalt in register 22, bit 17, indicates a timeout of the watchdog timer.
In case bit epr_pwr_prg is set to 1 (bit1, register 1) the EEPROM program code beginning at address 48 is started.

4.3 External Reset on Pin 27
In stand-alone mode (SPI_ENA = 0) it is possible to apply an external power-on at pin 27 (SPI_CSN_RST). This can be used for a reset button. The status of the button can be requested from status bit flg_rstssn in register 22, bit 18.

In case bit epr_pwr_prg is set to 1 (bit1, register 1) the EEPROM program code beginning at address 48 is started.

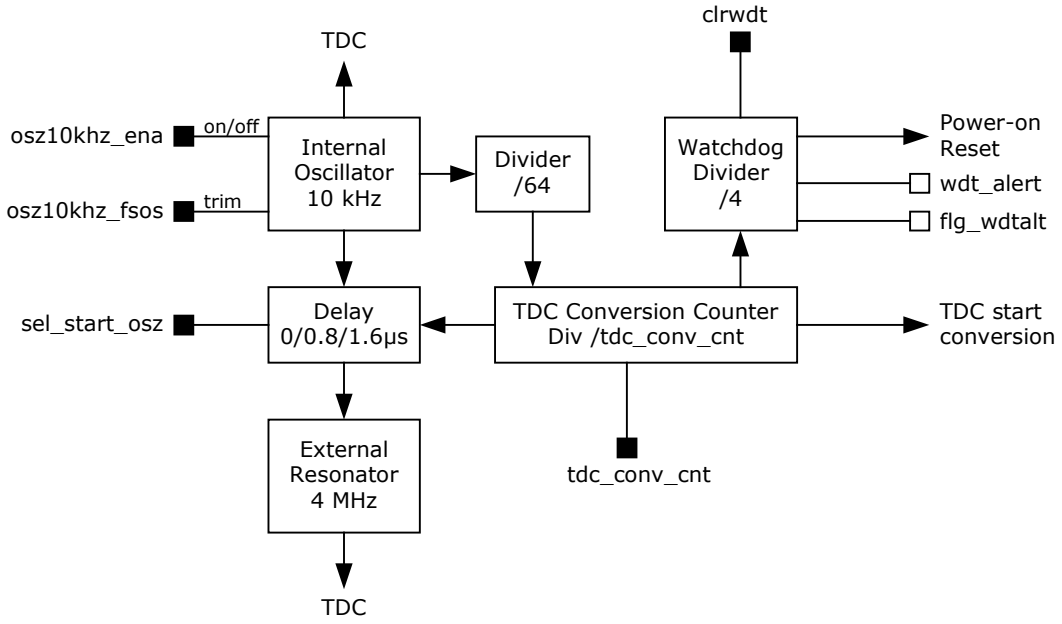
4.4 Sleep Mode
In the sleep mode only the 10 kHz oscillator is running. At regular intervals the microprocessor is waked up but without doing a measurement. In this phase it can check the I/O's.
A start-up of the microprocessor from sleep mode is indicated by status bit flg_intav0 in register 22, bit 22.

Configuration:
avrate[9:0]=0 Sleep mode is activated by setting avrate = 0 (Reg2, Bits 23 to 14)
tdc_conv_cnt[11:0] In sleep mode counter tdc_cnv_cnt (Reg0, Bits 23 to 14) is running to the end and then immediatedly the program starting at address 48 in the EEPROM is started.

Initialisation:
The TDC has to be reinitialized after programming the sleep mode.
Example:

```
ramadr 48+2          Konfigregister 2
and r 0x0003ff       AvRate=0
initdc          Reset der TDC
newcyc               Neustart des TDC
```

5 CPU Clock generation
The basic clock for the system is the internal, low-current 10 kHz oscillator. It is used to trigger measurements in single conversion mode. It is also used for the TDC unit in measurement range 2 as pre-counter.



5.1 Watchdog counter and Single conversion counter

The single conversion counter starts a measurement in single conversion mode. It is running continuously. It is stopped only by setting bit osz10khz_ena. The single conversion rate is given by 10kHz / 64 / tdc_conv_cnt.

With the beginning of a measurement the watchdog counter is increased. The watchdog counts the conversions. At the end of a measurement the microprocessor starts to run the user code. In normal operation the watchdog by CLRWDT has to be reset before the user code ends. The watchdog causes a power-on reset in case the TDC doesn't finish its measurement because of an error or the EEPROM code does not run to end.

It is possible to switch off the watchdog when controlloing the PS08 by the SPI interface (SPI_ENA = 1) sending spi opcode watch_dog_off. Further the watchdog is reset by each signal edge at the SPI_CSN_RST pin.

6 IO-pins

PS08 has 5 I/O pins: SPI_DO_IO0, SPI_DI_IO1, SPI_CLK_IO2, SEL_WHEAT_IO3 and SPI_CSN_RST.

Pins SPI_DO_IO0, SPI_DI_IO1, SPI_CLK_IO2, SEL_WHEAT_IO3 and SPI_CSN_RST can be programmed as inputs or outputs with pull-up or pull-down resistors in case the chip is in stanb-alone mode (SPI interface not used, SPI_ENA=1). PIN SEL_WHEAT_IO3 can be used as input only when Wheatstone mode is not used.

Pin 27, SPI_CSN_RST can be used as reset input in case the SPI interface is not used (SPI_ENA = 0). The input then needs an external pull-down resistor to GND. The reset is high active.

6.1 Configuration

Pin29 SEL_WHEAT_IO3	Configreg_11, bit 22,23	io_en_3_mio
Pin22 SPI_CLK_IO2	Configreg_11, bit 20,21	io_en_3_sck

Pin21 SPI_SDI_IO1 Configreg_11, bit 18,19 io_en_3_sdi
Pin20 SPI_SDO_IO0 Configreg_11, bit 16,17 io_en_3_sdo
Port definition 00 = input
01 = input with pull-down
10 = input with oull-up
11 = output

6.2 Output – write

The ouputs are set in configuration register 1.

Pin29 SEL_WHEAT_IO3	Configreg_01, bit 13	io_a[3]
Pin22 SPI_CLK_IO2	Configreg_01, bit 12	io_a[2]
Pin21 SPI_SDI_IO1	Configreg_01, bit 11	io_a[1]
Pin20 SPI_SDO_IO0	Configreg_01, bit 10	io_a[0]

6.3 Input – read

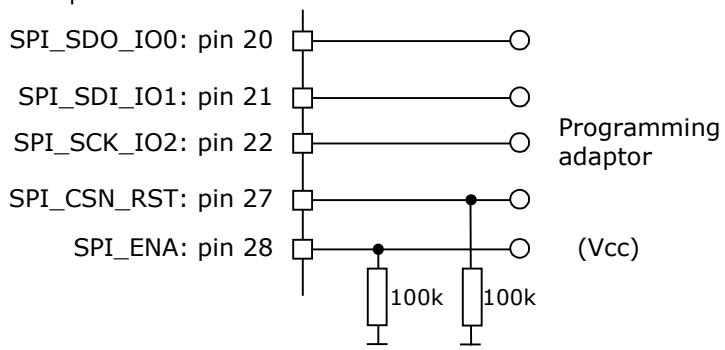
Status[23]= flg_io3_mio Pin29
Status[22]= flg_io2_sck Pin22
Status[21]= flg_io1_sdi Pin21
Status[20]= flg_io0_sdo Pin20

Status[07]= flg_io3_mio_r Rising edge at Pin29
Status[06]= flg_io2_sck_r Rising edge at Pin22
Status[05]= flg_io1_sdi_r Rising edge at Pin21
Status[04]= flg_io0_sdo_r Rising edge at Pin20
Status[03]= flg_io3_mio_f Falling edge at Pin29
Status[02]= flg_io2_sck_f Falling edge at Pin22
Status[01]= flg_io1_sdi_f Falling edge at Pin21
Status[00]= flg_io0_sdo_f Falling edge at Pin20

7 SPI-Interface

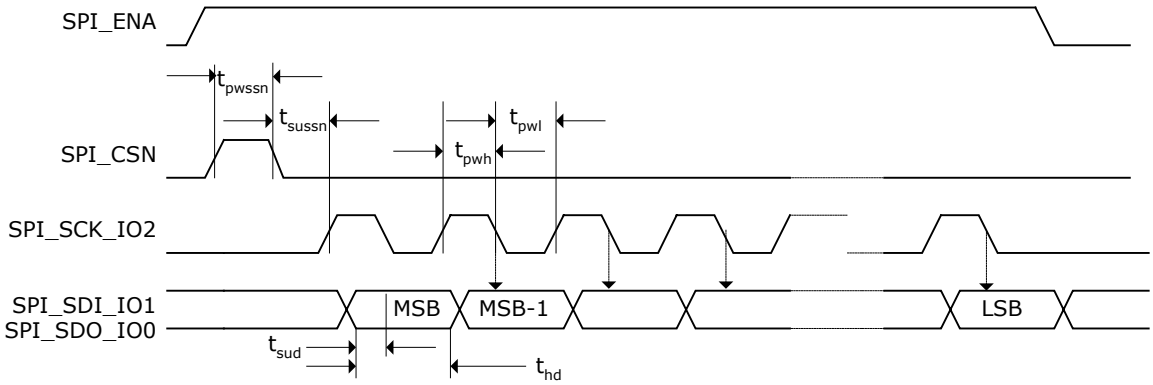
7.1 Interfacing

The SPI interface is used to write the program, configuration data and calibration data into the EEPROM.
It can further be used to operate the PS08 as an pure converter chip by means of an external microcontroller. In this case the pull-down resistors are no longer necessary.
Pulling SPI_ENA high switches the SPI interface on, the pins are used for the SPI interface and no longer as I/O ports. It is necessary to send a possitive pulse on the CSN line before each opcode.



7.2 SPI Timing

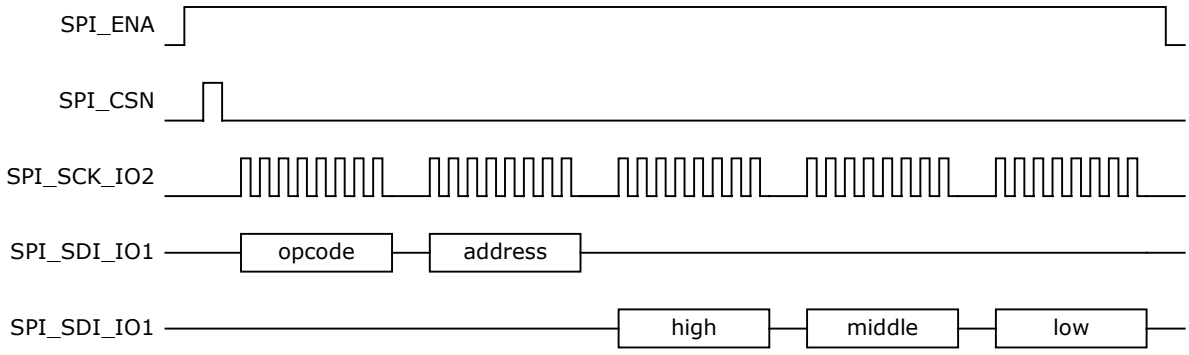
Here we describe only the SPI timing for operation as a pure converter that communicates with an external microcontroller.
PS08 supports only 1 mode out of 4 possible ones:
Clock Phase Bit = 1,Clock Polarity Bit = 0
Data transfer with the falling edge of the clock. The clock starts from low.



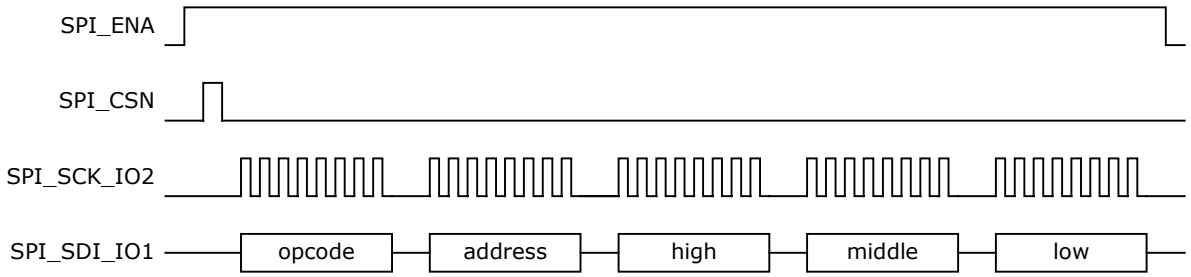
7.3 SPI-Instructions

RAM Write = b00000000
RAM Read = b01000000
Powerreset = b01111000
Initreset = b11000000
Start_new_cycle = b11000110
Start_TDC-cycle = b11000111
watch_dog_off = b10011110
watch_dog_on = b10011111

7.2.1 Read-Access



7.2.2 Write_Access



8 LCD-Driver

The LCD driver has the following features:
18 pins for
- 1/4 duty with maximum 6 digits including comma and 8 special characters
- 1/3 duty with maximum 5 digits inculding comma and 5 special characters
- 1/2 duty with maximum 4 digits including comma

Stabilization of the display voltage to 3 V, 2.5V and 2V
Integrated voltage doubler for 3 V and 2.5 V displays
Energy effiecient 2 V operation without voltage doubling
Operation at unstabilized supply voltage like lithium batteries or solar cells
Currentless stand-by
Driver strength adjustable to segment size and current consumption
Outputs free configurable so that already wired displays can be connected
Implemented conversion tables for 7 segment digits
Implemented ROM code for 24 Bit number conversions

8.1 Basic Configuration

With lcd_duty the LCD is switched and set to a specific multiplex mode
lcd_duty = 0 off
= 1 2x multiplex
= 2 3x multiplex
= 3 4x multiplex mode

lcd_freq controls the switch-on time of the pixels. The longer a pixel is on the less current is needed because of the lower number of reloads. For a flickerfree display an update rate > 30 Hz is recommended. Therefore the switch-on time depends on the selected multiplex mode.

lcd_freq[2:0] =	Pixel on-time	Multiplex mode 1/4	1/3	1/2
0	8.0 ms	15	20	31 Hz
1	4.8 ms	26	34	52 Hz
2	4.0 ms	31	42	62 Hz
3	3.2 ms	30	52	78 Hz
4	2.4 ms	52	69	104 Hz
5	2.0 ms	62	82	125 Hz
6	1.6 ms	78	104	176 Hz
7	1.2 ms	104	138	208 Hz

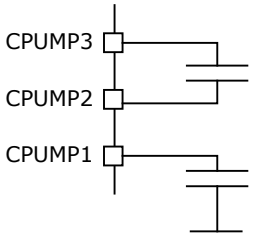
The display has a stand-by mode. I this mode the display is switched off, but the voltage generation is switched high resistive..So it is possible to switch on the display very fast. This might be helpful in auto-on mode.
lcd_standby = 0 LDC on
= 1 Standby

8.2 LCD-Power supply

The PS08 has an integrated charge pump to double and stabilize the voltage for driving 3 V and 2.5 V LCD displays. 2 V displays can easily be driven directly from the power supply. The choice for the external capacitors depends on the size or capacitance of the display.

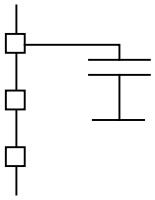
lcd_vlt[1:0] = 0 2.0 V
= 1 2.5 V
= 2 3.0 V

3 V and 2.5 V operation with voltage doubling



In a first step both capacitors are charged to half the display voltage, 1.5 V or 1.25 V. This voltage can be seen at pins CPUMP1 and CPUMP3. In a second step both capacitors are switched into series. Pin CPUMP3 then shows the full lcd voltage while pins CPUMP1 and CPUMP2 show half th lcd voltage.

2 V operation without voltage doubling



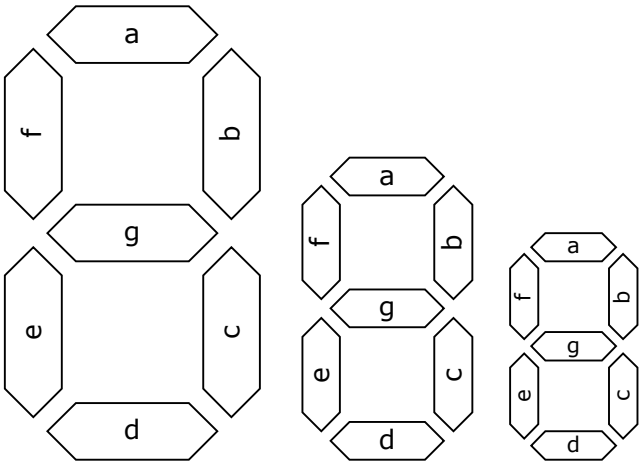
In this mode the lcd voltage is stabilized from an unstabilzed supply voltage charging the capacitor to the lcd voltage. The supply voltage may not drop below 2 V in this mode.

A third option is to drive the LCD directly from the power supply without regulation. Therefore no external capacitors are connected and the output drivers are set low resistive. This is done setting
lcd_r_const = 0 (10 kOhm).
lcd_vlt[1:0] = 0 2.0 V

8.3 LCD Output Driver configuration

The internal resistance of the output drivers can be adopted to the size of the display. By this means the current consumption can be optimized. The size of the display influences

- The size of the capacitors
- The inner resistance of the drivers
- The minimum charge time of the charge pump
- The reload time of the display



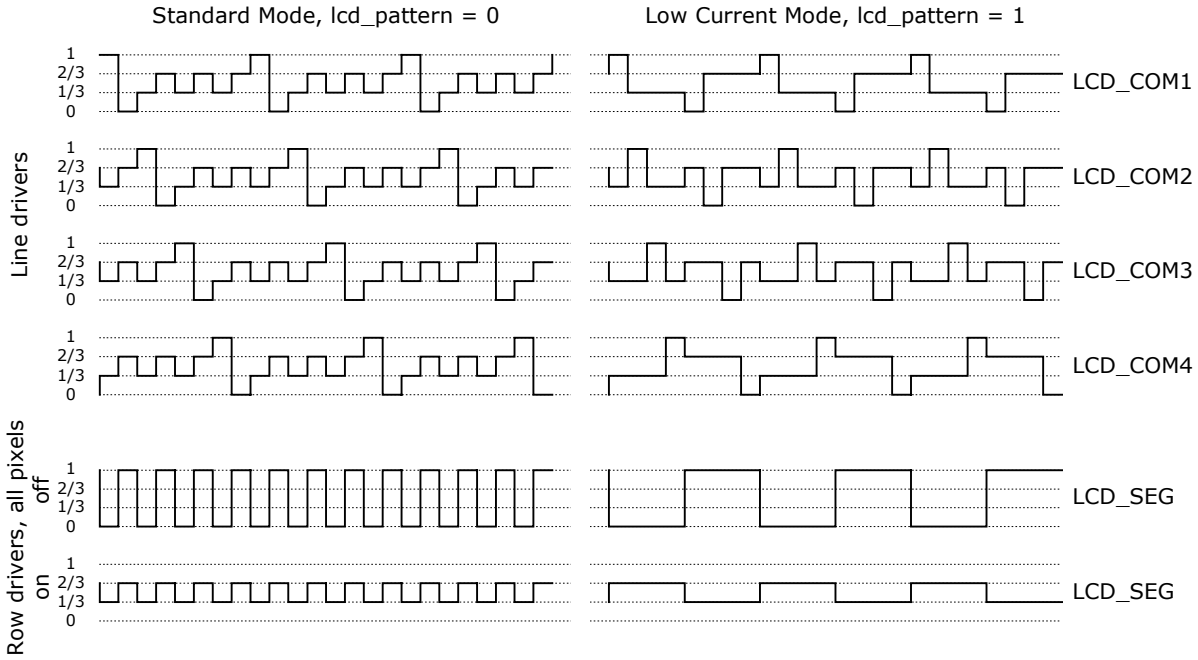
> 5cm big | 5 to 3 cm medium | < 3 cm small

Configuration

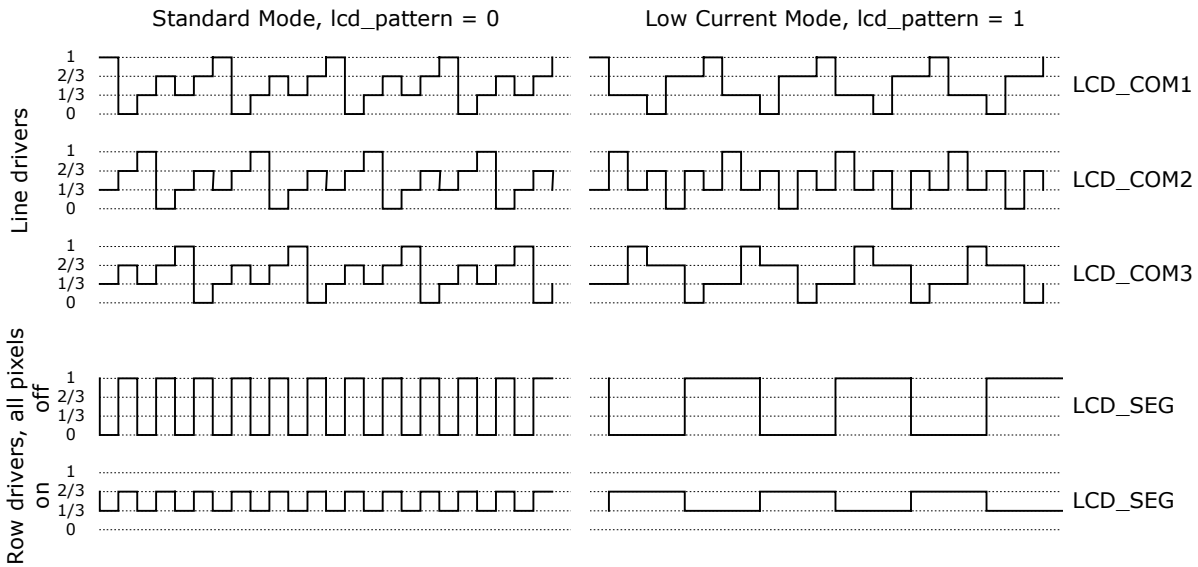
Config.bit	big	medium	small	Function
lcd_fastld[1:0]	3	2	1	Configures the number of fastload periods (10ms) with low-ohmic voltage divider
lcd_swload1k	1	1	0	0 = charge capacitors by 200 Ohm resistors 1 = charge capacitors by 1 kOhm resistors
lcd_r_const[1:0] voltage	1	2	3	Defines the cross resistance of the LCD divider 0 = 15 k 1 = 200 k 2 = 800 k 3 = 1600 k
lcd_charge[1:0]	0	2	3	Selects how many lcd clock cycles it is waited before recharging 0 = each cycle 1 = second cycle 2 = fourth cycle
lcd_r_fastld periods(10ms)	1	2	3	Configures the number of fastload with low-ohmic voltage divider
Capacitors	10 nF	33 nF	100 nF	

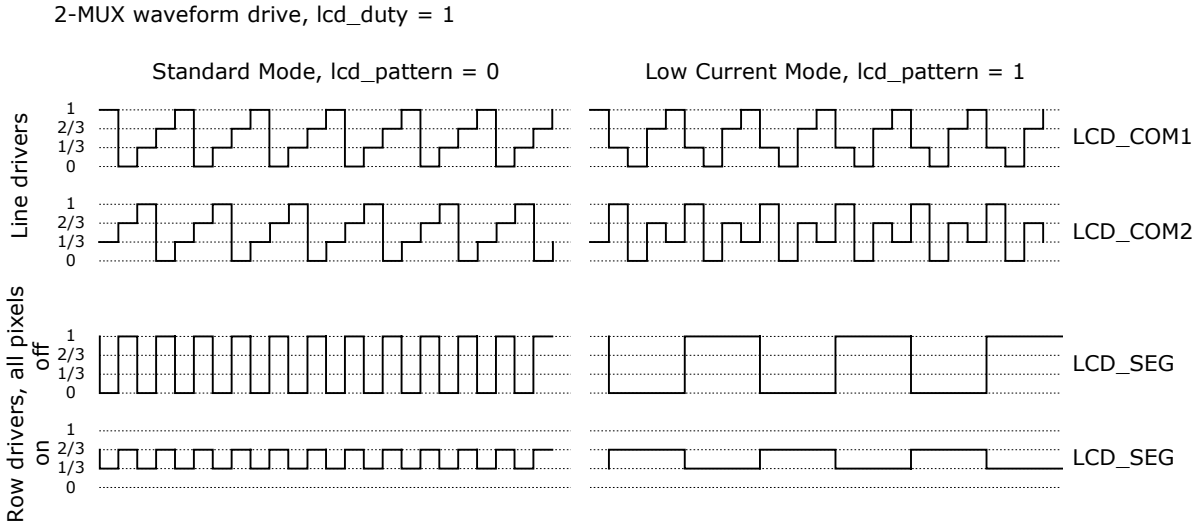
LCD driving methods
In general there are two driving methods for each multiplex mode. A standard one (Alt-Pleskho) and a current reduced one. The last one is selected by setting
lcd_pattern = 1
In each mode the outputs drive 4 voltage leves, 0, 1/3, 2/3 and full lcd voltage.

4-MUX waveform drive, lcd_duty = 3



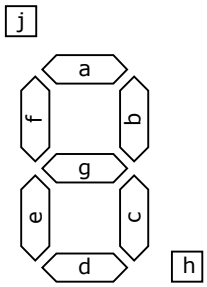
3-MUX waveform drive, lcd_duty = 2





8.4 LCD Control

8 segments are form a digit. Each segment is named by a charater from a to g. The dot is named h.



The single segments are switched on or off by setting the bits in the configuration registers 13, 14 and 15 to "1" or "0". The assignemnt does not depend from the multiplex mode. It looks like the following:

Digit	Segment		Hex Value
	hgfe	dcba	
"0"	0011	1111	3F
"1"	0000	0110	06
"2"	0101	1011	5B
"3"	0100	1111	4F
"4"	0110	0110	66
"5"	0110	1101	6D
"6"	0111	1101	7D
"7"	0000	0111	07
"8"	0111	1111	7F
"9"	0110	1111	6F

Position in the Configuration Memory
In 2x multiplex the lower 32 bit of lcd_segment are used.

in 3x multiplex each digit is represented by a 3x3 matrix, including one additional special character. The lower 40 bits of lcd_segment are used for the 5 digits. The special signs are

controlled by bits 40 to 44.

Digit	lcd_segment	configreg	Used with
6	[55:48]	15	1/4
5	[47:40]	14	1/4, 1/3
4	[39:32]	14	1/4, 1/3
3	[31:24]	14	1/4, 1/3,1/2
2	[23:16]	13	1/4, 1/3,1/2
1	[15:8]	13	1/4, 1/3,1/2
0	[7:0]	13	1/4, 1/3,1/2

With dez2lcd (D) there is a special code for the processor to convert decimal data to characters 0 to 9. It converts the lowest four bit of the addressed accumulator (representing 0 to 9) into standard 7 segment code.

For further comfort, in the ROM code there is a subroutine for a complete conversion of a 24 bit number.
In the assembler the subroutine is represented by opcode no2lcd. The value of the X-accumulator is converted and written into the lower 48 bit of the LCD memory (lcd_segment[39:0]. The signed original is written back to the X-accumulator and can be used to set the sign on the display. The position of the comma is shown in the Y-accumulator. Leading zero's are suppressed. The LCD driver ingores the upper 2 digits in 2x multplex and the upper digit in 3x multiplex. The special characters in 3x and 4x multiplex will not be changed (lcd_segment[55:48]). In 2xmultiplex the comma of the display might be used for special characters. In this case they must be restored after the conversion.

It is necessary to inform the LCD driver separately about new data in the lcd register 13 to 15. This is done by opcode newlcd.

Sample program:

```
move    x , r          ; Load x-accumulator with the result
move    y , 2          ; Load y-accumulator with the comma position
jsub    no2lcd         ; Convert into 7-Segment display format
  newlcd               ; Update LCD
  clrwdt               ; Set back the watchdog
stop                         ; Stop the uC
```

8.5 Setting the Segment Position

Each segment of the configuration bits lcd_segment can be linked to an arbitrary crossing of the line and common drivers. This offers a high flexibility and allows to connect existing LCDs independent from their current wiring. Each segment is assigned 3 bits to specify one out of the 8 tartget segments.

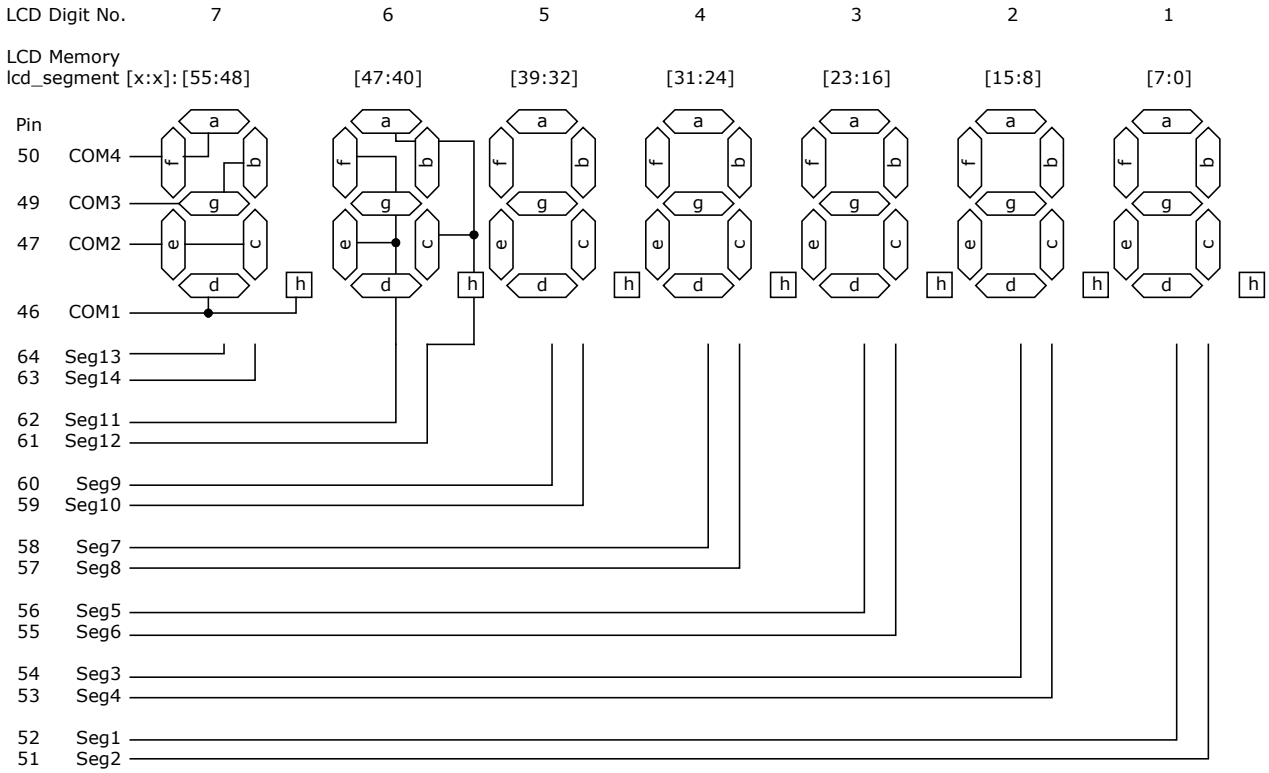
Segment	lcd_pos[]	lcd_segment[]
h	23 to 21	7, 15, 23, 31, 39, 47, 55
g	20 to 18	6, 14, 22, 30, 38, 46, 54
f	17 to 15	5, 13, 21, 29, 37, 45, 53
e	14 to 12	4, 12, 20, 28, 36, 44, 52
d	11 to 9	3, 11, 19, 27, 35, 43, 51
c	8 to 6	2, 10, 18, 26, 34, 42, 50
b	5 to 3	1, 9, 17, 25, 33, 41, 49
a	2 to 0	0, 8, 16, 24, 32, 40, 48

The PS08 is adjusted to an LCD by the following method:

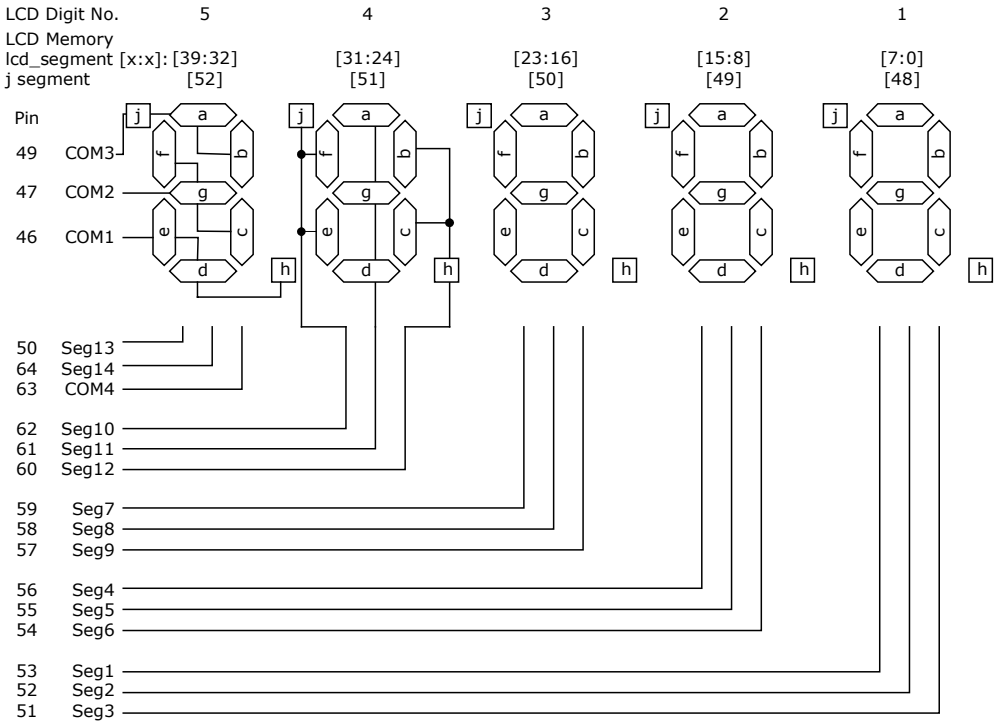
1. Switch on the LCD
2. Switch on the first segment a (Bit0 in config_reg31)
3. Set lcd_pos[2:0] so that the correct segment is on.
4. Repeat this for the remaining segments

8.6 Connecting Schemes

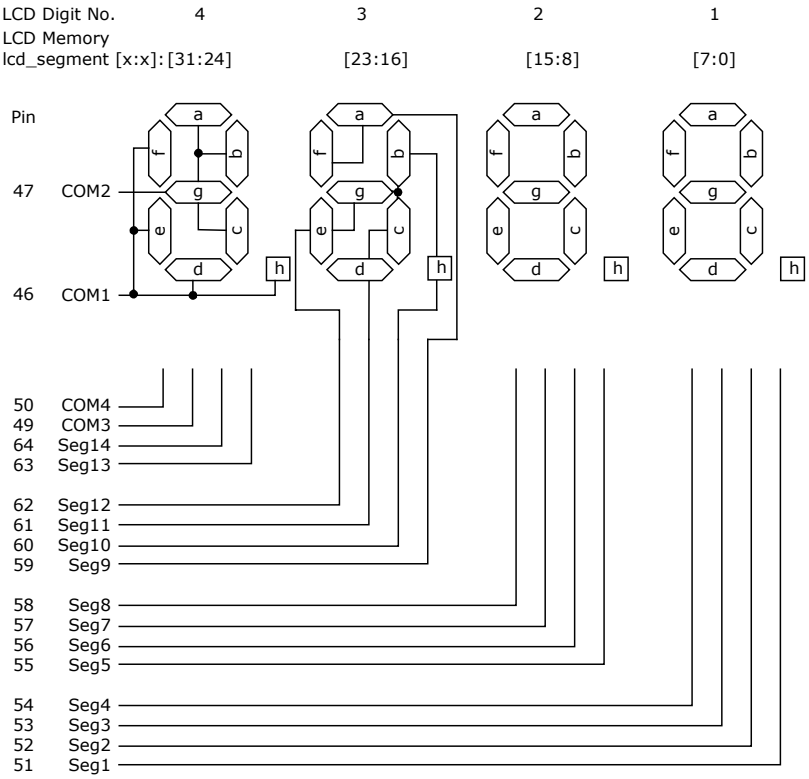
4-MUX



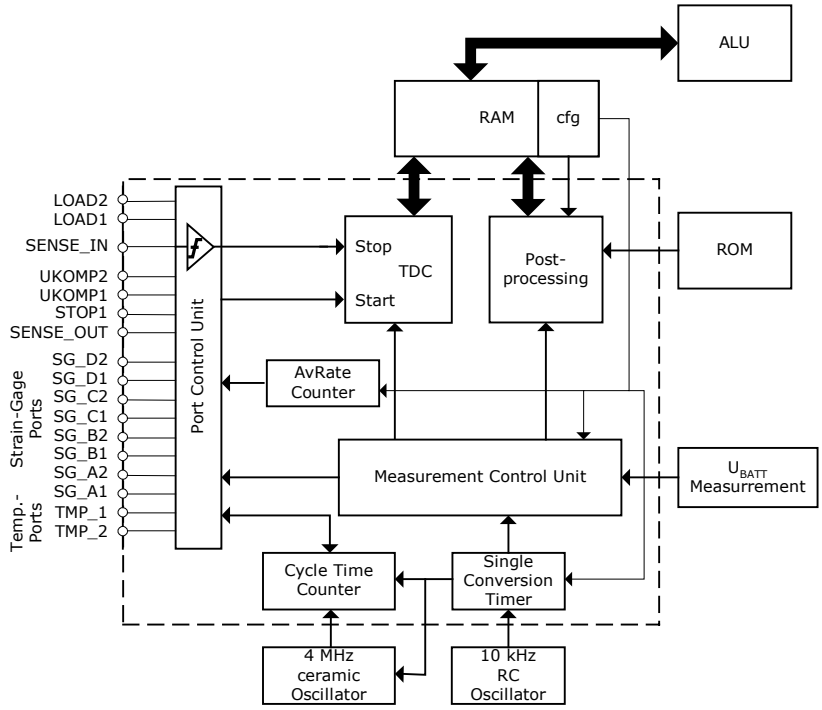
3-MUX



2-MUX


LCD Digit No. 4 3 2 1
LCD Memory
lcd_segment [x:x]: [31:24] [23:16] [15:8] [7:0]
Pin
47 COM2
46 COM1
50 COM4
49 COM3
64 Seg14
63 Seg13
62 Seg12
61 Seg11
60 Seg10
59 Seg9
58 Seg8
57 Seg7
56 Seg6
55 Seg5
54 Seg4
53 Seg3
52 Seg2
51 Seg1
a
b
c
d
e
f
g
h

9 Converter Front-End



9.1 Operating Principle

The PICOSTRAIN based converter has ports to measure 4 independent half bridges, two half bridges that form a full bridge or a classical Wheatstone bridge or a single half bridge

Additionally there are ports for measuring temperature by means of an NTC or a carbon film resistor.

The strain itself is measured by means of discharge time measurements. The discharge time is defined by the strain gauge resistance and the capacitor Cload.

The recommended values for Cload are:

Rsg = 350 Ohm: Cload ~ 270 nF
Rsg = 1000 Ohm: Cload ~ 82 nF

The measuring unit controls the on-time of the 4 MHz oscillator and can measure the operating voltage.

Finally it does the data processing and transfers the final result into the RAM.

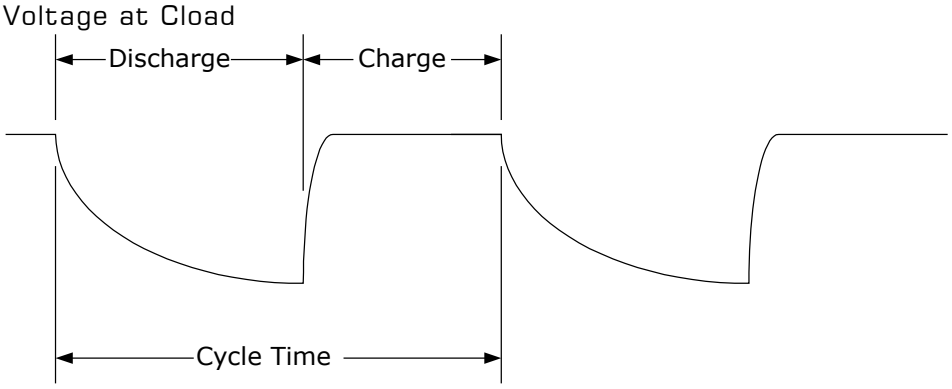
9.2 Parameter Settings

Two major paramters define for a measurement cycle:

- Cycle Time
- AVRate

9.2.1 Cycle Time

The Cycle Time ist the time PS08 needs to discharge and charge C_{load}. Following figure illustrates this relation.



The discharge time is given by the value of the strain gage resistor and the selected capacitor C_{load}. If the recommended values are used (e.g. 270 nF with 350 Ohm SG) the discharge time is in the range of 70 to 75 μs. The charge time has to be large enough to provide a full recharge of C_{load}. The minimum should be 30% of the total time (discharge + charge).

The user sets the cycle time with the 8-bit register CYTIME(7:0). In measuring range 1 the cycle time is generated by the internal 10 kHz clock. In measuring range 2 the cycle time is generated by the high speed clock divided by 8.

Examples:
Measuring Range 1: CYTIME=3 → Cycle Time is approx. 300 μs
Measuring Range 2: CYTIME = 60 → 60 *8 *250 ns = 120 μs @ 4 MHz high speed clock.

The user has to set a suitable cycle time. Too short cycle times should be avoided. Measurements are not possible if the cycle time is shorter or in the range of the discharge time and an overflow will occur. The upper limit of the cycle time is only given by the max. value of the cycle time register. e.g. 512 μs in measuring range 2 with 4 MHz reference.

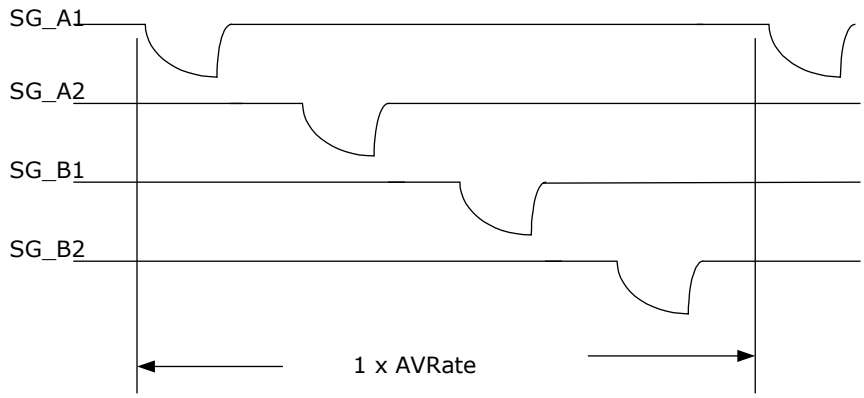
With the recommended C_{load} values the cycle time should not fall below 110μs.

9.2.2 AVRate

The averaging rate AVRate [AVRATE(9:0)] sets the number of complete ratio measurements within one complete measurement cycle. The number of cycle times for one AVRate depends on the number of selected halfbridges. For every halfbridge 2 cycle times are needed.

Halfbridge → 2 Cycle Time per AVRate
Fullbridge → 4 Cycle Time per AVRate
Quattro Bridge → 8 Cycle Time per AVRate

Example: Full bridge with AVRate = 1



The selected AVRate for one measurement defines the possible resolution. The higher is AVRate the higher is the possible resolution. The dependency between AVRate and resolution is given by a square root function. The resolution increases by the square root of the total number of cycle times.

Calculating the resolution:

The base resolution for a half bridge at AVRATE =1 and recommended discharge time (70 to 75 μs) in fast settling mode and 2mV/V excitation is:

With internal comparator: 12.5 Bit eff.
With external bipolar comparator: 12.8 Bit eff.

At higher values of AVRATE the resolution is calculated as:

$$\mathrm{Re}\,solution = \mathrm{Re}\,solution[AVRate = 1] + \frac{\ln\left(\sqrt{AVRate * Brige - factor}\right)}{\ln(2)}$$

The Bridge-factor is: 1 for half bridges
2 for full bridges
4 for quattro bridges

Example 1:

AVRate = 12, Quattro bridge, internal comparator

Resolution = 12.5 + LN(√(12*4))/ln(2) = 12.5 + 2,8 = 15.3 Bit eff. = 40.000 effective divisions = 6.600 peak-peak divisions in fast settle mode.

Example 2:

AVRate = 450, Full bridge, external comparator

Resolution = 12.9 + LN(√(450*2))/ln(2) = 12.8 + 4,9 = 17,7 Bit eff. = 215.000 effective divisions = 36.000 peak-peak divisions in fast settle mode.

9.2.3 Conversion Time/Measuring Rate

The time for one complete measurement can be calculated by means of following formula:

Tconversion = CycleTime*(2*AVRATE * Bridge-factor +3+MFake)

Mfake = #Fake measurements

Mfake-Register	#Fake Measurements
0	0
1	2
2	4
3	16

Example for above calculations:

Example1:

Cycle time = 110µs
AVRate=12
Quattro bridge
Mfake=1

Tconversion = 110µs*(2*12*4+3+2) =11,11 ms → The maximum measuring rate is 90,0 Hz

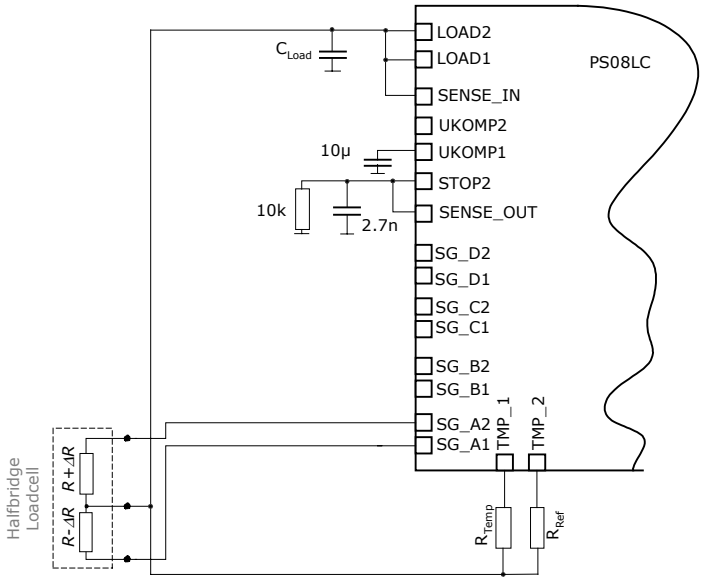
Example2:

Cycle time = 110µs
AVRate=450
Full bridge
Mfake=2

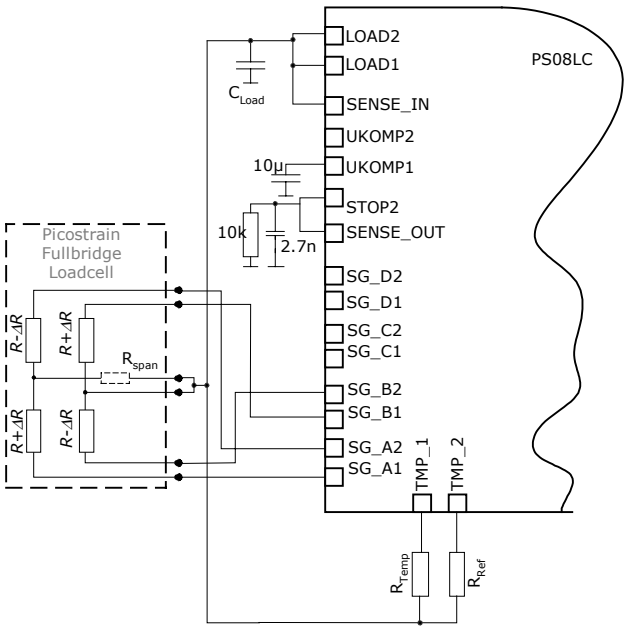
Tconversion = 110µs*(2*450*2+3+4) =198,77 ms → The maximum measuring rate is 5,03 Hz

9.3 Connecting the Strain Gage

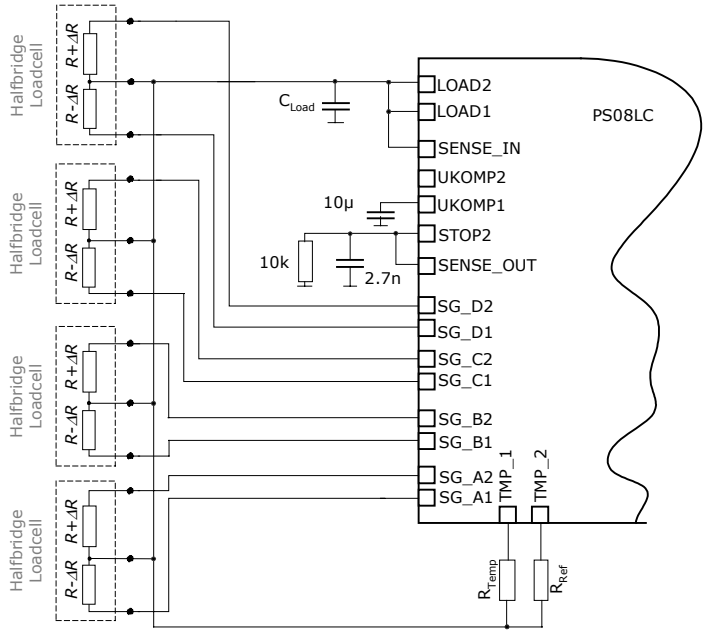
9.3.1 Half Bridge Mode



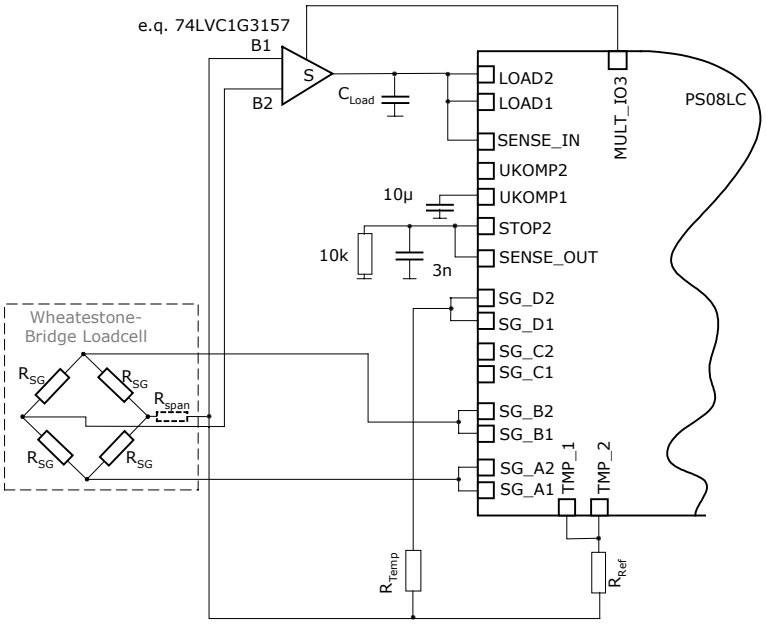
9.3.2 Full Bridge Mode



9.3.3 Quattro Bridge Mode
(Four load cells)



9.3.4 Wheatstone Mode
An external analog switch like 74LVC1G3157 is need when measuring Wheatstone bridges.

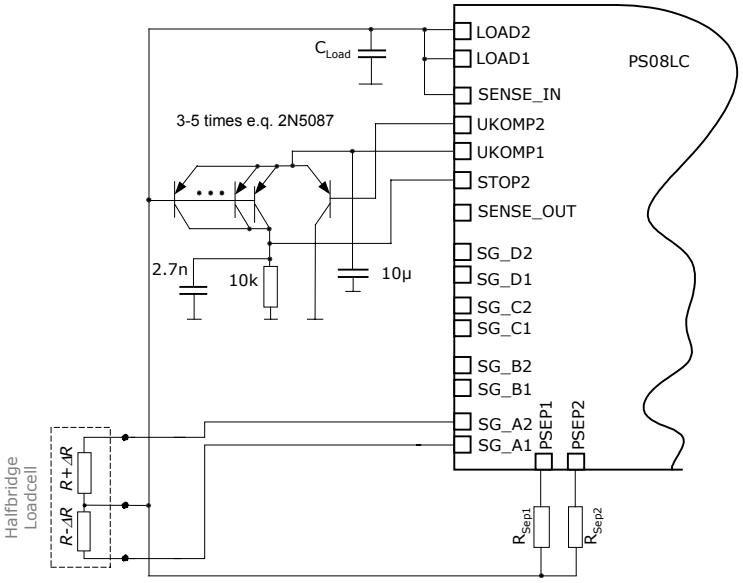


Note:
In Wheatstone mode the system looses 0.6 Bit of resolution. Therefore we recommend the wheatstone mode only for first tests with existing Wheatstone loadcells or for applications with long wires (> 1 m) between loadcell and electronics.

9.4 The Comparator

The end of the discharge cycle is triggered by a comparator. PS08 offers an internal comparator that is selected by setting reg11, sel_compint = 1. By means of the internal comparatot it is possible to reach about 60,000 divisions peak-peak at 2 mV/V, 5 Hz update rate and SINC3 software filter.

The precision of the measurement can be improved by using an external bipolar comparator. The improvement is by 30% to 80,000 divisions under the same conditions.



Recommendations:
- Low-noise PNP transistors like 2N5087 or BC859 should be used
- 3 to 5 transistors in parallel should be used at the LOAD side
- It is not necessary to have matched transistors

When to use an external comparator?

There are three possible reasons:

a) Very high resolution

In case the user looks for the best possible resolution in his application, e.g. in counting scales.

b) Lowest current

In case the user looks for the lowest possible current consumption in his application, e.g. in solar scales. Because of the lower noise, the AVRate can be reduced at a given resolution and therefore the operating current is reduced. With the bipolar comparator the operating current can nearly be halved.

c) Ultra low voltage

In case the customer wants to run his application down to < 2.1 V Vdd, e.g. with 1.55 V

silver oxide batteries. Here the bipolar comparator brings significantly better results (> 0.5 Bit).

9.4.1 Comparator Control

The comparator can be switched on for only the duration of the measurement for current saving reasons (con_comp[1:0]) or continuously. Further, the working resistance of the internal comparator can be changed (sel_compr[1:0]).

We recommend the following settings:

CON_COMP = 'b10 → on during measurement except load
SEL_COMPR = 'b00 → 10k resistor selected

If CON_COMP is set to 'b11 (on) the comparator needs approx. 150µA of constant current.

Capacitors at UCOMP1 and STOP2

The capacitors at UCOMP1 and STOP2 are important for the low noise figure. For best performance we recommend 10µF for Cucomp1 and 2.7 nF for Cstop2.

Nevertheless, other (smaller) values are possible.

Recommendation values:

Cucomp2: not below 1µF
Cstop2: lower than Cucomp2/3000

Example:
Ucomp2 = 1µF → Cstop2 < 1µF/3000 → 330 pF selected.
The noise will slightly increase by about 0.2 – 0.3 Bit.

9.5 Rtemp / Rref

The two resistors Rtemp and Rref are needed for two reasons

- Correction of the delay time of the comparator
- Temperature measurement

The two resistors have to be connected in any case. In case no temperature measurement both resistors can be of the same type (e.g. carbon resistors).

9.5.1 Correction of Comparator Delay

Because the focus of the comparator performance is on ultra low noise, the comparator has a delay time which cannot be neglected. This delay time is temperature dependent and results in a gain error which is too high for precise weight scale applications. Rtemp and Rref are used to measure the delay time periodically during the operation. The PS08 corrects the measuring result with the measured delay time value.

The delay time of the comparator depends on the value of Cucomp2 and Cstop2. Because these values can be changed by the user, there is the possibility to adjust the correction routine by the register Mult_PP[7:0].

A good value for the recommended Cucomp2 and Cstop2 values (10µF and 2.7 nF) ist 0x1A0 (dec160). If the capacitor values are increased also the correct Mult_PP value has to be higher or vice versa. If the selected Mult_PP value is too low the gain will decrease with higher temperature or lower voltage. . If the selected Mult_PP value is too high the gain will increase with higher temperature or lower voltage.

At the right value of Mult_PP the gain of the electronic is absolutely stable over a very wide temperature and voltage range. The temperature drift of the gain is <1ppm/K. The power supply rejection ratio (PSRR) is >140 dB.

9.5.2 Temperature Measurement

Temperature measurement is done by measuring the ratio of the discharge times of two resistors, a temperature dependend one and a temperature stable one. The sensitive resistor may be an NTC or even cheaper a carbon film resistor. The reference resistor can be a metal film resistor.

9.5.3 Values for Rtemp and Rref

The values for Rtemp and Rref has to be adjusted to the Strain Gage resistor and the kind of bridge.

Therefore the resistors should have following values:

Normal : R = Rsg (e.q. 1000 Ohm with 1000 Ohm bridges)
(Half-, Full-, Quattro Bridge):
Wheatstone Bridge: R = 0.75*Rsg (e.q. 750 Ohm with 1000 Ohm Bridges)

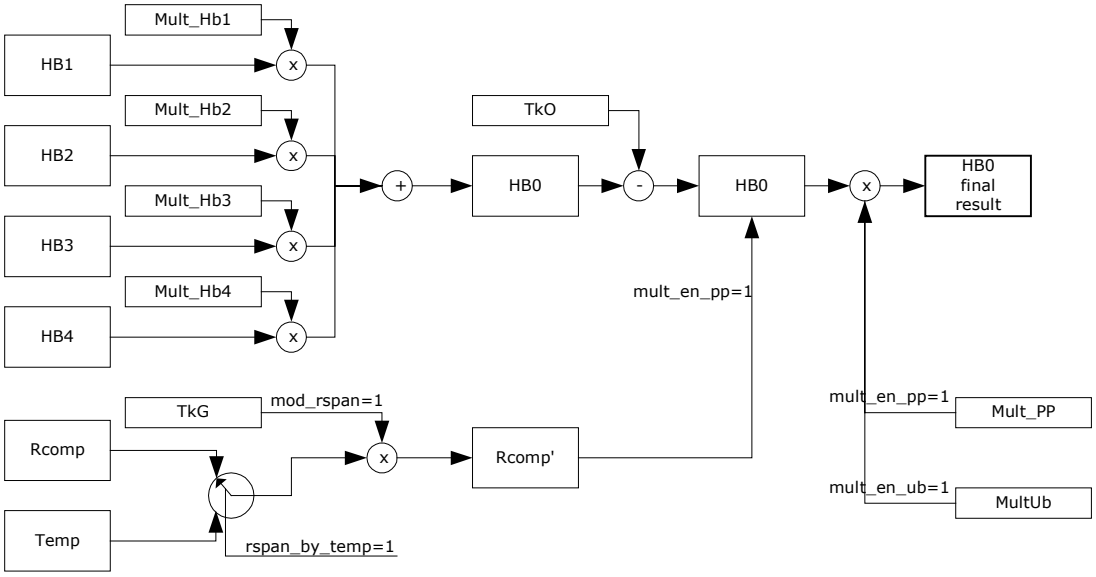
9.6 Post-processing

At the end of a measurement the converter also makes the post-processing of the measurement by means of ROM based routines. It stores the readily calibrated and scaled results in the result registers in the RAM. Afterwards, in case epr_usr_prg =1, the EEPROM program is started.

Post-processing:

Specialties of the post-processing are:

- The results of the four half-bridges may have independent multiplication factors. This offers the possibility to do a software correction for off-center weights.
- Strain sensors and the span compensation resistor are separated. The gain correction can therefore be adjusted by software. Even the temperature measurement can be used instead of the span compenssation resistor. By this method it is possible to make high-quality loadcells out of standard load cells just by software.
- The corrected result may further be multiplied by correction factors depending on the battery voltage. This supports power supply rejection and allows an operation directly from a battery without regulation.



A simle example program to display the results could be:

```
ramadr  20      ;
move    x , r   ; Load x-Accu with the result
move    y , 2   ; Load y-Accu with the comma position
no2lcd          ; Convert into 7-segment display
newlcd          ; Indicate new value to the LCD-driver
.c.lrwdt        ; setback the watchdog
stop            ; Hold the µC
```

9.6.1 Temperature Compensation of Gain and Offset

General Method
Six measurements have to be done with the load cell or scale, at the lower and upper limit of the temperature specification, without load or maximum load, at a given configuration for the PS021. The configuration parameters TKOffs and TKGain can be calculated from these six results, based on a linear approximation for the temperature dependency. It is assumed that non-linearities can be neglected.

Basic Configuration
The method can be used for one or two PICOSTRAIN fullbridges and also for one or two Wheatstone bridges. One, and only one, span compensations resistors has to be connected. The measurements are done at high averaging rate (e.g. 100).

Basic settings are:

		Mult_TkO may also have an initial value different from 0.
avrate	=	This might be reasonable in case of non-adjusted bridges
100		with a very large zero offset. Measuring without a rough
mod_rspan	= 1	pre-setting of the Mult_TkO would give quite a high offset
rspan_by_temp	= 0	drift, leading to additional errors. In this case it is helpful
Mult_TkG	= 0,5	to start with a Mult_TkO value close to the real zero
Mult_TkO	= 0	offset. In this case the initial Mult_TkO value has to be
		added to the calculated one.

M1:	= M(T1,L0,TkG0) Result with Mult_TkG = 0 (TkG0) at lower temperature (T1) and no load (L0)	No Load	Cold
M2:	= M(T1,L1,TkG0) Result with Mult_TkG = 0 (TkG0) at lower temperature (T1) and maximum load (L0)	Max. Load	
M3:	= M(T1,L1,TkG1) Result with Mult_TkG = 1 (TkG1) at lower temperature (T1) and maximum load (L1)		
M4:	= M(T2,L0,TkG0) Result with Mult_TkG = 0 at upper temperature (T2) and no load (L0)	No Load	Warm
M5:	= M(T2,L1,TkG0) Result with Mult_TkG = 0 at upper temperature (T2) and maximum load (L1)	Max. Load	
M6:	= M(T2,L1,TkG1) Result with Mult_TkG = 1 (TkG1) at upper temperature (T2) and maximum load (L1)		

The correction factors are calculated according to formulas:

$$TKOffs = \frac{M4 \times M2 - M1 \times M5}{(M4 + M2) - (M1 + M5)}$$

$$TKGain = \frac{(M5 - M4) - (M2 - M1)}{(M2 - M1)\left(\frac{M5}{M6} - 1\right) - (M5 - M4)\left(\frac{M2}{M3} - 1\right)}$$

These values have to be written into the appropriate TKOffs and TKGain registers of the PS08.

Note
This method works only in case that Rcomp has a significant influence on the compensation. Only then the denominator is different from zero.

The method doesn't work in the extreme case of self-compensated strain gages. Then MW(T2,L0,TKG0), MW(T1,L1,TKG0), MW(T1,L0,TKG0), MW(T2,L1,TKG0) are nearly trimmed and Rcomp is used only for fine adjustment.

9.6.2 Off-center Correction for Quattro Scales

Several scales like body scales have four loadcells, each with a half-bridge sensor on it. The indicated weight might vary in case the loadcells do not all have exactly the same sensitivity and the weight is not put on the the center point of the scale. PS08 allows to correct the gain of the half-bridges just by software without trimming or adding an additional trim circuit. Each half-bridge result is assigned its own multiplication factor (Mult_HB1 to Mult_HB4). Simply by four measurements it is possible to calculate the multiplication factors for the correction. Therefore a nominal load has to be put on each corner of the scale. At each corner the results of HB0 to HB4 are taken.
Please contact acam for the algorithm to calculate the factors Mult_HB1 to Mult_HB4.

10 Oscillators

PS08 has an internal low-current 10kHz oscillator which is used for basic timer functions and for the definition of the cycle time in measuring range 1.
Further, PS08 has an oscillator driver for an external 4 MHz ceramic resonator. This one is used for the time measurement and for the definition of the cycle time in measuring range 2. This oscillator can be switched on continously or only for the duration of the measurement, including some lead time to reach the full oscillation amplitude (sel_start_osz[1:0]).

11 Voltage Measurement

An internal bandgap reference is used for measuring the voltage. This is done 40 times per second. The result is stored in the RAM at address 25, UBATT. It is calculated as Voltage = 2.0 V + 1.6 V* UBATT/64.

The result can be used for
- Low-battery detection: the level is set in configuration register low_batt[2:0]

low_batt	0	1	2	3	4	5	6	7
Level (V)	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9

Flag flg_ub_low in the status register indicates if the voltage is below the set level .

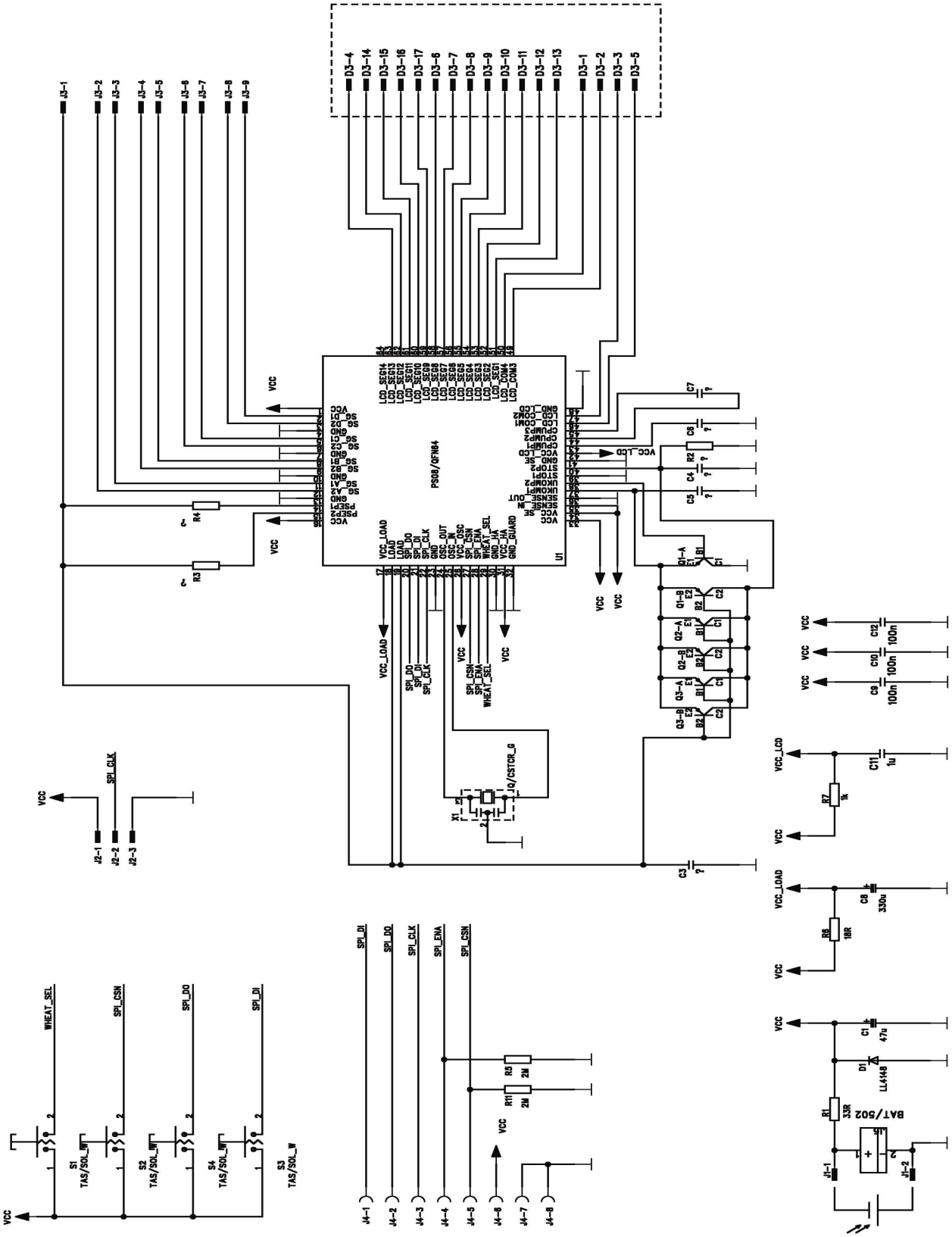
- Power supply rejection: The measured voltage can be used to correct the dependency of the gain from the voltage. It is switched on by setting configuration bits mult_en_ub = 1 and mult_ub[7:0]. The result of the strain measurement will be corrected according to HB = HB/(1 + UB*[-128 ... 127]/2^21).

- EEPROM protection: when the voltage is below 2.4 V the automatic EEPROM write (putepr) is prohibited. This protcets the EEPROM against corrupt data.

12 Auto-on

PS08 can be used to run scales in a true auto-on mode. During the stand-by phase the PS08 measures with avrate = 1 (minimum resolution) and low update rate (e.g. 1 Hz). In such a configuration the whole system current can be reduced to 2 μA. In case a significant change in weight is detected, the averaging rate and update rate can be increased by reconfiguring avrate and tdc_conv_cnt by means of the software. As a big advantage the scale can display immediatedly a correct result without delay.

13 Sample Circuit



14 Beta vs. Final Version

The planned final version of PS08 will have some more features respectively improvements towards the beta version of the chip. The following list shows a short overview about the planned measures:

- Fixing of the undersampling issue, see 5.1 for more details
- Decrease of the LCD current consumption
- Extension of internal ROM routines, which makes it easier to use rolling average
- Increase of 16 words in RAM up to 48 words in total
- Read protection for EEPROM
- Improved shift error correction for Quattrobridge mode
- Improved temperature compensation (ModRSpan)

Last Changes

12.04.2007 First Edition
17.07.2007

Contacts

Headquarter Germany	acam-messelectronic gmbh	Am Hasenbiel 27 76297 Stutensee-Blankenloch	Tel: +49 (0) 7247 7419-0 Fax: +49 (0) 7247 7419-29 support@acam.de www.acam.de

European Distributors

Belgium (Vlaanderen)	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.censelect.nl
France	microel (CATS S.A.)	Immeuble "Oslo" - Les Fjords 19, avenue de Norvège Z.A. de Courtaboeuf - BP 3 91941 LES ULIS Cedex	Tel. : +33 1 69 07 08 24 Fax : +33 1 69 07 17 23 commercial@microel.fr www.microel.fr
Great Britain	2001 Electronic Components Ltd.	Stevenage Business Park, Pin Green Stevenage, Herts SG1 4S2	Tel. +44 1438 74 2001 Fax +44 1438 74 2001 a.parker@2k1.co.uk www.2k1.co.uk
Italy	DELTA Elettronice s.r.l	Via Valpraiso 7/A 20144 Milano	Tel: +39 02 485 611 1 Fax: +39 02 485 611 242 email: afrigerio@deltacomp.it www.deltacomp.it
Netherlands	CenS (Micro) Electronics BV.	PO Box 2331/ NL 7332 EA Apeldoorn Lamfe Amerikaweg 67 NL 7332 BP Apeldoorn	Tel: +31 (0) 55 3558611 Fax: +31 (0) 55 3560211 info@censelect.nl www.censelect.nl
Switzerland	Computer Controls AG	Neunbrunnenstr. 55 8050 Zürich	Tel.: +41-1-308 6666 Fax: +41-1-308 6655 email: roeschger@ccontrols.ch www.ccontrols.ch
Russia	Galant Electronics, Ltd.	100, Prospekt Mira, Moscow, 129626, Russia	Tel\Fax: +7-495-987-42-10, Tel: +7-095-107-19-62 Mobile +7-916-993-67-57 Email: leonid-k@galant-e.ru www.galant-e.ru

American Distributors

United States of America	Transducers Direct, LCC	264 Center Street Miamiville, Ohio 45147	Tel: 513-583-9491 Fax: 513-583-9476 email: sales@acam-usa.com www.acam-usa.com

Asian Distributors

India	Brilliant Electro-Sys. Pvt. Ltd.	4, Chiplunker Building, 4 Tara Temple Lane, Lamington Road, Bombay – 400 007	Tel: +91 22 2387 5565 Fax: +91 22 2388 7063 www.brilliantelectronics.com besimpex@vsnl.net
Israel	ArazimLtd.	4 Hamelacha St. Lod P.O.Box 4011 Lod 71110	Tel: 972-8-9230555 Fax: 972-8-9230044 email: info@arazim.com www.arazim.co.il
Japan	DMD–Daiei Musen Denki Co., Ltd.	10-10, Sotokanda, 3-Chome, Chiyoda-Ku Tokyo 101-0021	Tel: +81 (0)3 3255 0931 Fax: +81 (0)3 3255 9869 sales@daiei-dmd.co.jp www.daiei-dmd.co.jp
P.R. China	Broadtechs Technology Co. Ltd.	Shanghai Office: 3C JinHuan Building, 489 Xiang Yang Road South Shanghai, 200031	Tel.: +86-21-54654391 Fax: +86-21-64454370 Email: info@acam-china.com www.acam-china.com
South Korea	SamHwa Technology Co., Ltd.	#4 4F Kyungwon building, 416-6 Jakjeon-dong GYEYANG-GU, INCHEON 407-060	Tel: +82 32 556 5410 Fax: +82 32 556 5411 www.isamhwa.com minjoonho@isamhwa.com