acam-messelectronic gmbH

is now

Member of the ams Group

The technical content of this acam-messelectronic document is still valid.

Contact information:

Headquarters:
ams AG
Tobelbaderstrasse 30
8141 Unterpremstaetten, Austria
Tel: +43 (0) 3136 500 0
e-Mail: ams_sales@ams.com

Please visit our website at **www.ams.com**

acam
mess • electronic

PICOSTRAIN

Data Sheet

PSØ81

Single Chip Solution for Strain Gauges

July 4th, 2012
Document-No.: DB_PSØ81_en V0.9

Published by acam-messelectronic gmbh

Disclaimer / Notes

The information provided by this data sheet is believed to be accurate and reliable. However, no responsibility is assumed by acam for its use, nor for any infringements of patents or other rights of third parties that may result from its use. The information is subject to change without notice and is provided „as is“ without warranty of any kind (expressed or implied). Picostrain is a registered trademark of acam. All other brand and product names in this document are trademarks or service marks of their respective owners.

Support

For a complete listing of Direct Sales, Distributor and Sales Representative contacts, visit the acam web site at: http://www.acam.de/company/distributors or refer to chapter 7.2 in this datasheet

For technical support you can contact the acam support team in the headquarter in Germany or the Distributor in your country. The contact details of acam in Germany are:
support@acam.de or by phone +49-7244-74190.

Table of Contents

Table of Contents

Page

1 Overview

1.1 Features

- RMS noise:
  - 20.1nV fast settle, 5 Hz
  - 11.5 nV SINC3, 5 Hz
  - 8.9 nV SINC5, 5 Hz
- Up to 250,000 peak-peak divisions in weighing applications (2 mV/V strain)
- Scalable udapte rate from < 1 Hz to 1000 Hz
- Current consumption:
  - ~ 0.39 mA PSØ81 itself (at maximum speed)
  - ~ 0.005 mA PSØ81 itself (at low current configuration)
  - ~ 0.001 mA standby current
- Power supply voltage: 2.1 V to 3.6 V
- Converter type: Time-to-digital converter (TDC)
- Resolution: 28 bit ENOB (RMS) or 25.8 bit noise-free (peak-to-peak)
- 24-Bit internal microprocessor with 2 KB reprogrammable EEPROM
- Internal LCD controller for 4x14, 3x15 and 2x16 segments
- 4-wire serial SPI interface
- Internal very low current 10 kHz oscillator
- 6 I/O pins, configurable up to 21 inputs or 5 outputs
- Very high power supply rejection ratio (PSRR)
- Very low gain and offset drift
- Embedded charge pump for driving the LCD
- Embedded bandgap voltage reference for low battery detection
- Watchdog timer

1.2 Advantages

- Single-chip solution for weighing applications
- Converter, microcontroller and LCD controller in one chip
- Extreme low total system current (down to 15µA including strain gages)
- Very low self heating of teh sensor
- Gain and offset correction of the load cell
- Available as dice (115 µm pitch) or packaged (QFN56, 7x7 mm²)

1.3 Applications

Industrial

- Torque wrenches
- Pressure indicators
- Legal for trade scales
- Counting scales

Consumer

- Pure solar driven scales
- Body scales
- Kitchen scales
- Pocket scales
- Hanging scales
- Postal scales
- Package scales

1.4 General Description

The PSØ81 is a system-on-chip for ultra low-power and high resolution applications. It was designed especially for weight scales but fits also to any kind of force or torque measurements based on metal strain gages. It takes full advantage of the digital measuring principle of PICOSTRAIN. Thus, it combines the performance of a 28-Bit signal converter with a 24-Bit microprocessor. Additional elements like an LCD driver, 3K ROM with many complex pre-defined functions,

2K EEPROM program memory and an integrated 10 kHz oscillator round off the device. A small amount of external components is sufficient to build a complete weighing electronic.
The part operates with a power supply from 2.1V to 3.6V and has a very low current consumption. As per configuration the current consumption ranges between 0.005 mA and 0.4 mA approximately. The update rate is scalable in a wide range from < 1 Hz up to 1000 Hz. With a maximum of mor then 1 million internal divisons (28 bit ENOB RMS) the resolution lies in the top range of today's converters. This high resolution is only comparable to the one of high-end AD converters, but at a much lower current consumption. Equipped with these features, a variety of scale electronics can be served with PSØ81. On the resolution side, it allows to build scales with up to 250,000 stable peak-peak divisons (at 2mV/V)!
On the other hand, a sophisticiated power management and the special features of the PICOSTRAIN measuring principle can reduce the total current of the system down to 15 µA, including the sensor current. This way, it is the first time possible to build pure solar driven weigh scales based on metal strain gages. Of course, the benefits can be combined, e.g. building a high resolution scale with a low current such as a legal for trade scale that runs more than 1,500 operating hours with 2x AA batteries.
Throwing a glance at further features like software adjustment of the offset and gain compensation or the possibility to operate only one half bridge reveals that the PSØ81 opens the door to new and innovative product solutions.

1.5 Functional Block Diagram

Figure 1.1 Block Diagram

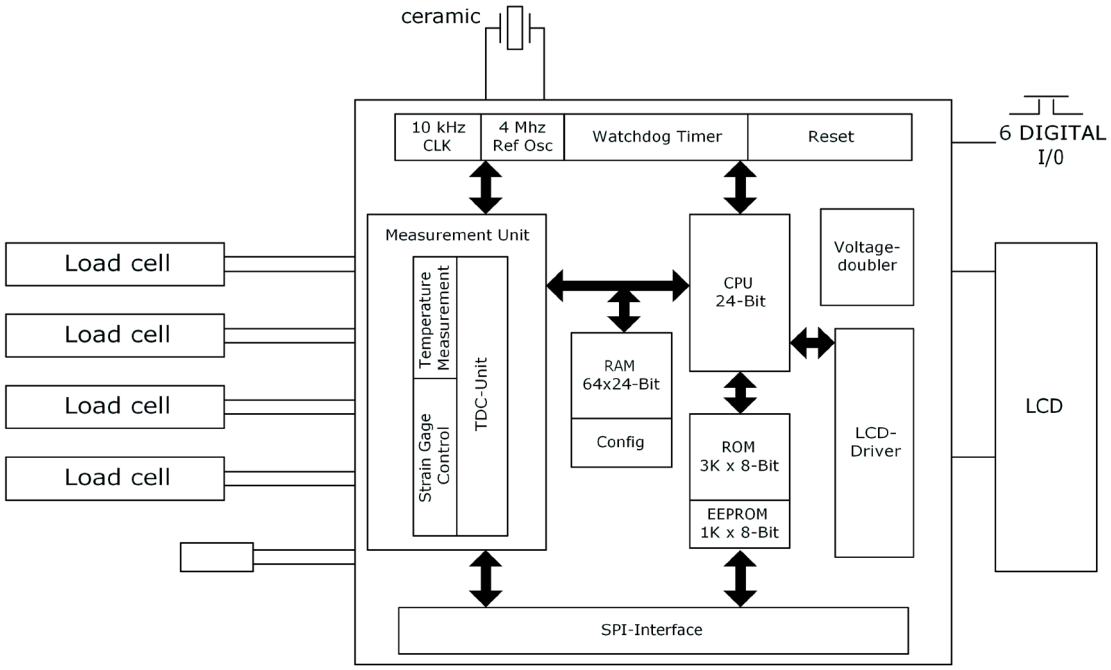


Table of Contents

Page

2 Characteristics and Specifications

2.1 Absolute Maximum Ratings

Table 2.1 Maximum Ratings

Symbol	Parameter	Conditions	Min	Max	Unit
Vcc Vcc_load Vcc_osc Vcc_LCD	Supply voltage	Vcc vs. GND	-0.5	5.0	V
Vin	DC input voltage		-0.5	Vcc + 0.5	V
ESD Rating	FICDM	All pins	2		kV
Tstg	Storage Temperature	Plastic package	-55	150	°C

2.2 Normal Operating Conditions

Table 2.2 Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
Vcc Vcc_load Vcc_osc Vcc_LCD	Supply voltage	Vcc vs. GND (* without EEPROM programming, voltage measurement and LCD)	2.1 (1.5*)	3.6**	V
Vin	DC input voltage		0.0	Vcc	V
Vout	Output voltage		0.0	Vdd	V
Top	Operating temperature		-40	125	°C
Tstg	Storage temperature	Plastic package	-55	150	°C

** 4.5 V for highest resolution within limited temperature range

2.3 Electrical Characterization

Table 2.3 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 hysteresis	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
Iq	Quiescent current	No oscillator, no TDC		150		nA
Iosc	Current 4 MHz oscillator continuously on	Vcc = 3.6 V Vcc = 3.0 V Vcc = 2.1 V		200 130 65		µA

2.4 Converter Precision

Table 2.4 Performance at Vcc = 3.3V with external comparator

	ENOB *dR/R strain resistance*		
Frequency (Hz)	**No filter**	**SINC3**	**SINC5**
500	23.8	24.8	25.2
250	24.4	25.2	25.7
100	25.2	25.8	26.1
50	25.5	26.2	26.5
20	26.0	26.8	27.0
10	26.6	27.4	27.7
5	27.2	27.9	28.3

Table 2.5 Performance at Vcc = 3.3V with external comparator, related to 2 mV/V strain (weigh scale)

	Resolution @ ***2 mV/V*** max. out, ***Fast settle****			
Frequency (Hz)	**ENOB**	**Divisions**	**Noise nV**	**Noise nV**
		effective	rms	peak-peak
500	14.8	28,000	231	1,386
250	15.4	44,000	148	891
100	16.2	74,000	89	535
50	16.5	95,000	69	416
20	17.0	133,000	49	297
10	17.6	200,000	33	198
5	18.2	294,000	22	135

** Fast settle = without filter*

Table 2.6 Performance at Vcc = 3.3V with external comparator, related to 2mV/V strain (weigh scale)
With SINC3 and SINC5 filter (rolling average of 3 respectively 5)

	Resolution @ ***2 mV/V*** max. out, ***SINC3 Filter***				Resolution @ ***2 mV/V*** max. out, ***SINC5 Filter***			
Frequency (Hz)	**ENOB**	**Divisions**	**Noise nV**	**Noise nV**	**ENOB**	**Divisions**	**Noise nV**	**Noise nV**
		effective	rms	peak-peak		effective	rms	peak-peak
500	15.8	55,000	118	713	16.2	74,000	89	535
250	16.2	74,000	89	535	16.7	105,000	62	376
100	16.8	114,000	57	347	17.1	142,000	46	277
50	17.2	153,000	42	257	17.5	181,000	36	218
20	17.8	222,000	29	178	18.0	266,000	24	149
10	18.4	344,000	19	115	18.7	416,000	15	95
5	18.9	476,000	13	83	19.3	625,000	10	63

Table 2.7 Performance at Vcc = 3.3V with internal comparator

	ENOB *dR/R strain resistance*			ENOB *2mV/V, Fast settle**
Frequency (Hz)	**No filter**	**SINC3**	**SINC5**	
500	23.0	24.0	24.4	14.0
250	23.6	24.4	25.1	14.6
100	24.4	25.0	25.3	15.4
50	24.7	25.4	25.7	15.7
20	25.2	26.0	26.2	16.2
10	25.8	26.6	26.9	16.8
5	26.4	27.1	27.5	17.4

** Fast settle = without filter*

Table 2.8 General parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
INL	Integral Non-linearity	Supply Voltage 3.0V to 3.6V		0.01*		µV/V
	Offset drift	Total system, 350 Ω SG				
		Full-bridge		± 10		nV/V/K
		Wheatstone		< 1		nV/V/K
	Gain drift over -20°C … +70°C	Total System. 350 Ω SG, 5V		~ 1		ppm/K
PSSR	Power Supply Rejection Ratio Vcc	1.8V or 3.3 V +-0.3 V	106 @1.8V	130 @3.3V		dB

** equals to ± 1.25 ppm of A/D-Converters with PGA setting 128*

*** using full bridge wiring for minimum zero drift*

2.5 Integral Nonlinearity

The integral nonlinearity (INL) of PS081 can be specified to ± 0.01 µV (1.25 ppm). Expressed in divisions this corresponds to ± 1: 200,000. This is a tremendous high linearity compared to the one of nowadays latest A/D converters. Ordinary A/D converts have a linearity in the range of ± 0.12µV/V (15 ppm), better A/D converters reach a linearity of ± 0.4 µV (5ppm).

Of course, you can only determine the linearity of the electronics itself, if you can provide a sensor which is accurate and linear enough to measure it. We used for this purpose the revised version of the acam load cell simulator (ALCS350-V2) which offers comfortable methods to investigate not only the linearity, but also variations over temperature or voltage. The linearity of ALCS350-V2 is ± 0.01 µV/V (1:200,000) with PICOSTRAIN wiring and ± 0.04 µV/V with Wheatstone wiring and therefore much more linear than an ordinary load cell. Linearity investigations with the PS081 and the ALCS350-V2 as a sensor are shown in the following tables. More details about the possibilities and the limitations of the load cell simulator are provided in the ALCS350-V2 datasheet.

Figure 2.1 Non-linearity over supply voltage

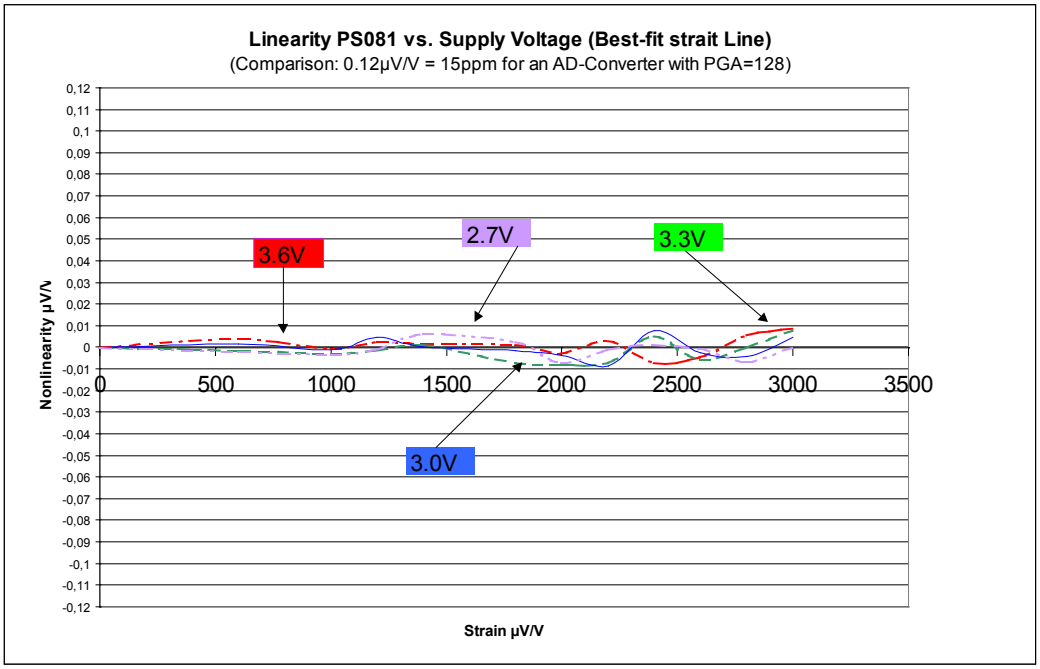
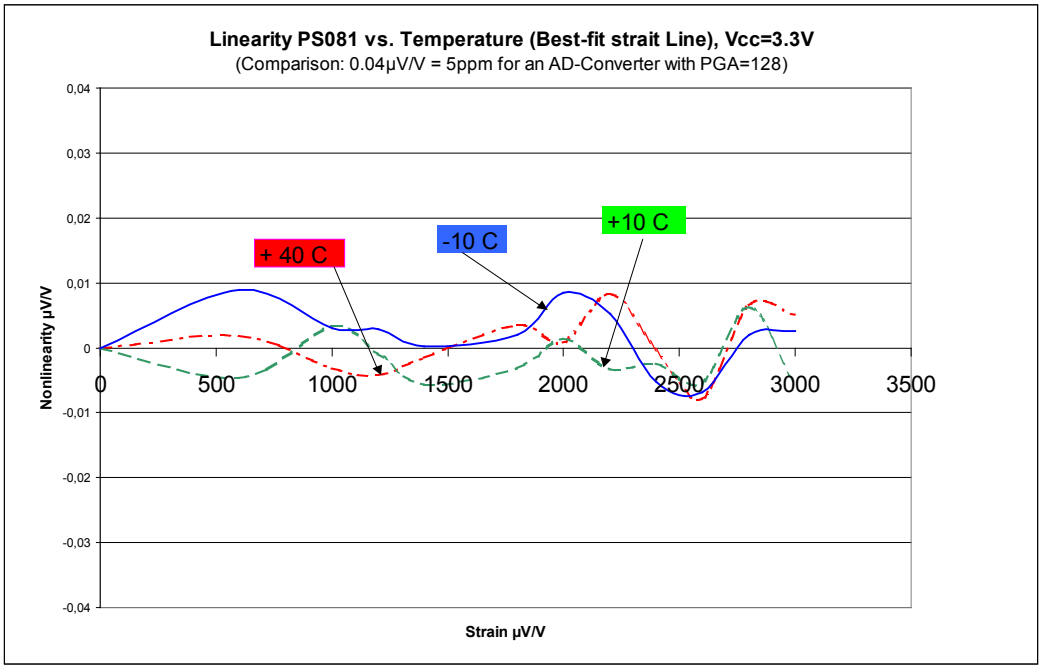


Figure 2.2 Non-linearity over temperature



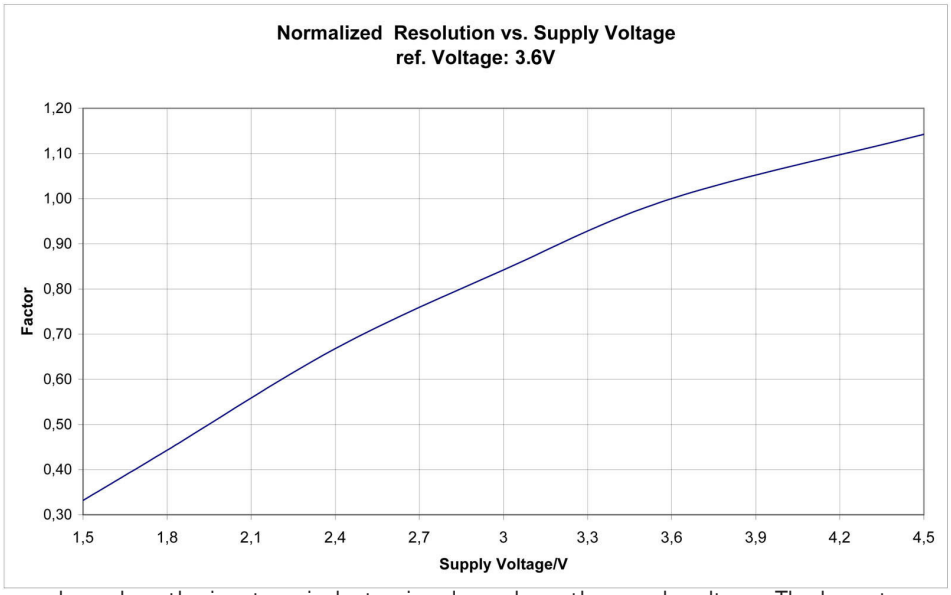
Note:

Best linearity is achieved with a supply voltage in the range of 3.0 V to 3.6 V. The linearity also depends on the setting for the cycle time. The measurements shown above are done with a cycle time setting of cytime = 85 (equals 170 µs).

2.6 Resolution vs. Supply Voltage

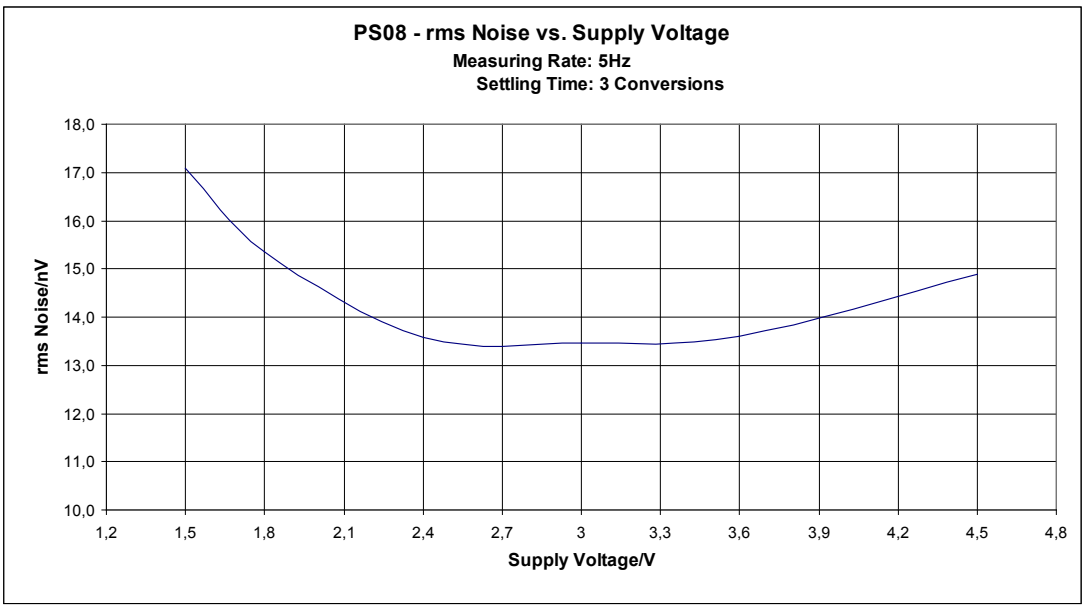
PSØ81 can be driven over a very large supply voltage range. The resolution depends on the supply voltage. The higher the supply voltage the higher the achievable resolution. The diagram below shows the resolution vs. supply voltage which can be achieved with PSØ81. The values refer to 3.6 V.

Figure 2.3 Resolution vs. supply voltage



Following diagram shows how the input equivalent noise depends on the supply voltage. The lowest input noise is archived between 2.4 V and 3.6 V. The maximum differential input voltage (e.g. 6.6 mV @ 2mV/V and 3.3 V supply voltage) divided by the input noise gives the effective resolution.

Figure 2.4 RMS-noise vs. supply voltage



2.7 Current Consumption

The following table shows the total system current of the scale (including current through sensor)

Table 2.9 Current consumption at different resolutions

Divisions *	Update Rate	Double Tara*	Operating Current @ 3V		Scale type	Operating hours
2,000	3 Hz	1 mV/V	1 kOhm	15 µA	Solar	
2,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	30 µA 70 µA	Postal, Body, Kitchen , Pocket	3,000 hours (1xCR2032)
5,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	80 µA 180 µA	High-end postal, Kitchen, Pocket	1,500 hours (1xCR2032)
10,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	300 µA 700 µA	High-end pocket, Coun- ting	2,000 hours (1xCR2430)
80,000	5 Hz	2 mV/V	1 kOhm 350 Ohm	1.9 mA 4.5 mA	Counting	1,500hours 2 x AA

** Divisions are peak-peak values with 5 Sigma (e.g. 80.000 divisions are 400.000 bits of effective resolution)*

2.8 Timings

All timings specified at 3.3V ±0.3V, Ta –40°C to +85°C unless otherwise specified.

Table 2.10 Oscillator timing

Symbol	Parameter	Min	Typ	Max	Units
Clk10kHz	10 kHz reference oscillator		10		kHz
ClkHS	High-speed reference oscillator		4		MHz
toHSst	Oscillator start-up time with ceramic resonator		50	150	µs

Table 2.11 Serial Interface Timing (SPI)

Symbol	Parameter	Min	Typ	Max	Units
f_{clk}	Serial clock frequency			1	MHz
t_{pwh}	Serial clock, pulse width high	500			ns
t_{pwl}	Serial clock, pulse width low	500			ns
t_{sussn}	SSN enable to valid latch clock	500			ns
t_{pwssn}	SSN pulse width between write cycles	500			ns
t_{hssn}	SSN hold time after SCLK falling				
t_{sud}	Data set-up time prior to SCLK falling	30			ns
t_{hd}	Data hold time before SCLK falling	30			ns
t_{vd}	Data valid after SCLK rising				ns

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

Figure 2.5 SPI - Write access

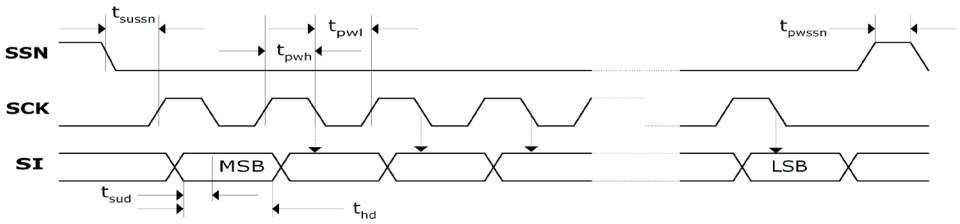
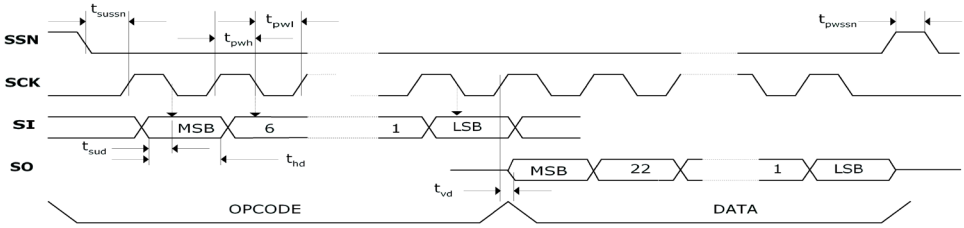


Figure 2.6 SPI-Read access



2.9 Pin Assignment

PSØ81 is avaliable as Die or in QFN56 package. The following pictures and tables show the pin assignment and the pin description.

QFN56

Figure 2.7 Pin assignment QFN56

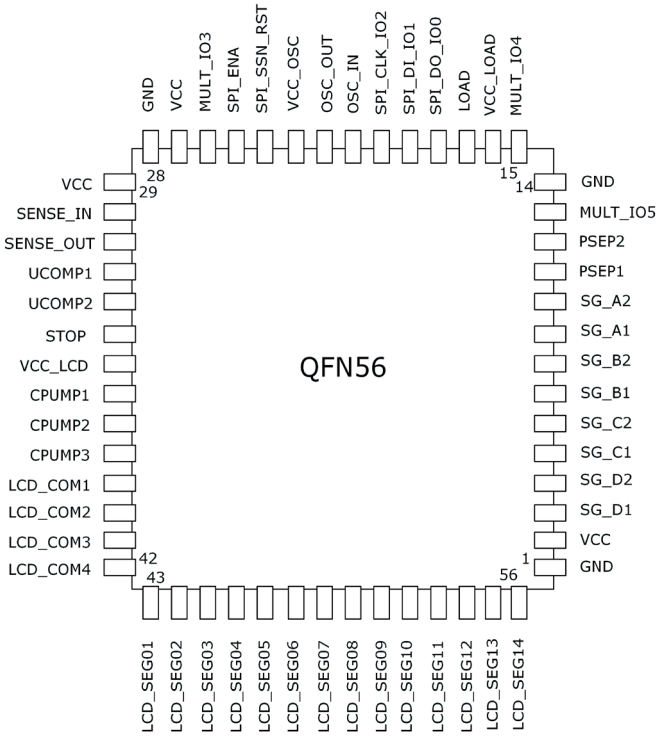


Table 2.12 Pin Description QFN56

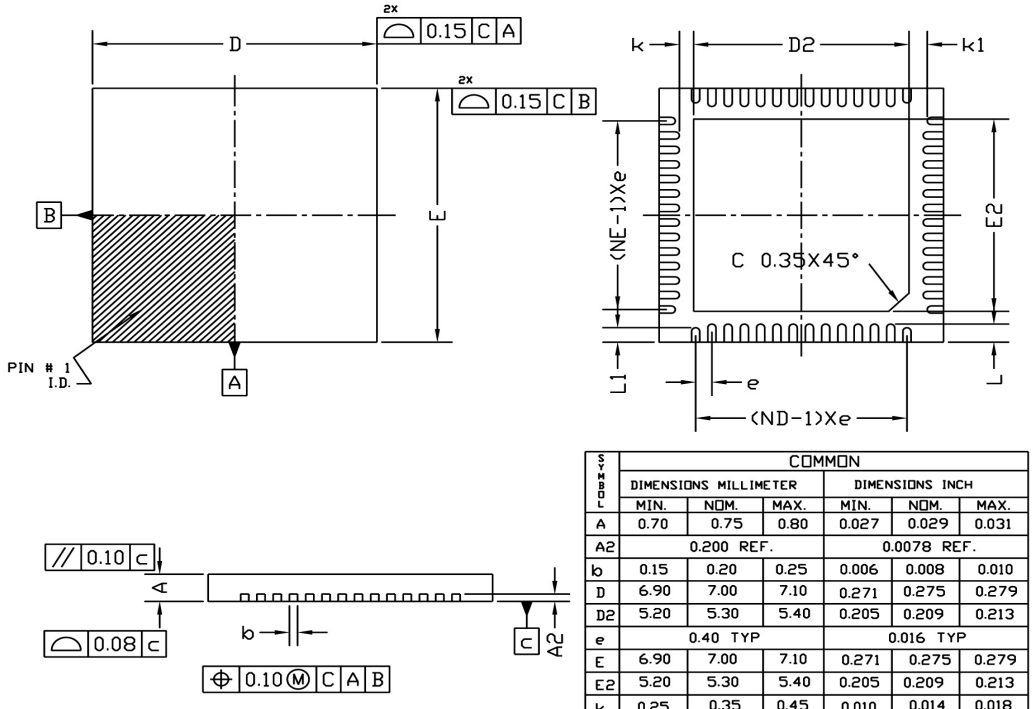
#QFN	Name	Description	Type
1	GND	Ground	
2	Vcc	Supply voltage digital part , I/O, 4MHz-osc.	
3	SG_D1	Port 1 halfbridge D	N Open Drain
4	SG_D2	Port 2 halfbridge D	N Open Drain
5	SG_C1	Port 1 halfbridge C	N Open Drain
6	SG_C2	Port 2 halfbridge C	N Open Drain
7	SG_B1	Port 1 halfbridge B	N Open Drain
8	SG_B2	Port 2 halfbridge B	N Open Drain
9	SG_A1	Port 1 halfbridge A	N Open Drain
10	SG_A2	Port 2 halfbridge A	N Open Drain
11	PSEP1	Port 1 temperature measurement	N Open Drain
12	PSEP2	Port 2 temperature measurement	N Open Drain
13	MULT_IO5	Multi purpose I/O no. 5	Mult-IO
14	GND	Ground	
15	MULT_IO4	Multi purpose I/O no. 4	Mult-IO
16	Vcc_load	Power supply load output pin	
17	Load	Load output to measuring capacitor	P Open Drain
18	SPI_DO_IO0	Output serial SPI interface or IO0	Mult-IO
19	SPI_DI_IO1	Input serial SPI interface or IO1	Mult-IO
20	SPI_CLK_IO2	Clock serial SPI interface or IO2	Mult-IO
21	OSC_IN	Input to 4MHz ceramic resonator	
22	OSC_OUT	Output to 4MHz ceramic resonator	
23	VCC_OSC	4MHz Oscillator supply voltage	
24	SPI_CSN_RST SPI_SSN_RST	Slave select or RST input (High active)	Input with pull-down
25	SPI_ENA	Serial SPI interface enable	
26	MULT_IO3	Select for Wheatstone comparator MUX or Interrupt or Multi purpose I/O no. 3	Wheatstone select Mult-IO
27	Vcc	Supply voltage digital part , I/O, 4MHz-osc.	
28	GND	GND	
29	Vcc	Supply voltage digital part , I/O, 4MHz-osc.	
30	SENSE_IN	Input internal CMOS comparator, connect to Vcc if not used	Analog In
31	SENSE_OUT	Output internal CMOS comparator	Analog Out
32	UCOMP1	External comparator circuit connection	Analog Out
33	UCOMP2	External comparator circuit connection	Analog Out
34	STOP	Stop input measuring signal	
35	VCC_LCD	Supply voltage LCD, 10kHz osc., bandgap	
36	CPUMP1	LCD voltage doubling and stabilization	Analog Out
37	CPUMP2	LCD voltage doubling and stabilization	Analog Out
38	CPUMP3	LCD voltage doubling and stabilization	Analog Out
39	LCD_COM1	LCD line driver for 1/2, 1/3, 1/4 duty	LCD Buffer

40	LCD_COM2	LCD line driver for 1/2, 1/3, 1/4 duty	LCD Buffer
41	LCD_COM3	LCD line driver for 1/3, 1/4 duty, row driver for 1/2 duty	LCD Buffer
42	LCD_COM4	LCD line driver for 1/4 duty, row driver for 1/2, 1 /3 duty	LCD Buffer
43	LCD_SEG1	LCD row driver	LCD Buffer
44	LCD_SEG2	LCD row driver	LCD Buffer
45	LCD_SEG3	LCD row driver	LCD Buffer
46	LCD_SEG4	LCD row driver	LCD Buffer
47	LCD_SEG5	LCD row driver	LCD Buffer
48	LCD_SEG6	LCD row driver	LCD Buffer
49	LCD_SEG7	LCD row driver	LCD Buffer
50	LCD_SEG8	LCD row driver	LCD Buffer
51	LCD_SEG9	LCD row driver	LCD Buffer
52	LCD_SEG10	LCD row driver	LCD Buffer
53	LCD_SEG11	LCD row driver	LCD Buffer
54	LCD_SEG12	LCD row driver	LCD Buffer
55	LCD_SEG13	LCD row driver	LCD Buffer
56	LCD_SEG14	LCD row driver	LCD Buffer

QFN56 Package Outline

QFN56, 7x7 mm², 0.4mm Pitch

Figure 2.8 Package outline (QFN56)

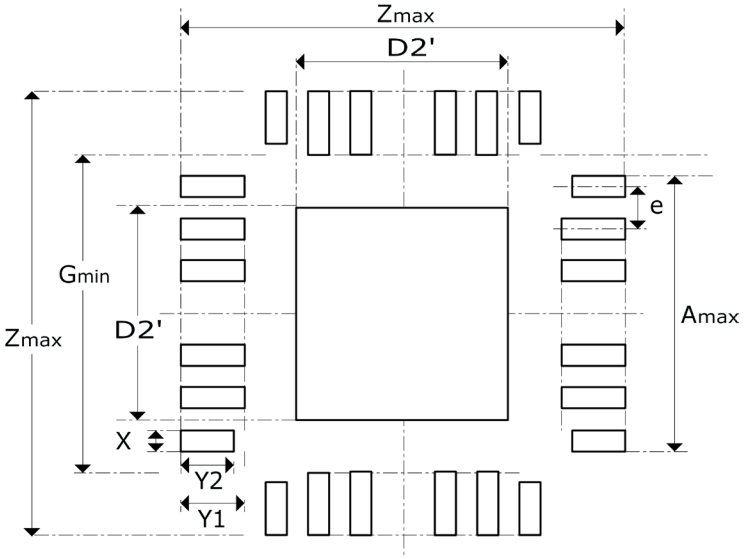


SYMBOL	COMMON					
	DIMENSIONS MILLIMETER			DIMENSIONS INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.75	0.80	0.027	0.029	0.031
A2		0.200 REF.			0.0078 REF.	
b	0.15	0.20	0.25	0.006	0.008	0.010
D	6.90	7.00	7.10	0.271	0.275	0.279
D2	5.20	5.30	5.40	0.205	0.209	0.213
e		0.40 TYP			0.016 TYP	
E	6.90	7.00	7.10	0.271	0.275	0.279
E2	5.20	5.30	5.40	0.205	0.209	0.213
k	0.25	0.35	0.45	0.010	0.014	0.018
k1	0.35	0.45	0.55	0.014	0.018	0.022
L	0.40	0.50	0.60	0.016	0.020	0.024
L1	0.30	0.40	0.50	0.012	0.016	0.020

QFN56 Recommended Pad Layout

		mm	inch
e	=	0.4	0.016
Gmin	=	6.3	0.248
Zmax	=	8.0	0.315
D2'	=	5.4	0.213
Amax	=	5.45	0.215
X	=	0.25	0.010
Y1	=	0.85	0.033
Y2	=	0.75	0.030

Figure 2.9 Pad Layout



Note: Size of ground plane may not be reduced. It should not contain any vias.

RoHS: PS081FN in QFN56 is RoHS compliant

Material list:	Lead frame	C194 Cu with PPF finish (NiPdAU)
	Die Attach	Ablebond 8600, ABlestik
	Bond wires	Gold
	Mold	CEL9220HF13H, Hitachi
	Marking	Laser

Moisture Sensitivity Level 1 (JEDEC J-STD-020,033)

Reflow Soldering Profile

Average ramp-up rate (TL to Tp)		3 °C/second max.
Preheat		
- Temperature Min (TSmin)	140 °C	
- Temperature Max (TSmax)		200 °C
- Time (min to max) ts		60 - 120 seconds
Tsmax to TL		
- Ramp-up rate		3 °C/second max.
Time maintained above:		
- Temperature (TL)		220 °C
- time (tL)		30 seconds
Peak Temperature (Tp)	245 +0 -5 °C	
Time within 5°C of actual Peak	10 seconds	

Die

Figure 2.10 Pad assignment Die

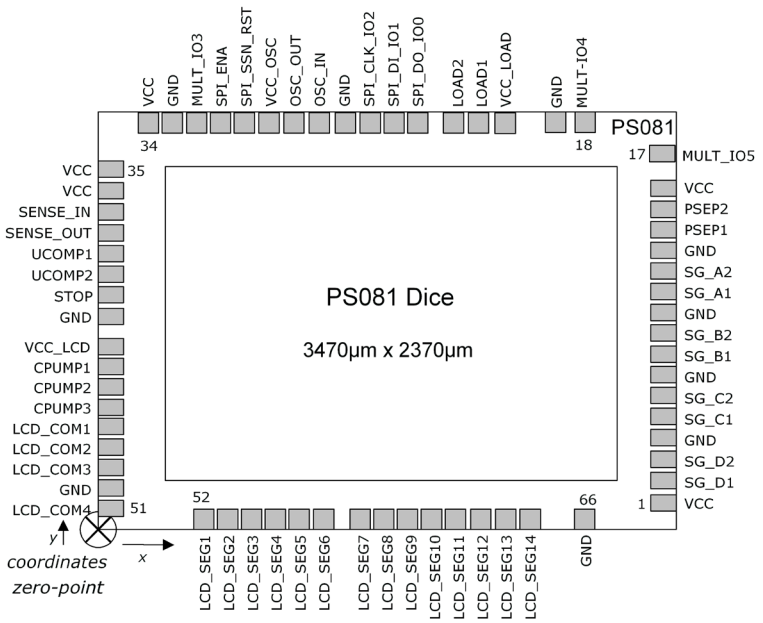


Table 2.13 Pad assignment and location Die

#Pad	Name	Description	Type	X-Pos Center	Y-Pos. Center
Right					
1	VCC	Supply voltage digital part, I/O, 4MHz-osc.		3392	193.8
2	SG_D1	Port 1 halfbridge D	N Open Drain	3392	316
3	SG_D2	Port 2 halfbridge D	N Open Drain	3392	431
4	GND	Ground		3392	546
5	SG_C1	Port 1 halfbridge C	N Open Drain	3392	661
6	SG_C2	Port 2 halfbridge C	N Open Drain	3392	776
7	GND	Ground		3392	891
8	SG_B1	Port 1 halfbridge B	N Open Drain	3392	1006
9	SG_B2	Port 2 halfbridge B	N Open Drain	3392	1121
10	GND	Ground		3392	1236
11	SG_A1	Port 1 halfbridge A	N Open Drain	3392	1351
12	SG_A2	Port 2 halfbridge A	N Open Drain	3392	1466
13	GND	Ground		3392	1581
14	PSEP1	Port 1 temperature measurement	N Open Drain	3392	1696
15	PSEP2	Port 2 temperature measurement	N Open Drain	3392	1811
16	VCC	Supply voltage digital part , I/O, 4MHz-osc.		3392	1926
17	MULT_IO5	Multi purpose I/O no. 5	Mult-IO	3392	2151
Top					
18	MULT_IO4	Multi purpose I/O no. 4	Mult-IO	2862	2286
19	GND	Ground		2730	2286
20	VCC_LOAD	Power supply load output pins 1 and 2		2115	2286
21	LOAD1	Load output to measuring capacitor	P Open Drain	1976.6	2286
22	LOAD2	Load output to measuring capacitor	P Open Drain	1835	2286
23	SPI_DO_IO0	Output serial SPI interface or IO0	Mult-IO	1656.2	2286
24	SPI_DI_IO1	Input serial SPI interface or IO1	Mult-IO	1544.2	2286

25	SPI_CLK_IO2	Clock serial SPI interface or IO2	Mult-IO	1432.2	2286
26	GND	Ground		1320.2	2286
27	OSC_IN	Input to 4MHz ceramic resonator		1208.2	2286
28	OSC_OUT	Output to 4MHz ceramic resonator		1096.2	2286
29	VCC_OSC	4MHz Oscillator supply voltage		984.2	2286
30	SPI_CSN_RST SPI_SSN_RST	Slave select or RST input (High active)	Input with pull-down	872.2	2286
31	SPI_ENA	Serial SPI interface enable		760.2	2286
32	MULT_IO3	Select for Wheatstone comparator MUX or Interrupt or Multi purpose I/O no. 3	Wheatstone select Mult-IO	648.2	2286
33	GND	Ground		536.2	2286
34	Vcc	Supply voltage digital part , I/O, 4MHz-osc.		424.2	2286
Left					
35	Vcc	Supply voltage digital part , I/O, 4MHz-osc.		83	2003.3
36	Vcc-SENSE	Supply voltage SENSE pins		83	1891.3
37	SENSE_IN	Input internal CMOS comparator, connect to Vcc if not used	Analog In	83	1779.3
38	SENSE_OUT	Output internal CMOS comparator	Analog Out	83	1667.3
39	UCOMP1	External comparator circuit connection	Analog Out	83	1555.3
40	UCOMP2	External comparator circuit connection	Analog Out	83	1443.3
41	STOP	Stop input measuring signal		83	1331.3
42	GND	Ground		83	1219.3
43	VCC_LCD	Supply voltage LCD, 10kHz osc., bandgap		83	1045
44	CPUMP1	LCD voltage doubling and stabilization	Analog Out	83	933
45	CPUMP2	LCD voltage doubling and stabilization	Analog Out	83	821
46	CPUMP3	LCD voltage doubling and stabilization	Analog Out	83	709
47	LCD_COM1	LCD line driver for 1/2, 1/3, 1/4 duty	LCD Buffer	83	597
48	LCD_COM2	LCD line driver for 1/2, 1/3, 1/4 duty	LCD Buffer	83	485
49	LCD_COM3	LCD line driver for 1/3, 1/4 duty, row driver for 1/2 duty	LCD Buffer	83	373
50	GND	Ground		83	261
51	LCD_COM4	LCD line driver for 1/4 duty, row driver for 1/2, 1 /3 duty	LCD Buffer	83	149
Bottom					
52	LCD_SEG1	LCD row driver	LCD Buffer	612.6	83
53	LCD_SEG2	LCD row driver	LCD Buffer	724.6	83
54	LCD_SEG3	LCD row driver	LCD Buffer	836.6	83
55	LCD_SEG4	LCD row driver	LCD Buffer	948.6	83
56	LCD_SEG5	LCD row driver	LCD Buffer	1060.6	83
57	LCD_SEG6	LCD row driver	LCD Buffer	1172.6	83
58	LCD_SEG7	LCD row driver	LCD Buffer	1347	83
59	LCD_SEG8	LCD row driver	LCD Buffer	1459	83
60	LCD_SEG9	LCD row driver	LCD Buffer	1571	83
61	LCD_SEG10	LCD row driver	LCD Buffer	1683	83
62	LCD_SEG11	LCD row driver	LCD Buffer	1795	83
63	LCD_SEG12	LCD row driver	LCD Buffer	1907	83
64	LCD_SEG13	LCD row driver	LCD Buffer	2019	83
65	LCD_SEG14	LCD row driver	LCD Buffer	2131	83
66	GND	Ground		2928	83

Dimensions and Pad Opening

The exact Die size is 2.37 x 3.47 mm, the Wafer thickness 725µm. The IC is expected to be used predominantly as Chip On Board (COB). Therefore it is essential to have a Pad Opening that is suitable for bonding machines:

Width: 90µm

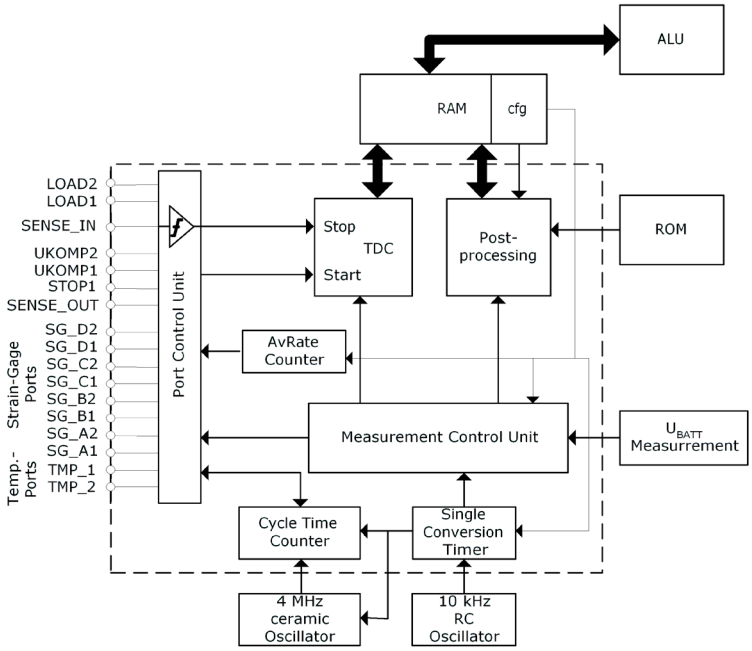
Height: 116µm

Table of Contents Page

3 Converter Front End

3.1 Overview

Figure 3.1 Overview



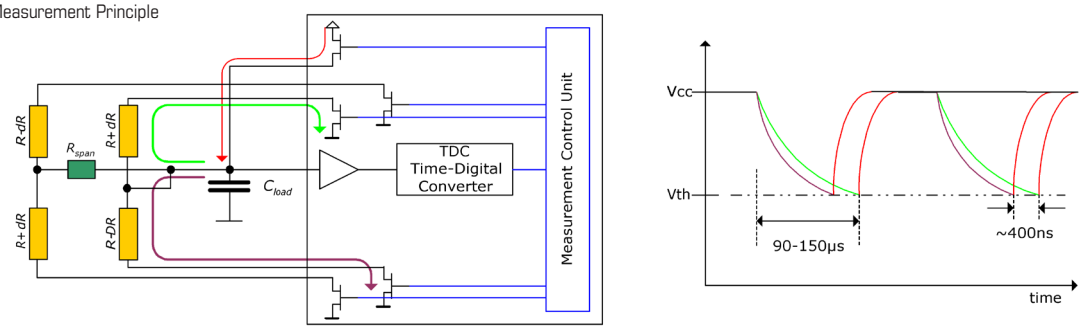
The PICOSTRAIN based converter has the strain gage ports (SG_Ax to SG_Dx) to measure:

- 4 independent half bridges (quattro mode)
- 2 half bridges that form a full bridge
- 2 independent half bridges
- 1 classical Wheatstone bridge
- 1 single half bridge

3.2 Measurement Principle

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. Both, the strain gage with positive change and the one with negative change are measured. The ratio of the two discharge times provides the strain information. The precision of the time measurement is done with about 15 ps resolution (0.5 ps with averaging).

Figure 3.2 Measurement Principle



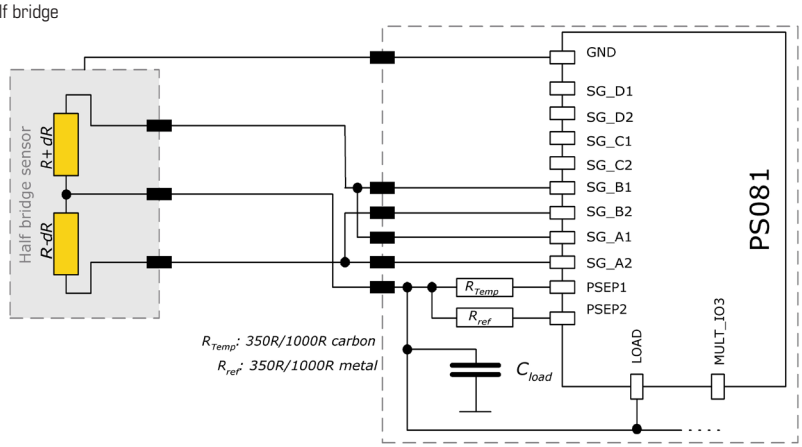
This chapter will explain the components of the front-end, the parameters to set and how to dimension external components.
There are many ways to connect your strain gage sensor to PSØ81. In this section we show how to connect them.

3.3 Connecting the Strain Gauges

Caution: To get good results it is mandatory to connect the load cell body to GND of the electronic. A simple standard wire is sufficient.

3.3.1 Half Bridge

Figure 3.3 Connecting a half bridge



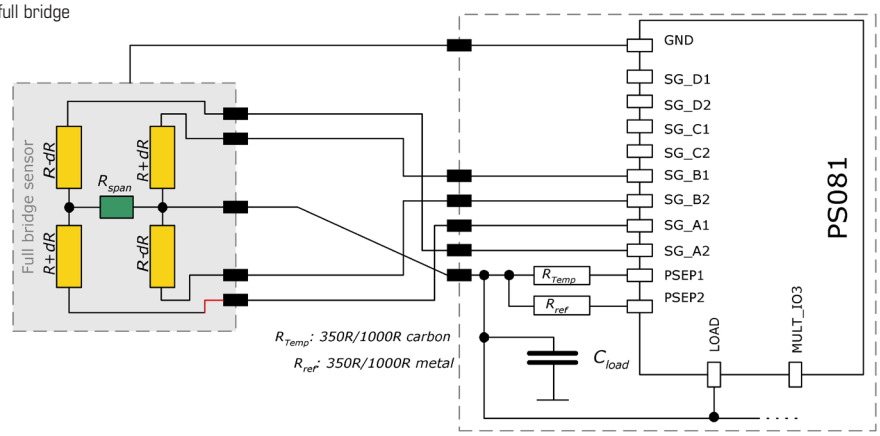
Note:
The half bridge is connected like a full bridge to get a better zero drift behavior. This requires the bridge setting = 1 (register 3, bridge[1:0] = 1)
The multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1.
With this kind of connection the zero drift of the electronic is the same as in full bridge mode and deeply within OIML specification.

For maximum speed it may be helpful to connect the half bridge as a half bridge and not as a full bridge (register 3, bridge[1:0] = 0, pins SG_A1 & SG_A2 only). With AVRate = 2 the maximum speed is possible and up to 1 kHz can be reached. An additional systematic zero drift will occur which is nearly the same on all devices. The value of this zero drift is approx. ± 6 nV/V/K. This zero drift is a result of a longer bond wire on SG_A2 compared to SG_A1. The bond wire resistance is not compensated by the RDSON compensation of the chip. Therefore, this systematic drift can be compensated on the PCB by a longer wire on SG_A1.

When using a standard PCB with cooper layers 35 µm thick, the SG_A1 trace on the PCB from the chip to the connecting pads of the load cell should be (t.b.d.) mm longer than the SG_A2 trace (if a width of 8 mil is used). This compensation method is very reliable and stable an gives a zero drift behavior deeply within OIML specifications.

3.3.2 Full Bridge

Figure 3.4 Connecting a full bridge



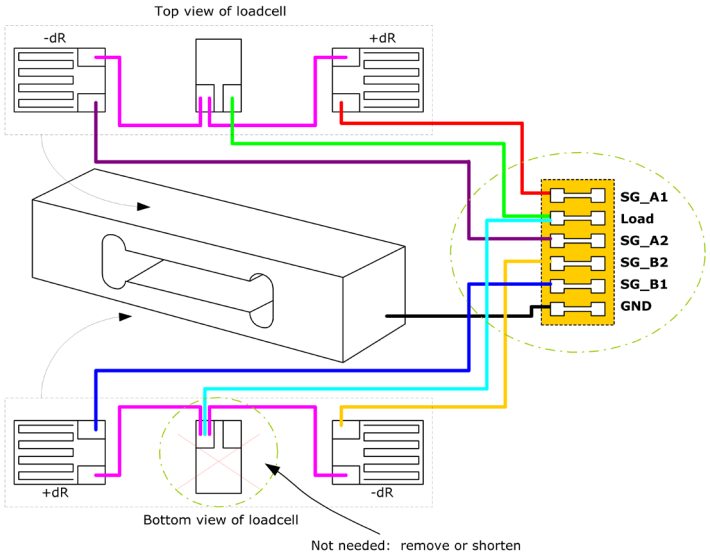
Note:

This is the standard PICOSTRAIN bridge (a full bridge made of 2 half bridges with a single Rspan resistor optionally). The bridge setting is 1 (register 3, bridge[1:0] = 1).

The multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1. Therefore, it is necessary to follow exactly the wiring with respect to positive and negative strain.

Existing sensors with Wheatstone bridge connection can be adopted easily by changing the wiring in the patch-field of the load cell as shown in the following picture:

Figure 3.5 Adapted load cell wiring



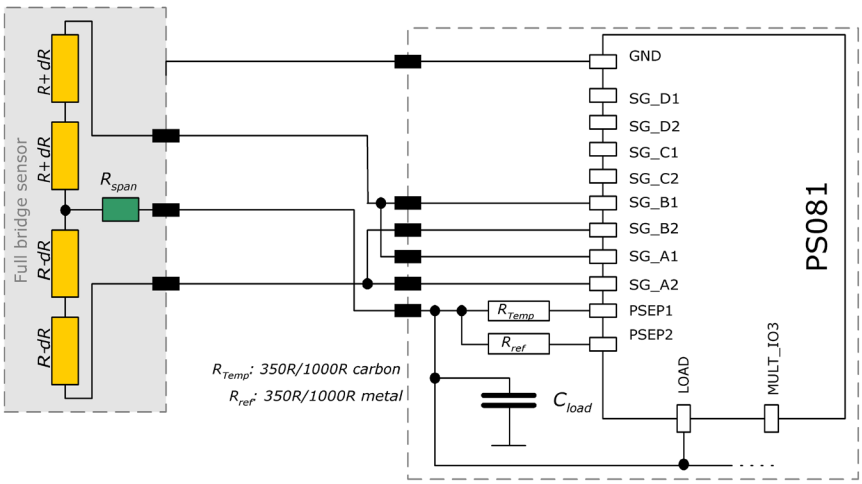
The advantage of the PICOSTRAIN full bridge compared with the Wheatstone bridge is a higher resolution of approximately 0.6 bits (factor 1.5 higher).

3.3.3 Full Bridge Parallel (zero drift optimized)

This mode is not recommended any longer (canceled in December 2009). Please see bug report (Section 7.1) for further details.

3.3.4 Full Bridge connected as Half Bridge (Current Saving Connection)

Figure 3.6 Current saving full bridge wiring



Note:

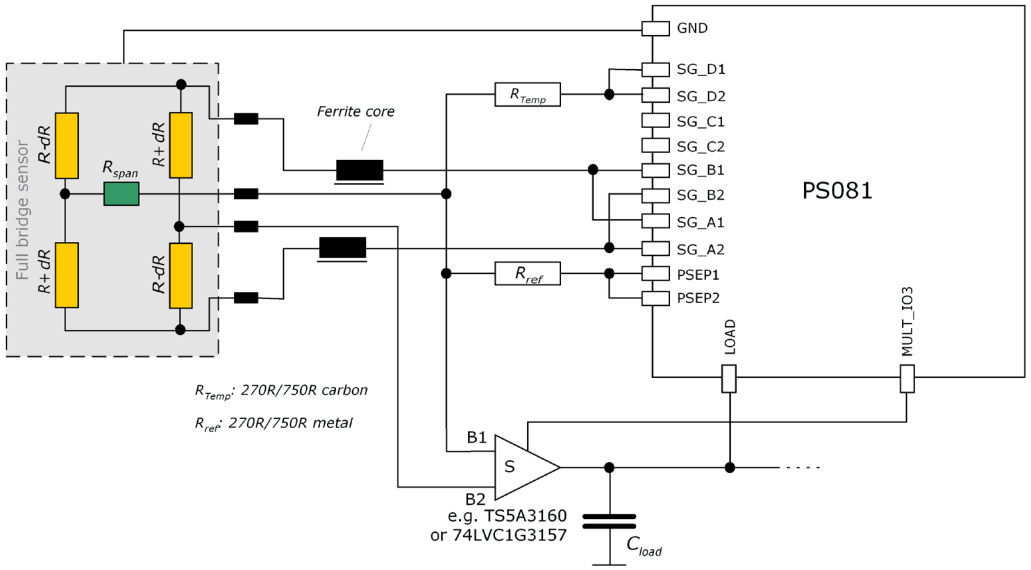
This wiring is suited if all strain gages are on the same site (top OR bottom) of the load cell. Then 2 strain gage resistors can be connected in series to get a 2 kOhm half bridge. This way, the current into the sensor is reduced by factor 2 and therefore this wiring is especially suited for minimum current e.g. solar driven applications.

The bridge setting is 1 (register 3, bridge[1:0] = 1).

The multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1.

3.3.5 Wheatstone Bridge

Figure 3.7 Connecting a Wheaston bridge



For Wheatstone bridges an additional external analog switch is needed. We recommend TS5A3160 because it has a good behavior even at supply voltages lower than 2.7 V. For supply voltages of 3.0 V or higher 74LVC1G3157 is a good choice, too.

Note:

In Wheatstone mode the system looses 0.6 bit of resolution. Because of this, Wheatstone connection is only recommended for applications with long wires (> 1 m) and for first tests if you don't want to modify your load cell wiring.

The bridge setting is 1 (register 3, bridge[1:0] = 1).

The multiplication factors must have opposite sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = -1.

If the Wheatstone bridge has a gain compensation resistor (Rspan) the standard setting for TKGain is 0.75 (ConfigReg08). The factor 0.75 doesn't modify the span compensation behavior of the load cell. In any case "Mod_Rspan" has to be set to 1 (ConfigReg01, Bit 6)

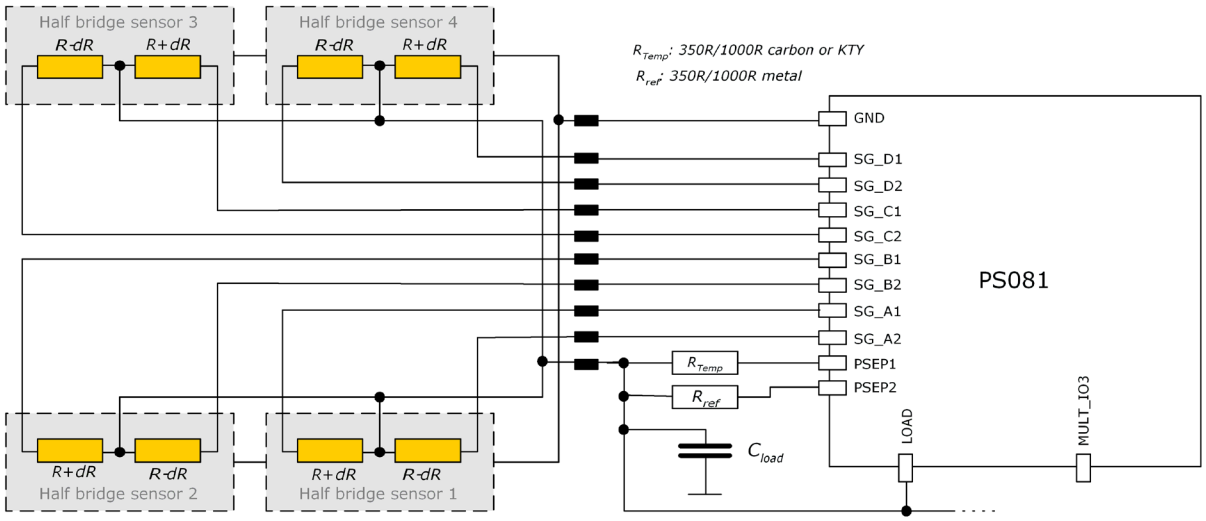
To avoid reflections in the Wheatstone bridge we do strongly recommend the use of ferrite cores. They are placed in the two lines which are connected directly to PS081. Ordinary (SMD-)ferrite cores with a damping of 1000hm @ 100MHz with a low DC resistance (<0.10hm) can be used. As a consequence a lower offset drift and better EMI behavior can be expected.

Caution:

Only Wheatstone bridges with one Rspan or without Rspan (uncompensated) can be used. PICOSTRAIN cannot work properly with Wheatstone bridges that have two Rspan.

3.3.6 Quattro Bridge (4 sensors)

Figure 3.8 Connecting a quattro bridge



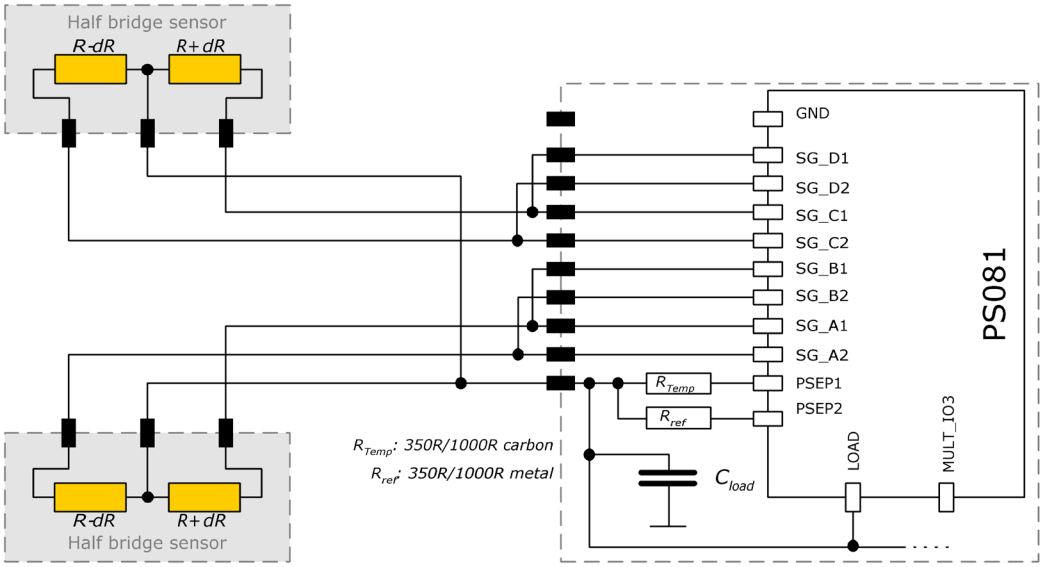
In some cases four sensors are used. Then, each half bridge is connected to one port. This is a typical connection e.g. for quattro body scales. The result of each half bridge can be read but also the overall result.

The bridge setting is 3 (register 3, bridge[1:0] = 3).

Each half bridge is assigned its own multiplication factor. This allows to trim the gain of the four load cells just by software. All multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1, Mult_Hb3 = +1, Mult_Hb4 = +1.

3.3.7 2 Half Bridges separately

Figure 3.9 Connecting 2 separate half bridges



Note:

Normally, two half bridges are wired as full bridge (one result). Nevertheless, sometimes the result of the half bridge is of interest and the two half bridges shall be measured separately. In this case, the two half bridges can be connected in the quattro mode as shown in the picture above and so the results can be read separately. Connecting this way guarantees that the results are gain-compensated.

The result of each half bridge can be calculated then as follows:

HB1 = (A-B) / 2 and HB2 = (C-D) / 2.

If you want to read 2 half bridges separately but with a low offset drift, please connect like suggested in ‚Full bridge' and contact the acam team for further steps.

The bridge setting is 3 (register 3, bridge[1:0] = 3).

All multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1, Mult_Hb3 = +1, Mult_Hb4 = +1.

3.3.8 6-wire Technology

PS081 may be connected to the load cell also in 6-wire technology. Figures 3.11 and 3.12 show the wiring. 6-wire technology is used to compensate the resistance of the sensor cable. With PS081 the cable resistance is compensated as part of the comparator gain compensation (Mult_PP, see also section 3.6.2).

Figure 3.10 6-wire technology with PICOSTRAIN wiring

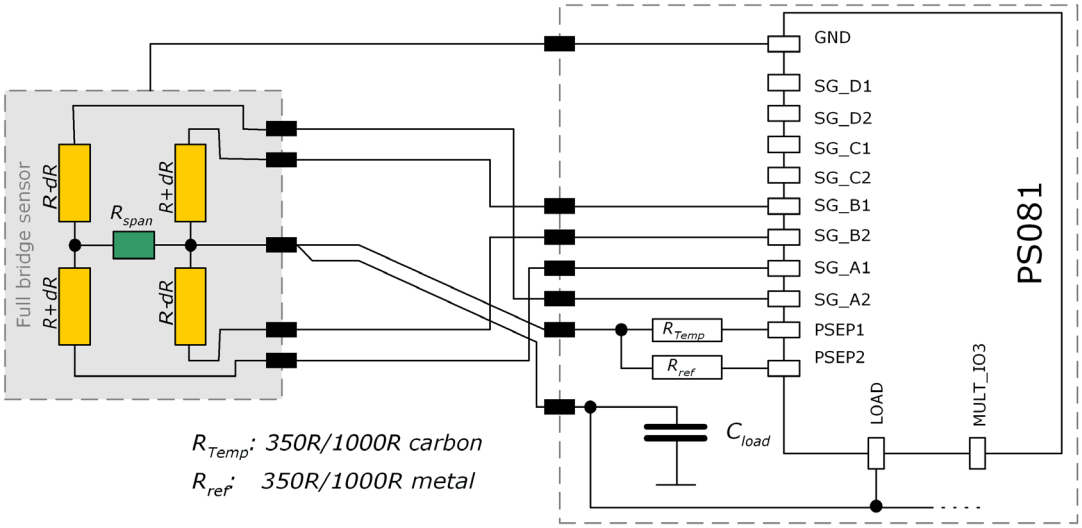
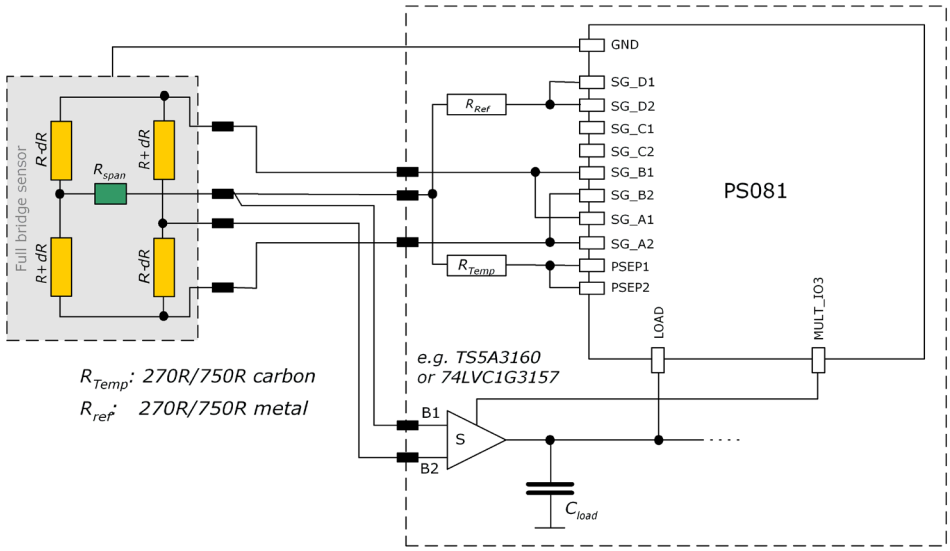


Figure 3.11 6-wire technology with Wheatstone wiring



3.4 Capacitor, Cycle Time, Averaging

3.4.1 Load Capacitor (Cload)

The load capacitor is an important part of the circuit and has direct influence on the quality of the measurement and the temperature stability. Therefore, we recommend the following values and materials:

Rsg = 350R ⟶ Cload = 300nF to 400nF

Rsg = 1000R ⟶ Cload = 100nF to 150nF

Cload can be calculated by = 0.7 x Rsg x Cload ≈ 100 µs – 150 µs

Recommended materials:

- COG* for highest accuracy
- CFCAP good, but not as good as COG
- X7R with some minor losses in temperature stability
- Polyester with some minor losses in temperature stability

We do not recommend the use of ZOG capacitors !

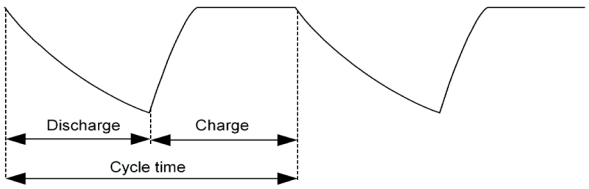
* COG capacitor up to 100nF are available by Murata GRM31 series
** Multi layer ceramic capacitor from Taiyo Yuden

Note:
COG capacitors are definitely the best choice for high end applications (e.g. 6000 divisions (or higher) legal-for-trade scales). CFCAP are also a good choice for high end scales and legal-for-trade scales. For consumer scales X7R are the first choice because of their low cost. But they introduce additional gain drift at lower temperatures (< +5 °C).
For consumer applications also a lot of other capacitors are well suited (e.g. Polyester).

3.4.2 Cycle Time (cytime)

The cycle time is the time interval between subsequent discharge time measurements. It covers the discharge time and the time to charge again Cload. Following figure illustrates this relation.

Figure 3.12 Cycle time



The discharge time is given by the value of the strain gage resistor and the given capacitor Cload. The recommended discharge time is in the range of 80 to 150 µs (at 3.3V). The charge time has to be long enough to provide a full recharge of Cload and is typically 30% of the cycle time. If the cycle time is set too small (in the range of the discharge time or smaller) an overflow will occur.

The cycle time is set in register 2, cytime[13:4]. The cycle time is normally generated by the high speed clock and can be set in steps of 2 µs. The only exception is the "Stretched Mode" (see chapter ‚Modes' 3.5) where the cycle time is generated by the internal 10 kHz oscillator and therefore configurable in steps of 100 µs.

Example:

cytime[13:4] = 80 → 80 x 2 µs = 160 µs cycle time in all modes except stretched mode

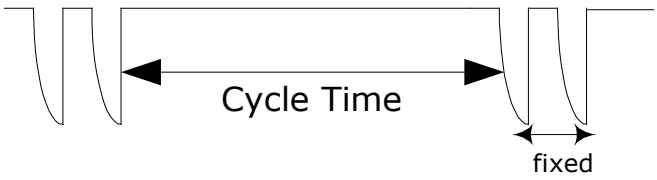
cytime[13:4] = 10 → 10 x 100 µs = 1 ms cycle time in stretched mode

The recommended minimum cycle time setting is 1.4 times the discharge time. E.g. 140 µs if the discharge time is 100 µs.

3.4.3 Cycle Time in Stretched Mode

In stretched mode the parameter cyctime has a special function. In this case, it does NOT define the time of discharging + charging, instead it defines the time between 2 discharging cycles in multiples of 100 µs:

Figure 3.13: Cycle time in streched mode



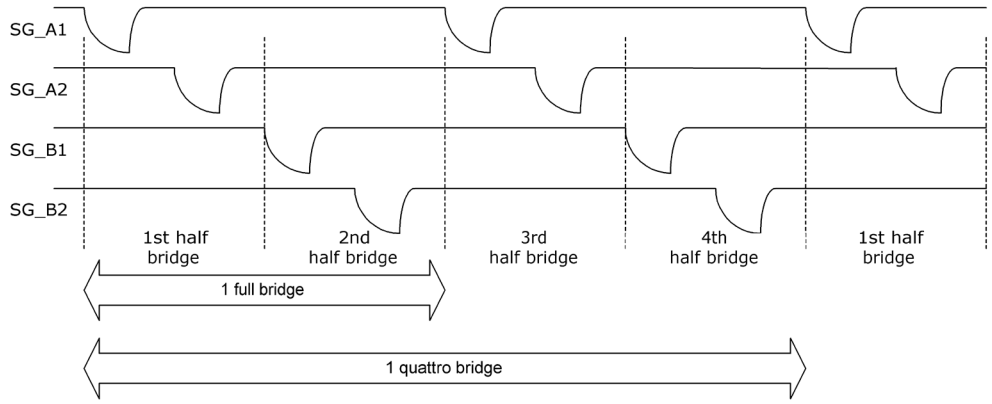
There are several parameters to adjust in stretched mode. Please have a look at Stretched Mode settings in section 3.5.3.

3.4.4 Averaging (avrate)

The number of strain gages respectively half bridges connected defines how many discharging cycles are needed to make one complete ratio measurement:

Half bridge	→	2 cycles
Full bridge	→	4 cycles
Quattro bridge	→	8 cycles

Figure 3.14 Discharge cycles for a complete measurement

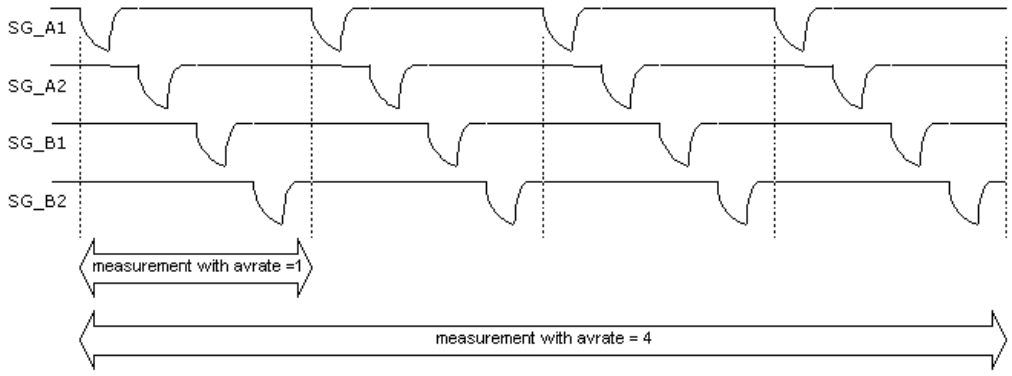


Those numbers of cycles for each mode together define 1 sample (avrate=1). This is also the minimum needed for one complete ratio measurement.

3.4.5 Better resolution by averaging

In PS081, the resolution can be increased by internal averaging. The sample size of the averaging is specified by parameter avrate in register 2. The standard deviation of the result will be improved by nearly the square root of the sample size. The following picture shows the correlation for a full bridge:

Figure 3.15 Averaging



One sequence in this example is made of 4 samples, each made of 4 discharge cycles. So in total 1 measurement takes 16 discharge cycles.
Besides the discharging cycles given by the sample there are additional measurements like gain compensation or fake measurements for better stability. All those measurements together form in total then a measurement sequence. In other words, a sequence contains all measurements needed to get the final result. It also defines the total conversion time. For more details on conversion time please see the chapters ‚Conversion Time' 3.5.5 and ‚Modes' 3.5.
Of course, the sample size of averaging dominates the update rate. While the resolution is improved by a factor 1/√avrate, the maximum update rate is reduced by the factor avrate.

sample size (avrate) ↗ → Resolution ↗
→ Max. update rate ↘

Also the lowest possible current consumption is influenced by the sample size. It grows by a factor avrate.

sample size (avrate) ↗ → Minimum current ↗

Recommendation: We strongly recommend not to use avrate = 1. In principle it works but the drift significantly increases and it can be used only for low end resolution applications. The recommended minimum sample size is avrate = 2. It is also not recommended to use odd numbers at lower avrate up to 50. E.g., do not use avrate = 7, use instead avrate = 8 or avrate = 6.
Important: At low avrates (<= 32) the factor ps081adjust (Configreg_03, Bit [9:4]) should be set to 2x avrate. Example: avrate = 8 --> set ps081adjust to 16

3.4.6 Resolution and Converter Precision

In this document the terms resolution and converter precision are often used in the same context, however, there is a difference in their meaning:

- Resolution: refers to the digital value which can be displayed (or resolved) within the chip. This is basically the HBO register in a 24-bit format, where the MSB is indicating a negative number (two's complement). Expressed in numbers the result can be shown from -8388608 (0x800000) to +8388607 (0x7FFFFF). One LSB has thereby the valency of 10nV/V (2mV/V divided by 200,000). Example: at 3V the valency of 1 LSB is 10nV/V x 3V = 30nV.

- Converter Precision (sometimes also refered to as „accuracy"): this is the accuracy given by the converter, normally defined by the standard deviation or RMS (root mean square) noise. The value of the precision is normally given in effective number of bits (ENOB). With PS081 an RMS

noise as low as 10nV at 3.3V can be achieved, or expressed in ENOB up to19.5 Bits (related to 2mV/V). An overview of the converter precision at different update rates is given in several tables in section 2.4. However, a rough estimation can be done by the equations given in the following.

The base converter precision for a half bridge at avrate = 1 (only for calculation purposes, not recommended to be used) and a recommended discharge time of 90 to 150 µs in fast settling mode and 2 mV/V excitation is:

With internal comparator: 13.3 Bit eff.

With external bipolar comparator: 13.8 Bit eff.

At higher values of avrate[] the resolution is calculated as:

$$ENOB = ENOB[AVRate = 1] + \frac{\ln(\sqrt{AVRate * Bridgefactor})}{\ln(2)}$$

The Bridge-factor is: 2 for full bridges

4 for quattro bridges

Example 1:

avrate = 12, Quattro bridge, internal comparator

ENOB = 13.3 + ln(√(12*4))/ln(2) = 13.3 + 2.8 = 16.1 Bit eff. = 70,000 effective divisions = 10,000 peak-peak divisions in fast settle mode (without SINC-filter).

Example 2:

avrate = 450, Full bridge, external comparator

ENOB = 13.8 + ln (√(450*2))/ln(2) = 13.8 + 4.9 = 18.7 Bit eff. = 425,000 effective divisions = 70,000 peak-peak divisions in fast settle mode (without SINC filter).

Example 3:

avrate = 100, Full bridge, external comparator, expressed in nV RMS

ENOB = 13.8 + ln (√(100*2))/ln(2) = 13.8 + 3.8 = 17.6 Bit eff.

2^17.6 = 198,700 eff. divisions with a 2mV/V sensor operated at 3V

→ 3V x 2mV/V = 6000µV divided by 198,700 = 30.2nV RMS

Note: The effective number of bits (ENOB) in the equation are related to 2mV/V sensitivity of the sensor. If the sensitivity is different with your sensor, the result needs to be corrected by the reduction in sensitivity. E.g.: you calculate 18.7 bit with the equation, but your sensor has only 1mV/V

instead of 2mV/V. Then this corresponds to a reduction by factor 2, which is 1-bit, so the ENOB is 17.7 bit in this case.

The effective resolution (ENOB) is also reduced when Wheatstone connection is used instead of PICOSTRAIN connection. The reason for that is the reduction of the strain by 1/3 because when discharging over 1 strain gage of the bridge the other 3 strain gages are in parallel and lower the extension/strain of the gage to measure. Expressed in ENOB the reduction is -0.6 bit.

3.5 Modes and Timings

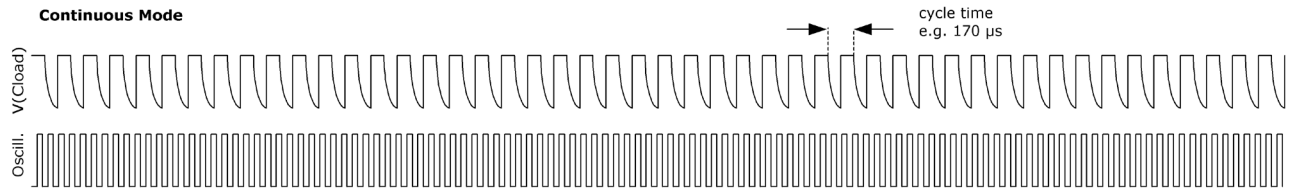
The PS081 has 3 basic operating modes as well as combinations of them. They are related to the sampling frequency and the active time of the 4 MHz oscillator. Therefore, the selection has influence on the stability of the result and the current consumption.

The basic modes are:

- Continuous Mode
- Single Conversion Mode
- Stretched Mode

3.5.1 Continuous Mode

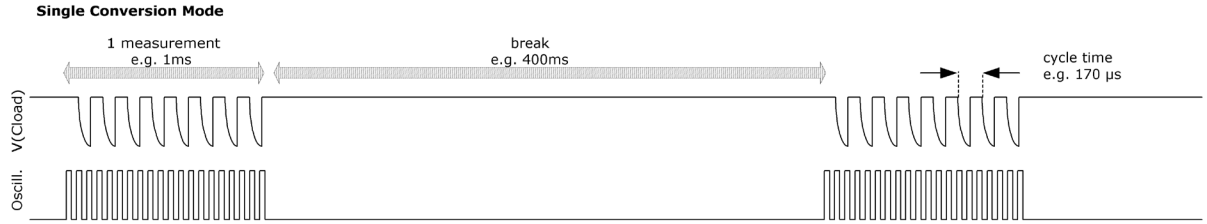
Figure 3.16 Continuous mode



The chip is making continuously discharge time measurements. The oscillator is on all the time.This mode is the choice for applications targeting highest resolution. It is the standard mode for all applications that allow a current consumption > 500 µA.

3.5.2 Single Conversion Mode

Figure 3.17 Single conversion mode



The chip makes a complete measurement sequence and then goes to sleep mode. The oscillator is

on only for the sequence. This mode offers the lowest current consumption and is best choice for body scales.

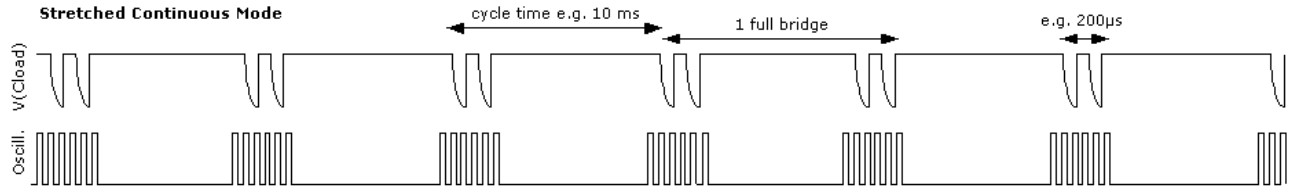
Pure Single Conversion Mode should be used only in mechanically stable systems like body scales, because it implies undersampling. The consequence of undersampling is that mechanical oscillations of the weighing system will end up in unstable data.

3.5.3 Stretched Mode

Streched Mode combines the advantage of a few measurements (to save current) and a reasonable distribution of these measurements for avoiding undersampling. Hence, the discharge cycles are stretched in a way that the total number is not increased but the distribution is improved.

3.5.3.1 Stretched Continuous Mode

Figure 3.18 Stretched continuous mode

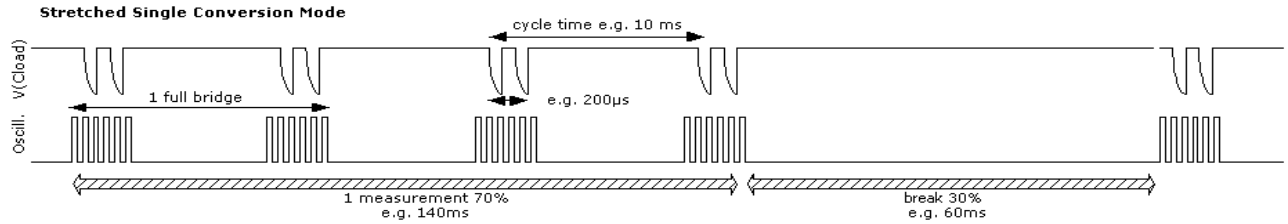


Stretched Conitnuous Mode combines stretched mode and continuous mode. There are longer intervals between the discharge time measurements for the half bridges. The oscillator is activated only for each half bridge measurement.

This mode is used in applications that target high resolution at low current (< 500 µA). It also has a good frequency response (e.g. load cell vibrations) on the input signal. The response can easily be calculated by the Nyquist theorem. This mode together with a good software anti-vibration filter gives best vibration suppression at lowest current. This mode is recommended e.g. for battery driven solar kitchen scales.

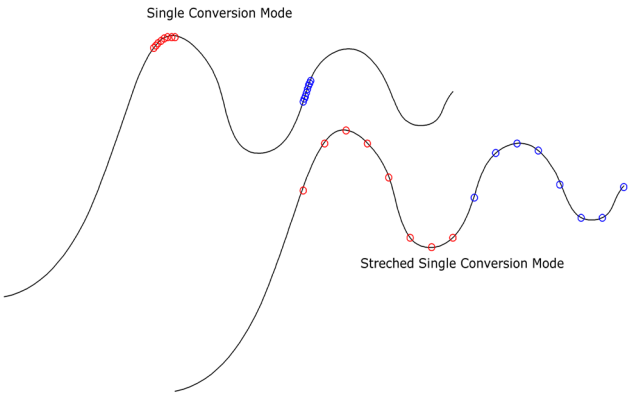
3.5.3.2 Stretched Single Conversion Mode

Figure 3.19 Stretched single conversion mode



For mechanically sensitive weigh scales like kitchen scales the PS081 provides the stretched mode combined with the single conversion mode. In this mode the two resistors of a half bridge are measured subsequently, but the next pair of discharge time measurements follows delayed. Therefore, the sample points of a single sequence can cover minimum a full period of the mechanical oscillation. Thanks to the integration of the samples within one sequence the result will normally be stable if the break is <30% of the time.

Figure 3.20 Undersampling



Again, the oscillator is switched on only for each discharge time measurement. But, as the oscillator needs some time to reach the full amplitude, the total active time of the oscillator is longer than for pure Single Conversion Mode. The current consumption in stretched single conversion mode is therefore a little bit increased compared to the single conversion mode.

Four major parameters define the operation mode:

single_conversion: Selects between continuous operation and single separated measurements.

stretch: Selects between 4 MHz oscillator continuously running while measuring and running the oscillator only for the duration of 1 or 2 discharge cycles (recommended 2 discharge cycles)

cycletime: Defines the time interval between single or pairs of discharge cycles. It is based on the 4 MHz clock or in stretched mode on the 10 kHz clock.

avrate: Sample size of averaging. Defines the number of complete ratio measurements that make a single measurement sequence (internal averaging).

single_conversion / continuous

Configuration:	Register 2, Bit 2: single_conversion
single_conversion = 0	Selects continuous mode. In this mode the PSØ81 is continuously measuring. The 4 MHz oscillator is on continuously. This takes about 130 µA @ 3.0 V.
single_conversion = 1	Selects single conversion mode. In this mode the PSØ81 makes one complete measurement and then switches off the 4 MHz oscillator for the duration of the single conversion counter.

stretch

Configuration:	Register 3, Bits 12, 13: stretch
stretch = 0	off
stretch = 1	The 4 MHz oscillator is on only for the duration of a single discharge time measure ment. The cycle time (time between subsequent discharge time measurements) is calculated on the basis of the 10 kHz oscillator. - not recommended -
stretch = 2 or 3	The 4 MHz oscillator is on only for the duration of a single half bridge measure-ment (two discharge time measurements). The cycle time (time between subse-quent half bridge time measurements) is calculated on the basis of the 10 kHz oscillator. The time interval between the two discharge time measurement for a halfbridge is 200 µs in case stretch = 2 or 300 µs in case stretch = 3

Stretched Mode Settings

In stretched mode there are several parameters which configure the mode, these are:

stretch[13:12] in Configreg_03

cytime[13:4] in Configreg_02

sel_start_osz[19:17] in Configreg_03

single_conversion [2] in Configreg_02

tdc_conv_cnt[23:16] in Configreg_00

Those parameters set the stretch mode. The following 2 pictures show how the parameters are applied, one showing the continious stretched the other the single conversion stretched mode.

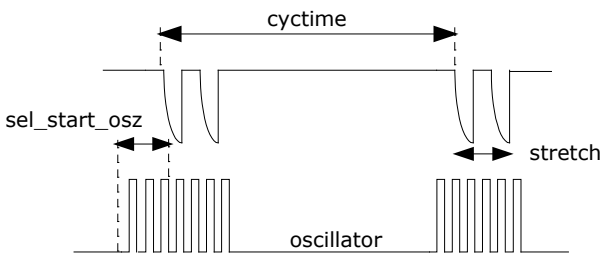
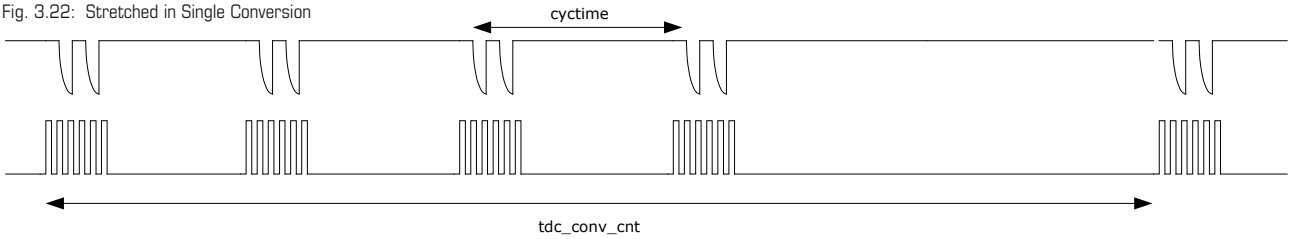


Fig. 3.21: Parameters in continious stretched mode

The cyctime-parameter defines the time waited for the next discharging. By the parameter stretch, the time between 2 discharging cycles can be defined. The parameter sel_start_osz defines what time the oscillator is started before the discharging cycles are coming.

Fig. 3.22: Stretched in Single Conversion



Basically in Single Conversion Stretched mode the parameters remain the same. But tdc_conv_cnt defines additionally the time between measurement sequences.

3.5.4 Mode Selection Criteria

Table 3.1 Mode Selection criteria

Applications	Mode		Parameters	Description
Highest resolution with no current limitation Standard mode for all applications with > 500 µA current capability	Stretched / Continuous	Continuous mode	single_conversion = 0 stretch = 0	Continuously measuring, 4 MHz oscillator on all the time
High resolution but lower current		Stretched continuous mode	single_conversion = 0 stretch = 2 or 3 cycle time = cytime*100µs	Continuously measuring. 4 MHz oscillator on only during the discharge time measurement.
Lowest current consumption Mechanically stable applications like pressure sensors	Stretched / Single conversion	Single conversion mode	single_conversion = 1 stretch = 0	option with lowest current consumption, undersampling -> no suppression of mechanical vibrations
High resolution but low current, e.g. battery driven legal-for-trade scales with 3000 divisions		Stretched mode	single_conversion = 0 stretch = 2 cycle time = cytime*100µs	option with low current consumption and oversampling for suppression of mechanical vibrations.
High resolution but lowest current, e.g. solar scales		Stretched single conversion mode	single_conversion = 1 stretch = 2 or 3 cycle time = cytime*100µs	option with very low current consumption and oversampling for suppression of mechanical vibrations.

3.5.5 Conversion Time / Measuring Rate (Continuous Mode)

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

Tconversion = CycleTime*(2*avrate * Bridge-factor + 6 +MFake*2 + 1)

Mfake = #Fake measurements, Temperature measurement on

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

Note:
Fake measurements are necessary to avoid that the next measurement starts while the ALU is still processing data from the former measurement.

Example1: Cycle time = 110 µs
AVRate =12
Quattro bridge
Mfake = 1
Tconversion = 110 µs*(2*12*4 + 6 + 2 + 1) =11.55 ms
The maximum measuring rate is 86.6 Hz

Example2: Cycle time = 110 µs
AVRate = 450
Full bridge
Mfake = 2
Tconversion = 110 µs*(2*450*2 + 6 + 4 + 1) =199.21 ms
The maximum measuring rate is 5.02 Hz

3.5.6 Conversion Time / Measuring Rate (Single Conversion Mode)

If PSØ81 is configured to run in Single Conversion Mode (Bit 4 in configreg_02), the measuring rate is defined by the value in tdc_conv_cnt[23:16] in configreg_00. This value corresponds directly to the conversion time (multiplied by 6.4ms).

Example:

configreg_00: 0x158200 → tdc_conv_cnt[23:16] = 0x15 = 21 decimal
→ 21 x 6.4ms = 0.1344 second
→ measuring rate = 1 / 0.1344 second = 7.44 Hz

Note:
In case you use single conversion the time needed for one complete measurement should fit into the time slot given through the conversion counter (tdc_conv_cnt).

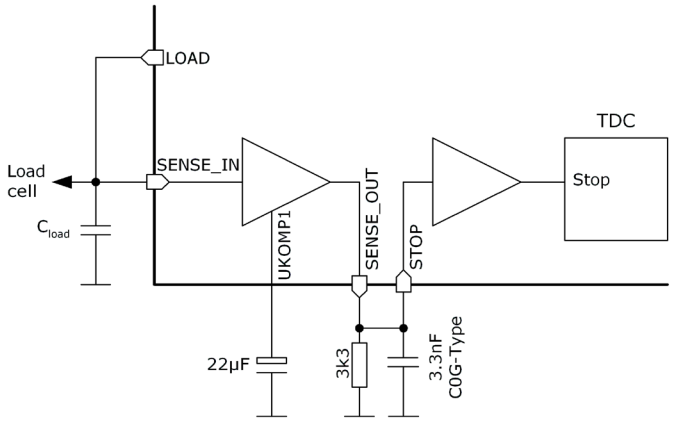
3.5.7 Comparator

The end of the discharge cycle is triggered by a comparator. PSØ8 offers an internal low noise comparator. Alternatively, an external comparator might be used.

3.5.7.1 Internal Comparator

The internal comparator is selected by setting reg11, sel_compint = 1. By means of the internal comparator it is possible to get about 60,000 divisions peak-peak at 2 mV/V, 5 Hz update rate and MEDIAN 5 software filter.

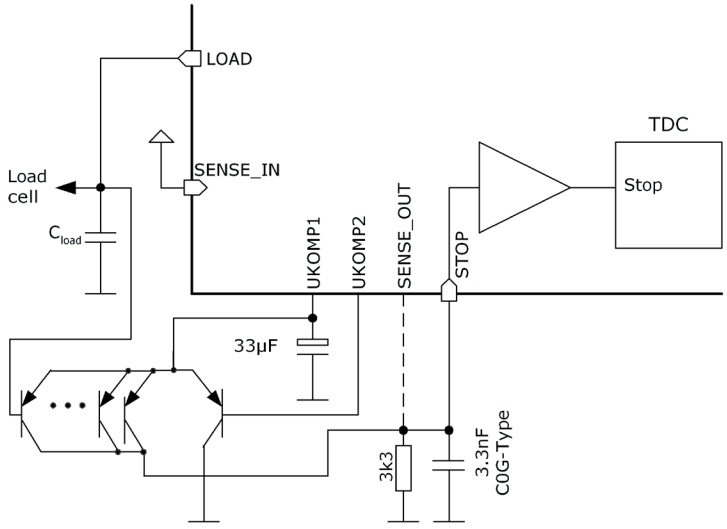
Figure 3.23 Internal comparator



3.5.7.2 External Comparator

The precision of the measurement can be improved by using an external bipolar comparator. With an external bipolar comparator it is possible to get up to 150k divisions at 5Hz update rate.

Figure 3.24 External comparator



Recommendations:

Low-noise PNP transistors like 2N5087 / CMKT5087 or BC859 should be used. 5 transistors in parallel should be connected at the LOAD side. It is not necessary to have matched transistors. Use a COG-type capacitor for the low-pass filter capacitance.

3.5.7.3 When should an external comparator be used?

There are three reasons for the choice of an external comparator:

a) Very high resolution

The user is looking for the best possible resolution in his application, e.g. in counting scales, high end legal-for-trade scales.

b) Lowest current

The user is looking 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 be more than halved compared to the internal comparator.

c) Ultra low voltage

In case the user wants to run his application down to < 2.1 V Vcc, e.g. with 1.55 V silver oxide batteries. Then the bipolar comparator shows significantly better results.

3.5.7.4 Comparator Control

The comparator can be switched on for only the duration of the measurement for current saving reasons or continuously (con_comp[1:0]). 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

SEL_COMPR = 'b10 → 7k resistor selected

If CON_COMP is set to 'b11 (on) the comparator needs approx. 130 µA @ 3.0 V of constant current.

Capacitors at UCOMP1 and STOP:

The capacitors at UCOMP1 and STOP are important for the low noise figure. For best performance we recommend 33 µF for C_{UCOMP1} and 3.3 nF for C_{STOP}. For Cucomp1 an ordinary electrolytic capacitor can be used. As C_{STOP} a X7R capacitor can be used.

In case the internal comparator is used C_{UCOMP1} and C_{STOP} have to be connected as well as the 4.7k Ohm resistor. Nevertheless, smaller values are possible, too.

Recommended values:

C_{UCOMP1}: not below 1 µF

C_{STOP}: lower than Cucomp1/3000

Example:

C_{UCOMP1} = 1 µF à C_{STOP} < 1 µF/3000 → 330 pF selected.

The noise will slightly increase by about 0.3 – 0.5 Bit.

3.5.7.5 Correction of Comparator Delay

The focus of the comparator performance is on ultra low noise, and therefore it has a delay which cannot be neglected. This delay depends on temperature and results in a gain error which is too high for precise weight scale applications. The two resistors Rtemp and Rref are used to measure the delay time periodically during the operation. The PSØ81 corrects the measuring result with the measured delay.

The delay of the comparator depends on the value of C_{UCOMP1} and C_{STOP}. Because these values can be changed by the user there is a possibility to adjust the correction routine by the register Mult_PP[7:0]. A good value for the recommended C_{UCOMP1} and C_{STOP} values (33 µF and 3.3 nF) is Mult_PP = 1.28 (160 or 0xA3). If the capacitor values are increased, 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.

At the correct 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 >130 dB. See also section 3.6.2.

3.5.8 Temperature Measurement

PS081 has the possibility to measure the temperature by means of an external, temperature dependent resistor. The temperature information can be used to correct the gain drift of uncompensated load cells. This we call rspan-by-temp compensation.

Temperature measurement is done by measuring the ratio of the discharge times of two resistors, Rtemp and Rref, a temperature dependent one and a temperature stable one. The change of the ratio is used by an implemented hardware algorithm to correct the gain drift of an uncompensated load cell. Of course, the temperature coefficient of the resistor pair needs to match the load cell's temperature coefficient. This can be done by a factor TKGain (configreg_08), which scales the result

from the temperature measurement before it is used for the correction of the strain measurement.

Make sure that the bits mod_rspan & rspan_by_temp are set (configreg_01, bit 6 and 8, page ALU in the evaluation software).

3.5.8.1 Resistors Rtemp and Rref

The sensitive resistor may be a carbon film resistor with a temperature coefficient in the range of 200-300ppm/K. The reference resistor can be a metal film resistor which have typically <50ppm/K. Please make sure, that the temperature coefficient of the carbon resistor is not <200ppm/K because in this case the temperature measurement will not work properly.

The values for Rtemp and Rref have to be adjusted to the strain gage resistance and the kind of bridge.

Therefore the resistors should have following values:

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

Note:

The two resistors have to be connected in any case as they are used also for the comparator delay correction. In case of no temperature measurement both resistors can be of the same type (e.g. carbon resistors).

The sensitive resistor may be a carbon film resistor with a temperature coefficient in the range of 200-300ppm/K. The reference resistor can be a metal film resistor which have typically <50ppm/K. Please make sure, that the temperature coefficient of the carbon resistor is not <200ppm/K because in this case the temperature measurement will not work properly..

For more information about this method of gain drift correction see application note AN021.

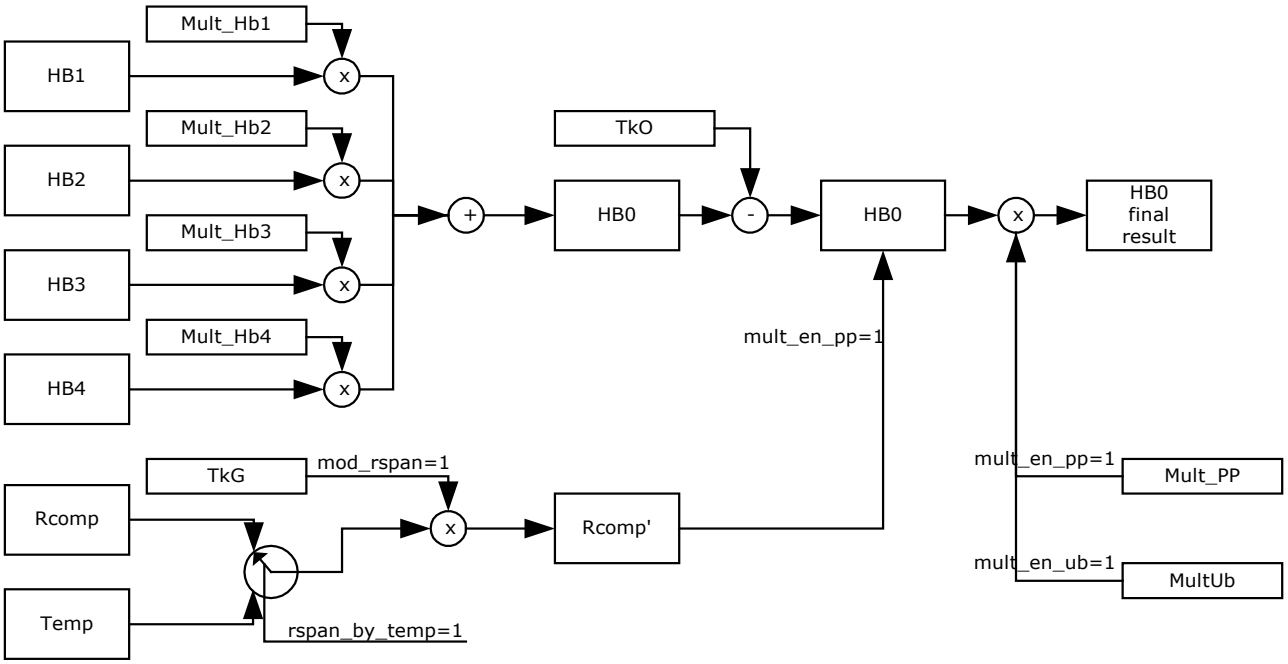
3.6 Post-processing

At the end of a measurement the converter does 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.

Specialties of the post-processing are:

- The results of the four half-bridges have independent multiplication factors. This offers the possibility to do a software correction for off-center weights in quattro applications.
- The strain sensors and the span compensation resistor are separated. The gain compensation resistor can therefore be adjusted by software. Even the temperature measurement can be used instead of the span compensation resistor. By this method it is possible to make high-quality load cells out of standard load cells just by software.
- As a consequence, with PICOSTRAIN the offset is not affected by the span compensation. The offset can be corrected 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.

Figure 3.25 Post-processing



A simple example program to display the results could be:

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

3.6.1 Off-center Correction for Quattro Scales

Several scales like body scales have four load cells, each with a half-bridge sensor on it. The indicated weight might vary with the position on the platform in case the load cells do not all have exactly the same sensitivity. PSØ81 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). By simply 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. Please contact acam for the algorithm to calculate the factors MULT_HB1 to MULT_HB4.

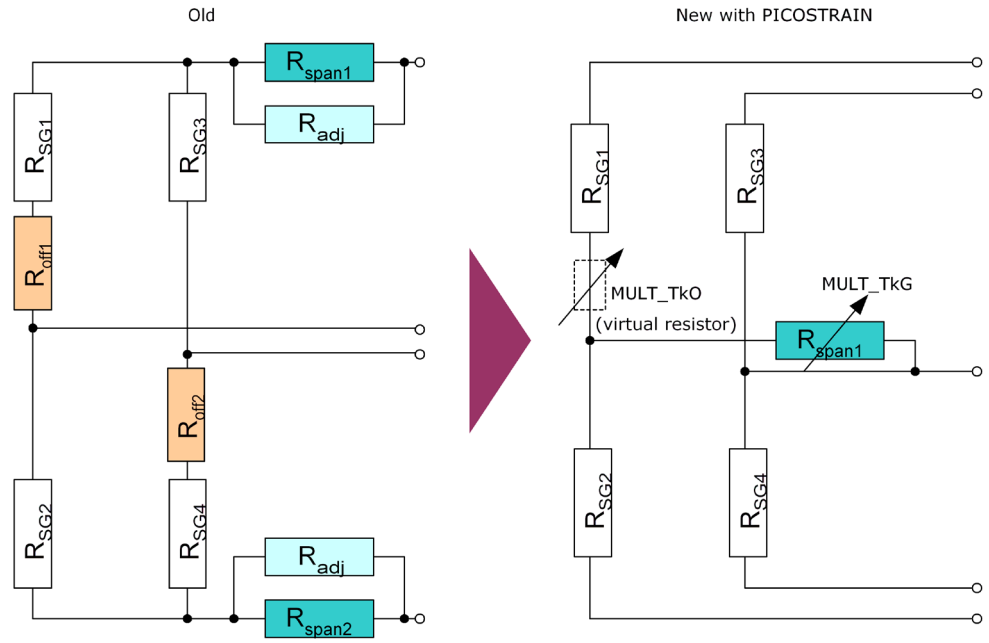
3.6.2 Compensation of Load Cell Gain & Offset Drift (Mult_TKG, Mult_TkO)

Todays high end converters have a very good zero drift and gain drift bevavior. It is about 5 to 10 times better than for a good load cell itself. The real goal is an optimized complete system (scale) and not only a very good electronic. Therefore, with the PICOSTRAIN family acam has indroduced a method which is also able to correct the zero drift and the gain drift of the load cell by software without touching the load cell. This method works only if the load cell has just <u>one</u> compensation resistor (Rspan)[1].

PS081 can measure this resistor and correct it by an algorithm in the µP, based on factors Mult_TkG for the gain drift and Mult_TKO for the zero drift. This can be done after the production of the load cell is completed. It is no longer necessary to have a precise compensation resistor on the load cell. It is not necessary to trim Rspan manually.
A further consequence of this possibility is that it is no longer necessary to trim the load cell exactly to zero, no zero offset compensation resistors are needed.

[1] see also 3.6.4 Nonlinearity of gain drift over temperature. It may be necessary to add a resistor in parallel to Rspan to reduce nonlinearity.

Figure 3.26 Correction of Gain and Offset Drift



By doing this, the gain and offset behavior of the load cell can be improved by PS081. This is a very comfortable method to improve the quality of the complete scale without modifying the load cell or the electronic. If this possibility is combined with an intelligent digital load cell concept the production can be simplified at higher quality level and lower cost.
Some examples how to use Mult_TKO, Mult_TKO:

- If the compensation resistor is matched to the sensor, but the bridge has an offset drift, this offset drift can be eliminated by software.
- If the gain error of the load cell is known (e.g. stable over production lot but wrong) it can be corrected directly by PS081 without going into a climate chamber.
- If a run in the temperature drift chamber is done, the correction factors for Mult_TkG and Mult_TkO can be determined very appropriate. In this case the compensation of the whole system can be improved significantly. With such a method of post correction after fabrication of the scale, the complete scale can be offset and gain adjusted nearly perfect and much better than required for high end scale. (e.g. gain drift < 1 ppm/K and offset drift < 10 nV/K for the complete scale have been achieved as best performance).

acam has written a special whitepaper (WP002) that explains in detail the many possibilities and the importance of this option. Furthermore it provides a step-by-step guidance how to make the temperature compensation by using Mult_TkG and Mult_TkO.

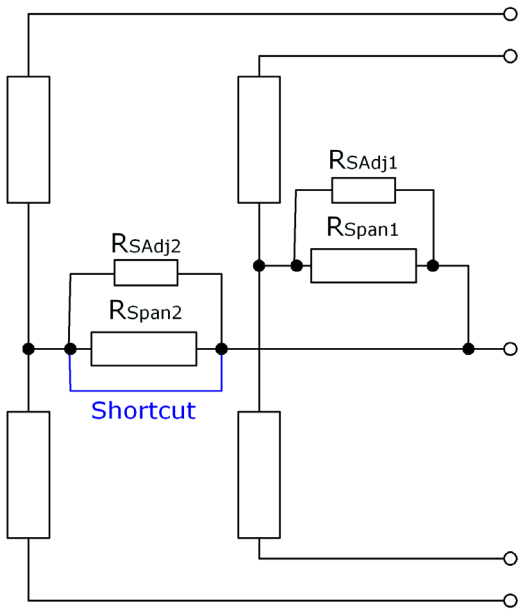
PS081 can correct uncompensated load cells (cells without a gain compensation resistor Rspan), too. It therefore uses the temperature measurement information instead of the Rspan value. This mode is activated by setting rspan_by_temp = 1 in register 1. Again, the adjustments are done by means of factors Mult_TKO and Mult_TKG. The accuracy of this compensation depends mainly on the accuracy and Tk of the resistors used for the temperature measurement. Improvements by a factor 6 to 8 compared to the uncompensated load cell can be achieved. This is normally sufficient for making a simple temperature correction for commercial scales. For high-end scales or legal for trade scales we recommend to use an ordinary Rspan in combination with the here described Mult_TkG and Mult_TkO method.
In application note AN021 we describe the method of the temperature compensation by the means of 2 resistors (Rspan_by_temp) and the scope of its use in detail.

3.6.3 Annotations Rspan

PICOSTRAIN needs only one Rspan resistor. As the common mode rejection ratio (CMRR) of the PICOSTRAIN products is nearly infinite (up to 135dB) there is no need to use two Rspan resistors. Indeed, PICOSTRAIN can not handle bridges with two Rspan resistors.

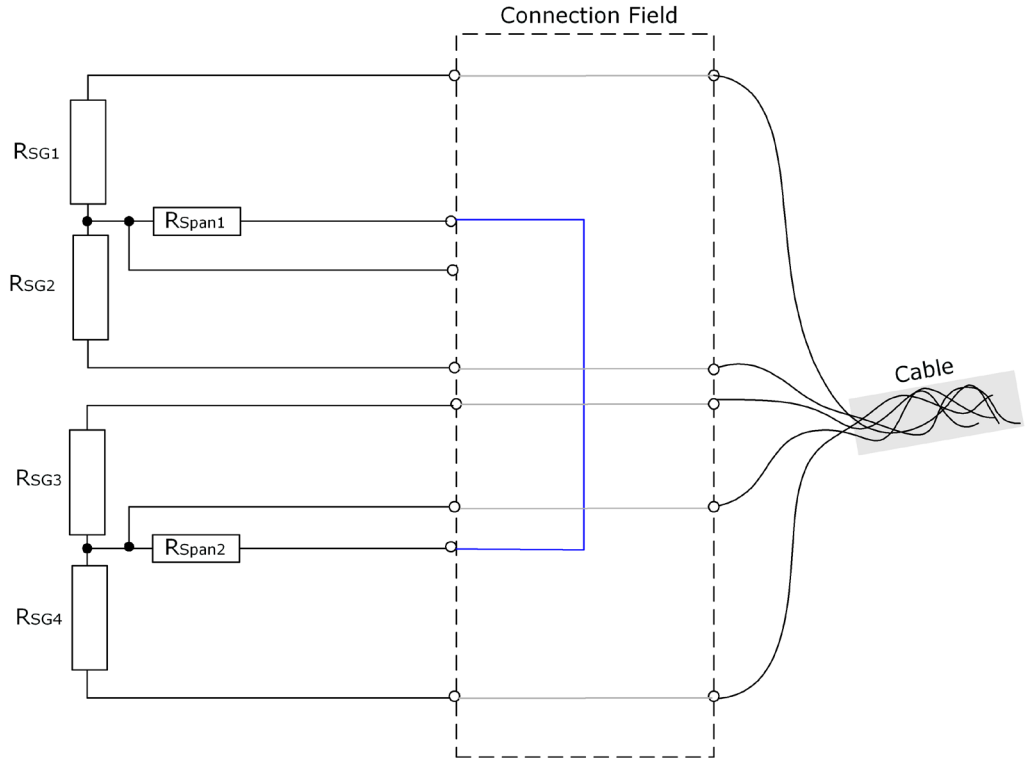
The easiest is to use load cells which natively have only 1 Rspan resistor. Nevertheless, if you want to make a first tests with a load cell which has 2 Rspan you simply can shortcut one of them in the following manner:

Figur 3.27 Shortcut Rspan



A more convenient way to change a load cell with 2 Rspan is to switch them in series. This is possible if the connections of the Rspan resistors are available as illustrated in the following picture:

Figure 3.28 Two Rspan in series



Please note: You are not touching the strain gage sensors or the Rspan resistors directly. Instead you make any re-wiring proposed in the connection field of the load cell. This is true for the changes regarding Rspan as well as the change from Wheatstone-wiring to PICOSTRAIN-wiring (please see also chapter 3.3.2).

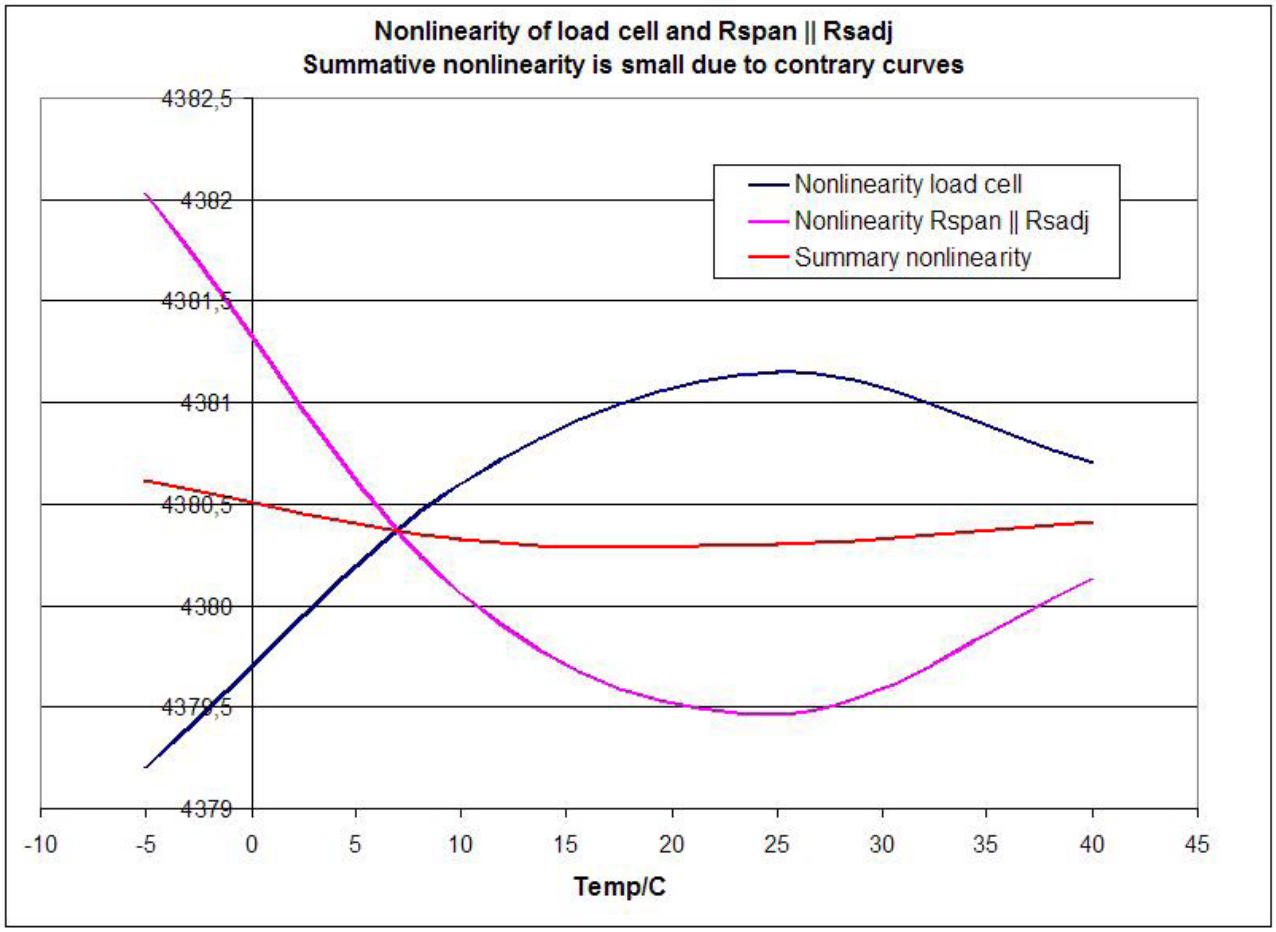
3.6.4 Nonlinearity of gain drift over temperature

Scope of this item: Only important for calibrated scales, e.g. according to OIML specification.

Independently of the PICOSTRAIN gain drift compensation we have always a nonlinearity of the load cell over temperature. This nonlinearity generally has two causes, the nonlinearity of the load cell itself (material, glue, wiring, etc.) and a nonlinearity coming from the paralleling of the Rspan resistor with its adjustment resistor (Rsadj). Normally these two effects are in opposite direction, so that overall nonlinearity can be reduced. In other words, the nonlinearity introduced by the paralleling of the resistors is compensating to some degree the nonlinearity coming from the load cell itself.

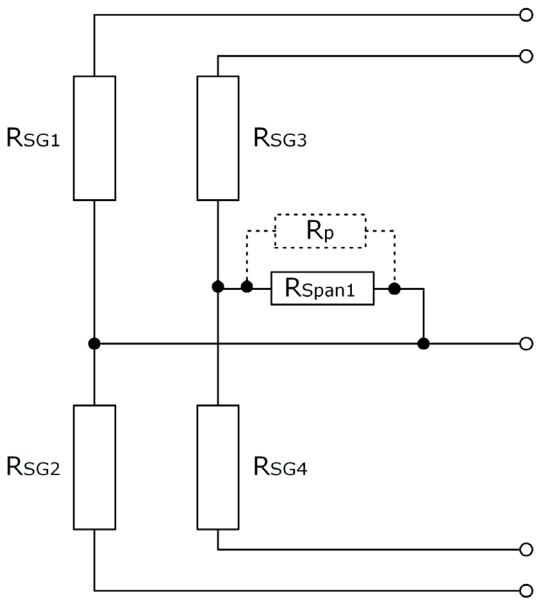
Please see the following diagram illustrating the effect:

Figure 3.29 Non-linearity



However there is a change in behavior if the adjustment resistor (Rsadj) is missing at all. There is no longer an effect of compensating the nonlinearity In the basic set-up of a PICOSTRAIN bridge with 1 Rspan there is no further adjustment resistor needed and therefore missing. This is not a problem if the load cell's nonlinearity itself is very low. But if it has a nonlinearity not neglectable, it may be necessary to add a parallel resistor to compensate for the nonlinearity. Please note that the purpose of this parallel resistor (Rp) is compensation of the non-linearity but not correcting Rspan in its resistor's value.

Figure 3.30 Rp placement



There are several criteria to decide whether a parallel resistor (Rp) should be used or not. Please find a detailed description in our dedicated Whitepaper WP002. Recommendations of whether the use of Rp is recommended or not and the size of it is automatically calculated by an Excel-Sheet available from acam. If it is recommended to add an Rp resistor it can be an ordinary fixed resistor with no special requirements.

3.6.5 Gain-Drift of PSØ81 itself – Optimization with Mult_PP

The PSØ81 has a very low gain drift of ~ 1ppm/K in case the Mult_PP factor is set properly. The reason for this gain drift is different than in an A/D-Converter. Because of this, we give some background information in this section to understand the cause of the gain drift of PSØ81 and also we give some hints how to measure it properly.

Background: In a classical A/D converter application the temperature drift of the resistors of the operational amplifier have to match very exactly. A mismatch is seen as gain drift. In PSØ81 the physical reasons are totally different. PSØ81 has a TD-Converter with no preamplifier.
The gain drift of the PS081 electronics comes mainly from the delay time of the comparator. In Wheatstone mode the analog multiplexer is part of this delay time and therefore the selected type of the analog multiplexer affects this value. When setting pptemp = 1 in register 2 the PS081 can measure the delay by means of the two resistors at ports PSEP1 and PSEP2. The determined correction factor can be adjusted by factor Mult_PP

The delay depends on the value of the low-pass capacitor which is parallel to the collector resistor of the comparator transistor. We recommend 3.3 kOhm for the collector resistor and 3.3 nF for the parallel capacitor. If this capacitor is changed, the optimum Mult_PP factor changes, too. This is the main dependency of Mult_PP from the hardware, besides the general selection between external and internal comparator.

Optimizing Gain Drift with Mult_PP
With our hardware recommendations the system has a remaining gain error of approximately 8 ppm/K if Mult_PP is set to 1.0 or not used. This remaining gain error can be reduced to < 1 ppm by choosing the right Mult_PP factor.

Once established during the development phase, this value can be used for the whole series production. It is definitely not necessary to adjust every single electronic.

Note:
The gain drift of the PS081 itself is very close to zero. The remaining gain drift comes mainly from the nonlinear part of the delay time of the comparator. This nonlinear part itself results mainly from the low-pass filter behind the first stage of the comparator (3.3 kOhm || 3.3 nF). It can be significantly reduced by the Mult_PP value. Because the low pass filter can be reproduced very accurately also the compensation is very stable over production and needs no adjustment.

With the recommended hardware we determined following Mult_PP factors:

- Wheatstone bridge with external comparator and analog mux TS5A3160 (TI) : 1.28
- Wheatstone bridge with external comparator and analog mux 74LVC1G3157(TI) : 1.14
- PICOSTRAIN standard wiring, external comparator, our recommended values: 1.28

For Wheatstone Mode we recommend TS5A3160 because it has a good behavior also at lower supply voltages < 2.7 V, but 74LVC1G3157 is a good choice, too, for supply voltages of 3.0 V or higher. If own hardware settings are used, especially if the low pass filter is changed, the MULT_PP factor may change and has to be determined. The value for this hardware setting is valid for the whole production and is independent from production batches. For more details see application note AN018.

3.6.6 Zero Drift of PS081 itself

Also the zero drift of the PS081 originates from a reason other than the drift of an AD-Converter. The reason of the remaining zero drift of our PICOSTRAIN products are parasitic resistor paths that

are not or not perfectly compensated. Mainly in the packaged version of PS081 differences in the bond wires are not compensated.

Because of the nature of the remaining zero drift, the value of this drift depends on the value of the strain gage resistor. The lower the strain gage resistor the higher is the remaining drift. E.g. with a 1 kOhm strain gage the zero drift is approximately 1/3 of the drift of a 350 Ohm strain gage with the same chip.
For best offset drift behavior we recommend the standard full bridge connection. The systematic offset drift in this mode is approx. ±10 nV/V/K and lies therefore in the 50% limit of OIML 10000. It is possible to use PS081 in high end scales, please see also Application Note AN018 for further details.

In all PICOSTRAIN modes the sensor wire resistance is part of the zero drift. To minimize the drift please have a close look on the length of these wires to the load cell. The most critical part is normally the PCB, a few millimeters of missmatch can be well seen in the offset drift. The cable to the load cell is not as critical because the wires have a much bigger diameter.

A special case is Wheatstone mode. In this mode nearly 100% of the remaining parasitic resistances are compensated because of the kind of the measurement. Therefore, in Wheatstone mode the zero drift of PS081 is close to zero and can be improved to < 1 nV/V/K also if the wires are not matched.

For comparison:

- To comply with OIML 3000 specifications the zero drift of the complete scale must not exceed 133 nV/V/K
- To comply with OIML 10000 specifications the zero drift of the complete scale must not exceed 40 nV/V/K

Following table gives an overview of the typical offset drift of PS081. To get a good idea of the min./max. values multiply the typical values by the factor of 3. You get a good estimation over the distribution of a production lot (not a guarantee).

Typical drift in different modes

Mode	350 Ohm SG	1 kOhm SG	OIML 10000
Fullbridge Standard *	±10 nV/V/K	±4 nV/V/K	±40 nV/V/K
Wheatstone	<±1 nV/V/K	<<±1 nV/V/K	±40 nV/V/K

*with cross-matched traces on the PCB, please refer to below explanation.

Zero Drift of Full Bridge Standard Wiring

In the packaged version of PS081 there is some systematic offset drift due to different bond wire lengths. This zero drift is ±10 nV/V/K and the same for every chip. Because of the nature of this drift it can be easily compensated on the PCB by cross-matching the different bond wire length on the PCB.

To compensate this systematical drift the trace from SG-B1 to the connection pad of the load cell wire should be (t.b.d.) mm longer than the trace from SG-A2 to the connection pad. SG-A1 and SG-B2 should have the same length.

Please note: In any case, the main source of the drift is not the electronic. PICOSTRAIN and also a good AD-converter have 5 to 10 times better drift values than a good load cell. Therefore, if the system performance should be significantly increased, the drift of the load cell has to be reduced. PS081 can solve this task by software with the Mult_TKO possibility.

3.6.7 Mult_UB - Power Supply Rejection

PS081 measures frequently the supply voltage. The measured voltage can be used to correct the dependency of the gain from the voltage. It is switched on by configuration bit mult_en_ub = 1. Factor mult_ub[7:0] defines the control ratio of the voltage measurement. The control ratio is generally very low. The result of the strain measurement will be corrected according to

$HB = HB/(1 + UB*[-128 \ldots 127] / 2^{21})$.

The standard setting for Mult_UB is 0xF7.

Table of Contents Page

4 Peripheral Components & Special Settings

4.1 Oscillators

PSØ81 has an internal low-current 10 kHz oscillator which is used for basic timer functions and for the definition of the cycle time in stretched modes and measuring range 1.
Further, PSØ81 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 non-stretched modes. It needs about 130 µA @ 3.0 V.

Configuration: Register 3, Bits 17 to 19: sel_start_osz
0 = Switch off oscillator
1 = oscillator continuously on
2 = Measurement started with 100 µs delay after switching on the oscillator
3 = Measurement started with 200 µs delay after switching on the oscillator
4 = Measurement started with 300 µs delay after switching on the oscillator
5 = Measurement started with 400 µs delay after switching on the oscillator 6 & 7 are not connected
Register 2, Bit 0: auto10k

This oscillator can be switched on continuously or only for the duration of the measurement, including some lead time to reach the full oscillation amplitude (sel_start_osz[2:0]). The startup time for the 4MHz oscillator is about 50 µs to 100 µs and slightly depends on the supply voltage.

Auto-calibration:
The internal 10 kHz oscillator may be automatically calibrated by means of the 4 MHz oscillator. The frequency varies with temperature and voltage. This would impact the update rate and sampling rate as the 10 kHz is the basis for the TDC conversion counter and in stretched mode also the cycle time. It is recommend to use the auto-calibration option setting auto10k = 1.

Note:
Auto-calibration should not be used in stretched single conversion modes

Layout Considerations:
The oscillator should be placed close to the PSØ81. The area arround the oscillator should be flooded by a ground plane. The SPI wires should not cross the oscillator lines.
The 4 MHz should have an additional 1 MOhm pull-down resistor to avoid cross currents during switch-off. The resistor reduces the oscillator current and is urgently needed in solar applications.

4.2 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 including 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 efficient 2 V operation without voltage doubling
- Operation at un-stabilized supply voltage like lithium batteries or solar cells
- Currentless stand-by
- Driver strength adjustable to segment size and current consumption
- Outputs configurable so that existing displays can be connected
- Implemented conversion tables for 7 segment digits
- Implemented ROM code for 24 Bit number conversions

4.2.1 Basic Configuration

With lcd_duty the LCD is switched on 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 flicker-free 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

4.2.2 LCD-Power supply

The PSØ81 has an integrated charge pump to double and stabilize the voltage for driving 3 V and 2.5 V LCD displays. 2 V displays need no voltage doubling. The choice for the external capacitors depends on the size or capacitance of the display.

Configuration: Register 11: lcd_standby, lcd_vlt
Register 16: en_noise_lcd, lcd_direct_drive, use_10kHz_lcd
Register 17: lcd_pulsed, c10_div_lcd

lcd_standby 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.
= 0 LDC on
= 1 Standby

lcd_vlt[1:0] Selection of LCD supply voltage
= 0 2.0 V
= 1 2.5 V
= 2 3.0 V
= 3 2.0 V without voltage doubling

lcd_directdrive LCD direct drive selection
= 0 off
= 1 LCD is driven directly from Vcc without regulation and charge pump. This reduces the LCD current and should be used in solar applications.

lcd_dis_chargm 1 = Suppresses the recharging of the charge pump capacitors during a measurement. This option is reasonable only in stretched modes (here it is recommended) and single conversion mode (at low internal averaging (avrate < 10), the time between measurements must be sufficient to recharge the capacitors, duty 1:3).
This option works only in combination with a pulsed 4 MHz oscillator

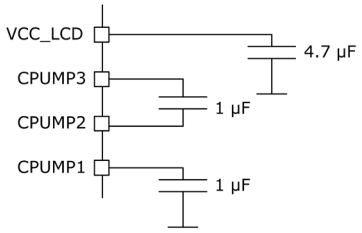
use_10kHz_lcd 1 = Select the low power 10 kHz oscillator for the LCD charge pump.
1 = recommended
0 = Charge pump clock derived from 4 MHz, divider can be set in the range 480 to 512. The selection avoids a fixed relation between charge pump and measurement and therefore reduced distortions of the measurement.

c10_div_lcd[5:0] 5 bit divider factor for deriving the 10kHz from the 4 MHz in continuous mode.

	divider factor = 448 + c10_div_lcd		
lcd_pulsed[3:0]	Automatically pulsed display, especially for stand-by in solar applications		
	Defines the period between the LCD switch-on as well as the on-time.		
	00xx	300 ms	blinking period
	01xx	600 ms	blinking period
	10xx	1 s	blinking period
	11xx	2 s	blinking period
	xx00	off	on-time
	xx01	100 ms	on-time
	xx10	200 ms	on-time
	xx11	400 ms	on-time
en_noise_C10_lcd	= 0	off	
	= 1	Adding noise to the divider factor specified in c10_div_lcd.	

4.2.2.1 3 V and 2.5 V operation with voltage doubling

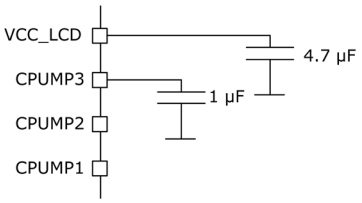
Figure 4.1 Charge Pump with voltage doubling



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 the LCD voltage.

4.2.2.2 2 V operation without voltage doubling

Figure 4.2 Charge Pump without voltage doubling



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

4.2.2.3 Direct Drive

A third option is to drive the LCD directly from the power supply without regulation. Therefore no external capacitors are needed and the output drivers are set low resistive.

lcd_r_const = 0 (10 kOhm).

lcd_vlt[1:0] = 0 2.0 V

Nonetheless, it might be helpful to have 1 µF capacitors at CPUMP1 and CPUMP2 for better noise reduction, as shown in figure 4.2.

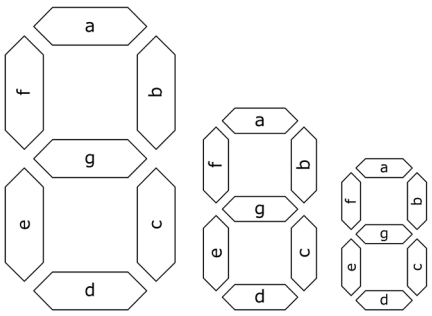
In Direct Drive mode it is recommended to use an LCD frequency as low as possible, e.g. lcd_freq = 1. The direct drive mode is generally the recommended operation mode for driving an LCD with the internal LCD-driver.

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

Figure 4.3 LCD *Segmentsize*



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	1	0 = charge capacitors by 200 Ohm resistors 1 = charge capacitors by 1 kOhm resistors not relevant in direct drive mode

lcd_r_const[1:0]	1	2	3	Defines the cross resistance of the LCD voltage divider 0 = 30 k 1 = 50 k 2 = 200 k 3 = 800 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
		0	0	0 Zero is mandatory in stretched modes, not relevant in direct drive mode
lcd_r_fastld		3	2	1 Configures the number of fast-load periods(10ms) with low-resistance voltage divider

4.2.2.5 Crosstalk LCD Charge Pump - Measurement

The LCD charge pump might have a negative impact on the measurement quality due to crosstalk from the charge pump. Typically, this becomes obvious by periodic oscillations of the measurement result.

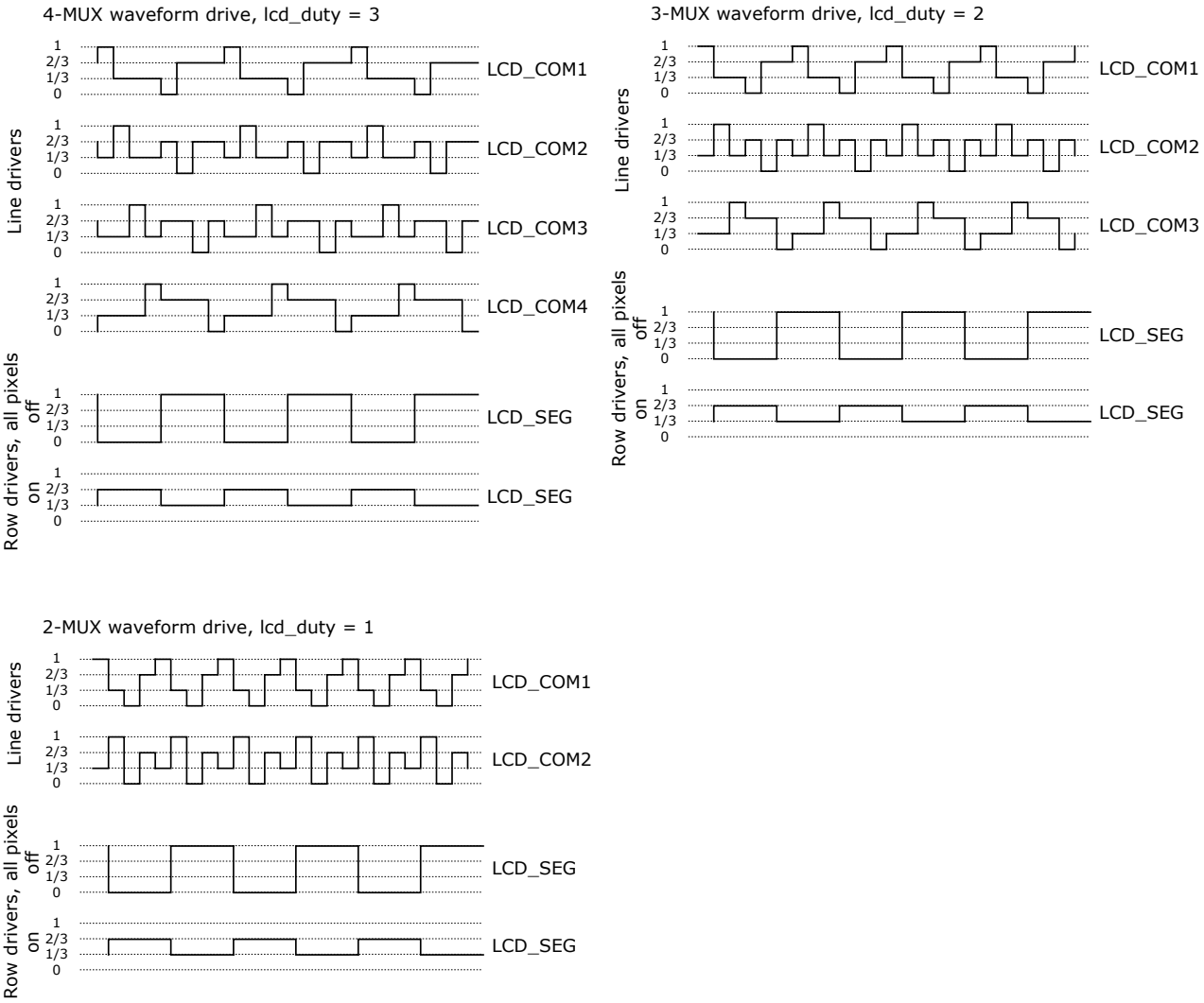
PSØ81 offers several configuration options to overcome this. In the following we show some measures. There is not a unique solution but a need to select the right one for the application. In any case of uncertainty please contact acam for support.

- In Direct Drive Mode the LCD does not affect the measurement.
- Selecting the right LCD frequency in many cases solves the problem. In general, the higher the frequency the lower the distortion. Testing the different lcd_freq values will show the best choice. The current consumption might be a little bit higher. In battery driven system this will not be a problem. In solar applications we recommend direct drive, anyway.
- LCD voltage divider resistors should be selected as high as possible. This might be in conflict to measure 2, and segments to be off can be seen. The right choice has to be proven by experiment.
- The blocking capacitor at VCC_LCD can be increased. As a standard we recommend 4.7 µF. In case of problems an increase up to 22 µF shows obvious improvements.

4.2.3 LCD driving methods

In each mode the outputs drive 4 voltage levels, 0, 1/3, 2/3 and full LCD voltage.

Figure 4.4 LCD driving



4.2.4 LCD Control

A digit is made of 8 segments. Each segment is named by a character 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 assignment does not depend from the multiplex mode. It looks like the following:

Figure 4.5 Segemnts

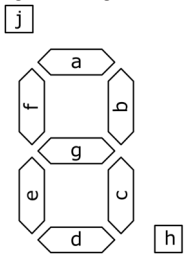


Table 4.1

Digit	Segment		Hex Value	Position in the Configuration Memory:
	hgfe	dcba		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.
“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	

Table 4.2

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 opcodes no2lcd, no2lcdAccu. 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 ignores the upper 2 digits in 2x multiplex and the upper digit in 3x multiplex. The special characters in 3x and 4x multiplex will not be changed (lcd_segment[55:48]). In 2x multiplex the comma of the display might be used for special characters. In this case they must be restored after the conversion.

Note:

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.

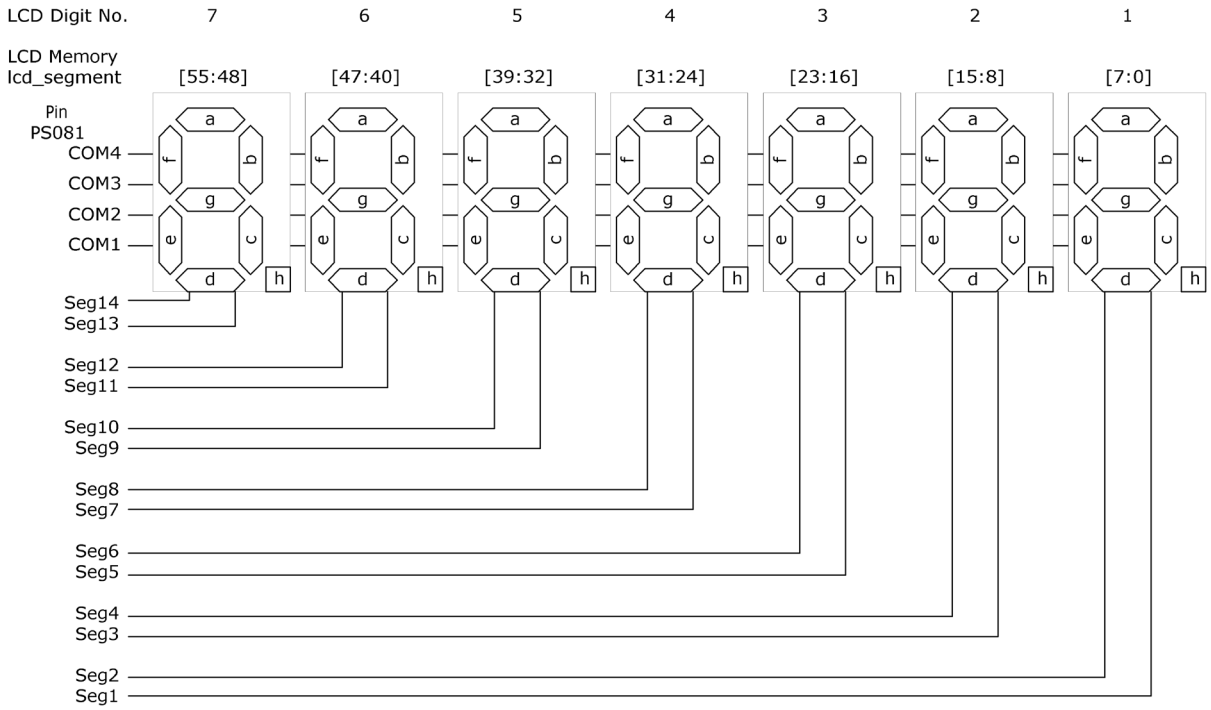
Code snippet:

```
ramadr 20           ; HBO result
move   x , r        ; Load x-accumulator with the result
move   y , 2        ; Load y-accumulator with the comma position
no2lcd              ; Convert into 7-Segment display format
newlcd              ; Update LCD
clrwdt              ; Set back the watchdog
stop                ; Stop the µC
```

4.2.5 Connecting Schemes

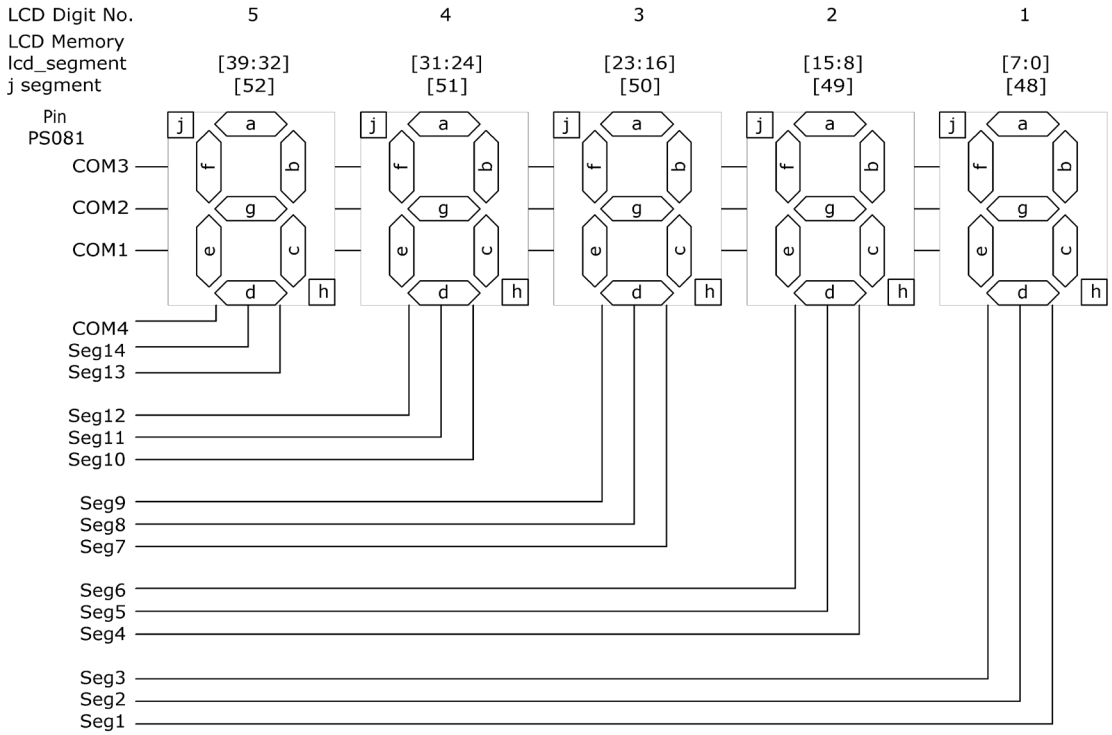
4.2.5.1 4-MUX (¼ duty)

Figure 4.6 4-MUX *(¼ duty)*



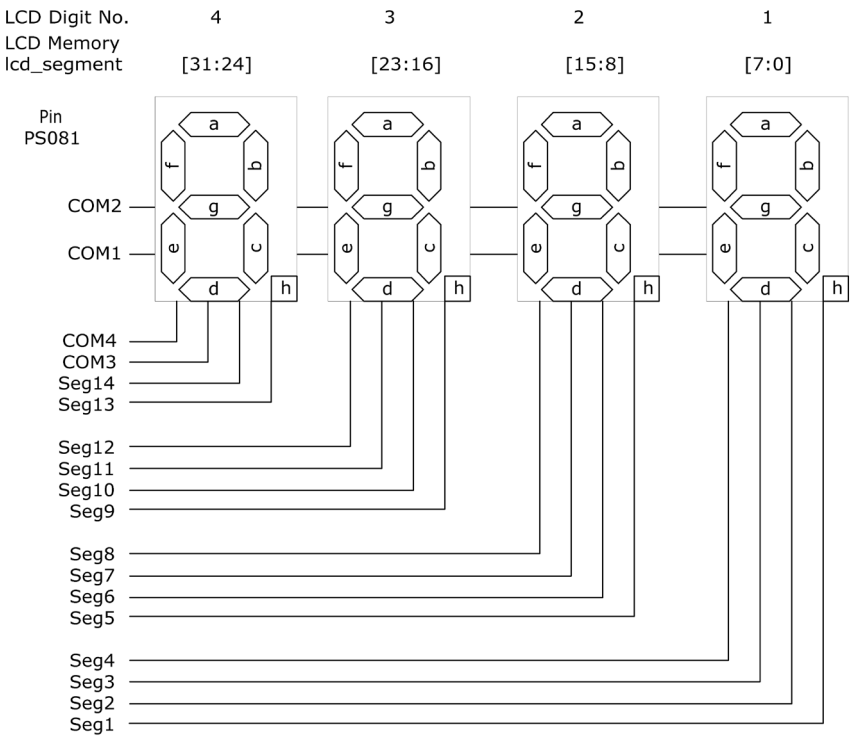
4.2.5.2 3-MUX (1/3 duty)

Figure 4.7 *3-MUX (1/3 duty)*



4.2.5.3 2-MUX (½ duty)

Figure 4.8 *2-MUX (½ duty)*



4.2.6 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 LCD's.

Limitations:

The segment lines and eventually common lines 3,4 have to be connected to the right digit of the display. The order within one digit is free. Otherwise the command no2lcd will mix up the digits.

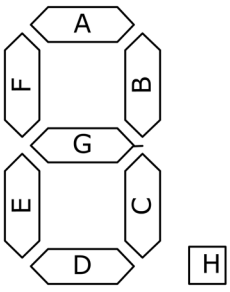
In register lcd_segment the program sets the segments to be displayed. lcd_pos defines which bit in lcd_segment refers to the one out of the 8 target segments.

Example: 4xMux Display

Defaut Wiring for the PSØ81:

	Last digit to the left							**Last digit to the right**	
	Seg 14	Seg 13	Seg 12	Seg 11	… …	… …	… …	Seg 2	Seg 1
Com1	lcd_pos [14...12]	lcd_pos [2...0]	lcd_pos [14...12]	lcd_pos [2...0]	…	…	…	lcd_pos [14...12]	lcd_pos [2...0]
Com2	lcd_pos [17...15]	lcd_pos [5...3]	lcd_pos [17...15]	lcd_pos [5...3]	…	…	…	lcd_pos [17...15]	lcd_pos [5...3]
Com3	lcd_pos [20...18]	lcd_pos [8...6]	lcd_pos [20...18]	lcd_pos [8...6]	…	…	…	lcd_pos [20...18]	lcd_pos [8...6]
Com4	lcd_pos [23...21]	lcd_pos [11...9]	lcd_pos [23...21]	lcd_pos [11...9]	…	…	…	lcd_pos [23...21]	lcd_pos [11...9]

Figure 4.9 4mux-digit.gif



for segment	**lcd_pos[]=**
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	7
i	8

Example 1:

LCD specification

	Seg 2,4,6...	Seg 1,3,5...
Com1	e	a
Com2	f	b
Com3	g	c
Com4	h	d

then

lcd_pos values

Seg 2,4,6...	Seg 1,3,5...
lcd_pos [14...12] = 4	lcd_pos [2...0] = 0
lcd_pos [17...15] = 5	lcd_pos [5...3] = 1
lcd_pos [20...18] = 6	lcd_pos [8...6] = 2
lcd_pos [23...21] = 7	lcd_pos [11...9] = 3

lcd_pos[23...0] = 111110101100011010001000

= 0xFAC688

Example 2:

LCD specification

	Seg 2,4,6...	Seg 1,3,5...
Com1	g	a
Com2	b	f
Com3	e	e
Com4	h	d

then

lcd_pos values

Seg 2,4,6...	Seg 1,3,5...
lcd_pos [14...12] = 6	lcd_pos [2...0] = 0
lcd_pos [17...15] = 1	lcd_pos [5...3] = 5
lcd_pos [20...18] = 2	lcd_pos [8...6] = 4
lcd_pos [23...21] = 7	lcd_pos [11...9] = 3

lcd_pos[23...0] = 111010001110011100101000

= 0xE8E728

Example: 3xMux Display

Default wiring for PSØ81:

	Last digit to the left						Last digit to the right		
	Com 4	Seg 14	Seg 13	... ...	... ...	... ...	Seg 3	Seg 2	Seg 1
Com1	lcd_pos [20...18]	lcd_pos [11...9]	lcd_pos [2...0]	...	...	...	lcd_pos [20...18]	lcd_pos [11...9]	lcd_pos [2...0]
Com2	lcd_pos [23...21]	lcd_pos [14...12]	lcd_pos [5...3]	...	...	...	lcd_pos [23...21]	lcd_pos [14...12]	lcd_pos [5...3]
Com3	**	lcd_pos [17...15]	lcd_pos [8...6]	...	...	...	**	lcd_pos [17...15]	lcd_pos [8...6]

Example1:

LCD specification

	Seg 3,6,9...	Seg 2,5,8...	Seg 1,4,7...
Com1	g	d	a
Com2	h	e	b
Com3	i	f	c

then

lcd_pos values

Seg 3,6,9...	Seg 2,5,8...	Seg 1,4,7...
lcd_pos [20...18] = 6	lcd_pos [11...9] = 3	lcd_pos [2...0] = 0
lcd_pos [23...21] = 7	lcd_pos [14...12] = 4	lcd_pos [5...3] = 1
**	lcd_pos [17...15] = 5	lcd_pos [8...6] = 2

lcd_pos[23...0] = 111110101100011010001000

= 0xFAC688

** Special symbols are fixed by LCD designand can not be rearranged. It has to be take care that those segments are connected to Com3 and Seg3,6,9... Lines

Example2:

		Seg 3,6,9...	Seg 2,5,8...	Seg 1,4,7...
if	Com1	h	e	d
	Com2	f	g	c
	Com3	i	a	b

then

Seg 3,6,9...	Seg 2,5,8...	Seg 1,4,7...
lcd_pos [20...18] = 7	lcd_pos [11...9] = 4	lcd_pos [2...0] = 3
lcd_pos [23...21] = 5	lcd_pos [14...12] = 6	lcd_pos [5...3] = 2
**	lcd_pos [17...15] = 0	lcd_pos [8...6] = 1

lcd_pos[23...0] = 101111000110100001010011
= 0xBC6853

** Special symbols are fixed by LCD design and can not be rearranged. It has to be taken care that those segments are connected to Com3 and Seg3,6,9... Lines

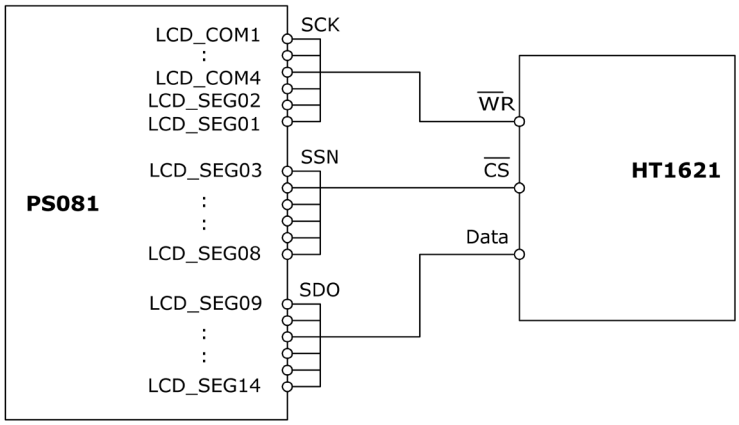
4.3 Support of an External LCD Driver

PSØ81 offers the possibility to control an external LCD driver. Therefore a simplified SPI master mode was implemented, especially adopted to Holtek HT1621 and FM µpd7225 devices.

For the SPI interface pins of the LCD driver are used. Up to 6 pins have to be used in parallel to have enough driver power.

Connecting an external LCD driver circuit:

Figure 4.10 External LCD Driver



Configuration:

lcd_spi_ena	1 =	Switch on SPI Master Mode on LCD pins
lcd_duty	2 =	It is necessary to set a duty mode to use the LCD as SPI master driver (1 = ½ duty, 2 = 1/3 duty, 3 = ¼ duty)
lcd_directdrive	1 =	LCD supplied directly by Vcc
lcd_standby	0 =	LCD voltage generation can be switched off.

lcd_spi_out[2:0] Set the LCD pins as SPI outputs, 6 pins in parallel each.

xx1 switch on LCD_COM1…LCD_COM4,LCD_SEG1,LCD_SEG2 as SCK

x1x switch on LCD_SEG3…LCD_SEG8 as SSN

1xx switch on LCD_SEG9…LCD_SEG14 as SDO

spi_delay[2:0] Sets the timing for the LCD SPI master interface

0 = fast (about 500 ns period)

:

7 = slow (about 10 µs period)

Programming:

After a no2lcd opcode the registers 61 and 62 hold the pixel data according to standard segments. The re-arrangement according to lcd_pos is done in the LCD driver. After opcode newlcd the re-arranged data can be read from RAM addresses:

reg_lcd_pix_sort61= 64+32+2 = 98

reg_lcd_pix_sort62= 64+32+3 = 99

It is reasonable to wait a few cycles after the new_lcd command.

There are new opcodes that support the LCD SPI master:

ssnPulse Generates a positive pulse on the “SSN” line (lines LCD_SEG3…LCD_SEG8)

ssnSet Sets “SSN” to HIGH.

spi2lcd This opcode sends the content of accumuator y to the SPI interface. The second parameter defines, how many bits are transmitted. This function is mainly needed when operating an external LCD driver via SPI.

Note:

- no2lcd opcode formats the number for output on the external LCD. It always has to be used in combination with newlcd opcode
- Maybe you need to adapt lcd_pos in register 12 according to your LCD
- Please be aware that accumulator x, y and z are used and modified by no2lcd and spi2lcd opcodes.
- There is no possibility to connect both – the internal and external LCD. Only one mode at a time can be driven.

Example:

```
;initialization
jsub      init_holtek
;measurement
ramadr 20         ;HB0
move              x, r
no2lcd            x, 1
newlcd
jsub      driver_4
;further routines
```

Please see code snippet in the appendix, explaining sub-routines 'driver_4' and 'init_holtek'. Basically, the holtek driver needs to be initialized and after the newlcd opcode the content is sent to the LCD by the sub-routine 'driver_4' (MUX4 LCD).

4.4 I/O-pins

PSØ81 has six I/O pins:

0 - SPI_DO_IO0	Serial data out (SDO) or multipurpose I/O
1 - SPI_DI_IO1	Serial data in (SDO) or multipurpose I/O
2 - SPI_CLK_IO2	Serial clock (SDO) or multipurpose I/O
3 - MULT_IO3	In Wheatstone application used for the analog multiplexer, interrupt. Otherwise, output or multipurpose I/O
4 - MULT_IO4	Multipurpose I/O
5 - MULT_IO5	Multipurpose I/O

The pins can be programmed as inputs or outputs with pull-up or pull-down resistors in case the chip is in stand-alone mode (SPI interface not used, SPI_ENA=0). Pin MULT_IO3 can be used as input/ output only when Wheatstone mode is not used. If none of the pins is configured as an output, the number of inputs can be increased up to 21 as described later on.
Additionally, Pin 24, SPI_CSN_RST can be used as reset input in case the SPI interface is not used (SPI_ENA = 0). The reset is high active.

4.4.1 Configuration

SPI_DO_IO0	Configreg_11, bit 16,17	io_en_0_sdo
SPI_DI_IO1	Configreg_11, bit 18,19	io_en_1_sdi
SPI_CLK_IO2	Configreg_11, bit 20,21	io_en_2_sck
MULT _IO3	Configreg_11, bit 22,23	io_en_3_mio
MULT _IO4	Configreg_17, bit 2,3	en_io4
MULT _IO5	Configreg_17, bit 4,5	en_io5

4.4.2 Port definition

00 = output

01 = input with pull-up

10 = input with pull-down

11 = input

MULT_I04 & 05 are set by default as inputs. MULT_I04 is set to 01 because in PSØ8 this pin is connected to Vcc. This way, the pin compatibility is kept.

4.4.3 Output – write

The outputs are set in configuration register 0 and register 17:

SPI_DO_I00	Configreg_0, bit 10	io_a[0]
SPI_DI_I01	Configreg_0, bit 11	io_a[1]
SPI_CLK_I02	Configreg_0, bit 12	io_a[2]
MULT _I03	Configreg_0, bit 13	io_a[3]
MULT _I04	Configreg_17, bit 6	io_a[0]
MULT _I05	Configreg_17, bit 7	io_a[1]

4.4.4 Increasing the number of Inputs

By means of external diodes, the maximum number of inputs is 21 , as long as no output is used. In dependency of how many pins you connect, the number of maximum inputs varies. The following table and illustration show the conection of the inputs. Please note, that in Wheatstone mode there is a natural reduction of 1 pin, as I/O3 is used to control an external switch in Wheatstone mode.

Table 4.3 Number of I/Os

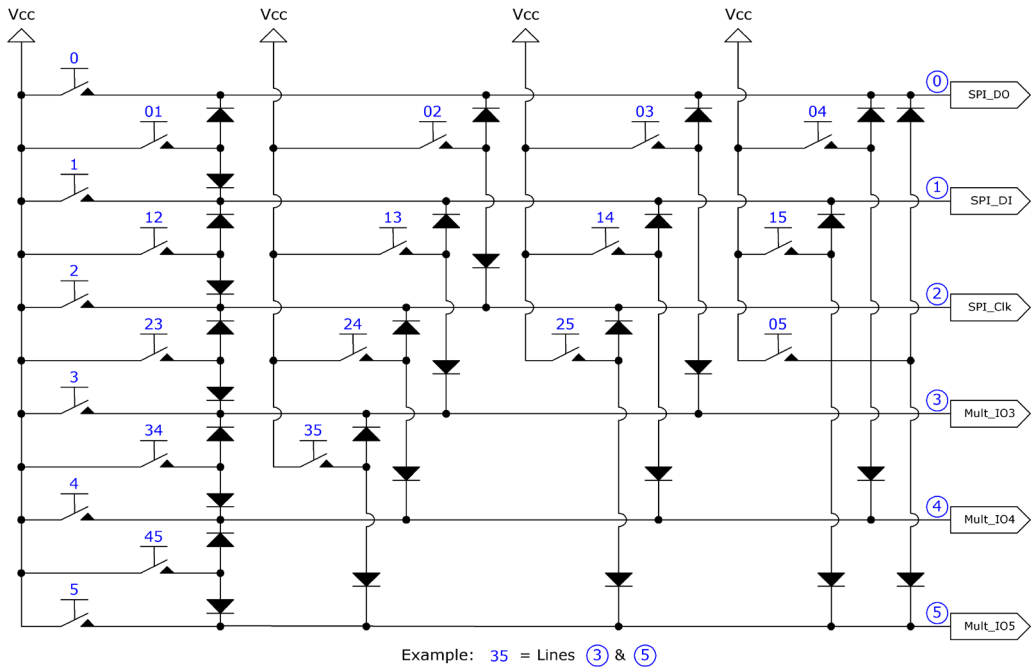
# of pins	# of inputs
6	21
5	15
4	10
3	6

The combination of 2 input lines gives another input. The 2 lines are connected over diodes electrically. Please see the follwing picture showing the maximum number of combinations (21). The possibility of increasing the number of inputs makes only sense if you want to have more than 4 inputs, otherwise you can use Mult_I0 0 to Mult I0 3 directly.

The recognition of synchronously pressed buttons is timing sensitive. Therefore the status register RAM address 26 to 28 have implemented the following features:

- Simultaneously pressed buttons are detected only if the single buttons haven't been pressed before.

Figure 4.11 Maximum Number of Inputs



- If a Simultaneous key is recognized, then the single button information is ignored.
- Only one pair of simultaneously pressed buttons can be detected at once. Further buttons will be ignored.
- More than one buttons can be pressed, but only subsequently, not at once.

Please note: If you connect several inputs to each pin according to the suggested method you must check the status of the inputs in register 26 to 28. You can NOT use register 22 in this case!

4.4.5 Input – read

4.4.5.1 4 inputs or less / inputs and outputs mixed

The flags of the inputs IO0 to IO3 are shown in status register, address 22. There is an indication of the rising edge, falling edge or whether the button was pressed:

Status information RAM address 22* (like in former PSØ8)

Status[23]= flg_io3_mio

Status[22]= flg_io2_sck

Status[21]= flg_io1_sdi

Status[20]= flg_io0_sdo

Status[07]= flg_io3_mio_r Rising edge

Status[06]= flg_io2_sck_r Rising edge

*) Important: Please use RAM registers 26 to 28 to look for the button status if you use more than 6 I/Os. If you have less than 6 I/Os you can check the status at RAM address 22.

Status[05]= flg_io1_sdi_r Rising edge
Status[04]= flg_io0_sdo_r Rising edge
Status[03]= flg_io3_mio_f Falling edge
Status[02]= flg_io2_sck_f Falling edge
Status[01]= flg_io1_sdi_f Falling edge
Status[00]= flg_io0_sdo_f Falling edge

4.4.5.2 More than 4 inputs, no outputs

The status of the inputs can be queried from the status registers at RAM address 26 to 28.

The status information is updated every time the program jumps into the EEPROM (bug fix PSØ8).
Status_F, address 26: Falling edges status
Status_R, address 27: Rising edges status
Status_P, address 28: Pressed edges status

Bit:	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input:	05	15	04	25	14	03	35	24	13	02	45	34	23	12	01	5	4	3	2	1	0

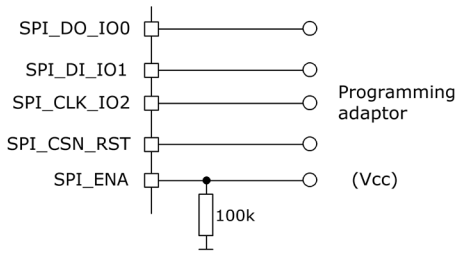
If two inputs are active at the same time, the PSØ81 handles this like an additional button. E.g., if buttons MULT_IO3 and MULT_IO4 are high at the same time, then bit [14] will be set in Status_R and Status_P. This allows the use of up to 21 buttons (see below).

4.5 SPI-Interface

4.5.1 Interfacing

The SPI interface is used to write the program, configuration and calibration data into the EEPROM. It can further be used to operate the PSØ81 as a 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 positive pulse on the CSN line before each opcode.

Figure 4.12 SPI interfacing



4.5.2 SPI Timing

Here we describe only the SPI timing for operation as a pure converter that communicates with an external microcontroller. PSØ81 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.

Figure 4.13 SPI timing

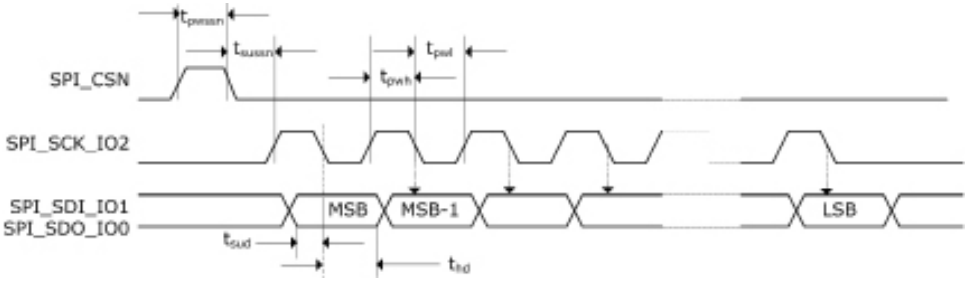


Table 4.4 SPI Timings

Time:	Description:	t_{min} [ns]
tpwssn	Pulse width SSN	500
tsussn	Setup time SSN / SCK	500
tpwh	Pulse width SCK high	500
tpwl	Pulse width SCK low	500
tsud	Setup time data	30
thd	Hold time data	30

tpwh and tpwl together define the clock frequency of the SPI interface. Consequently, 1µs corresponds to a clock rate of 1 MHz to run the SPI transmission. After sending a reset through the SPI, it is necessary to wait for 200 µs before sending the next opcode. If auto-configuration is on, it is necessary to wait for 1 ms. After writing to the RAM via SPI it is necessary to wait for 10 µs.

4.5.3 SPI - Instructions

4.5.3.1 RAM Access

RAM Write = b00000000 = h00

RAM Read = b01000000 = h40

New_LCD = b01000110 = h46

Power reset = b11110000 = hF0

Init reset = b11000000 = hC0

Start_new_cycle = b11001100 = hCC (continuous)

Start_TDC_cycle = b11001110 = hCE (single conversion)

watch_dog_off = b10011110 = h9E

watch_dog_on = b10011111 = h9F

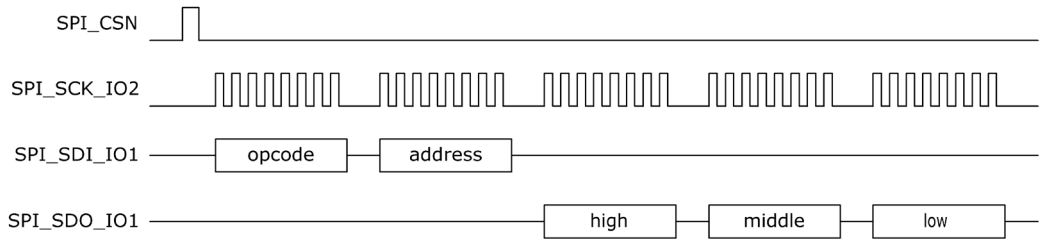
4.5.3.2 EEPROM Access:

EEprom_bgap_off	= b10000110	= h86	
EEprom_bgap_on	= b10000111	= h87	
EEprom_enable_off	= b10010000	= h90	
EEprom_enable_on	= b10010001	= h91	
EEprom_read	= b10100000	= hA0	(protected read)
EEprom_write	= b10100001	= hA1	
EEprom_bwrite	= b10100011	= hA3	(block write)
EEprom_berase	= b10100100	= hA4	(block erase)
userEEprom_read= b10100101		= hA5	address 2000-2047

It is necessary to switch on the bandgap and to enable the access before writing to or reading from the EEPROM.(send EEprom_bgap_on, EEprom_enable_on)

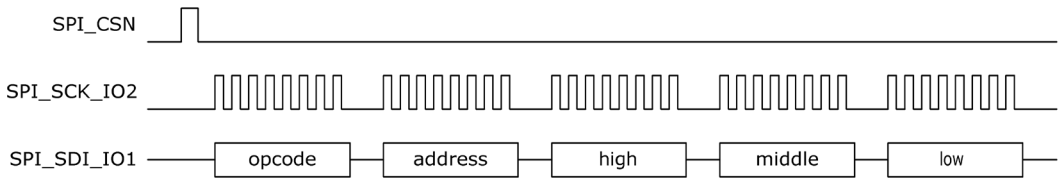
RAM Read Access

Figure 4.14 RAM read access



RAM Write Access

Figure 4.15 RAM write access

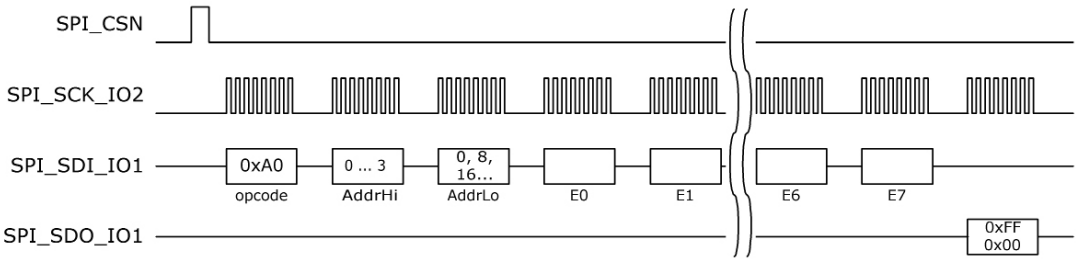


EEPROM Read Access / Read Protection

The PSØ81 EEPROM is protected against unauthorized reading. It is only possible to compare known data with the EEPROM content. When accessed through the interface, the EEPROM is addressed word-wise, thus giving 2 bytes (16 bits) per address. Reading from the EEPROM checks 8 bytes at once. The address therefore jumps by 4, the lower two bit should be always zero. The chip compares

the transmitted data with the EEPROM content. The result is available after 1 ms on the SDO port. =xFF stands for correct data, 0x00 for wrong data.
An unauthorized person has to test all possible combinations. This will last for 2^64 bits * 10 ns = 5849 years just for one read.

Figure 4.16 Read protection

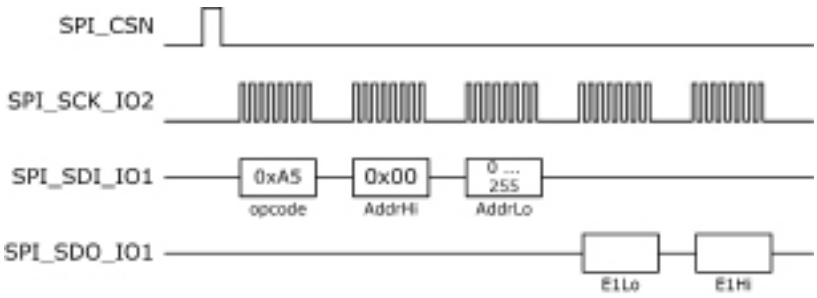


NOTE: E0, E2…E6 are the lower byte, E1, E3….E7 are the corresponding higher bytes in a word.

User-EEPROM Read Access

The upper 48 bytes of the EEPROM, address 2000 to 2047, are specified as userEEPROM. The user EEPROM is also accessed word-wise (16 bits) when addressed through the SPI interface. It is possible to read from these cells by means of opcode userEEPROM_read. userEEPROM_read reads back one 16 bit word.

Figure 4.17 User-EEPROM read access



EEPROM Write Access

The EEPROM is split into four blocks of 512 bytes or 256 words. The blocks are addressed by the lowest two bits of the first sent address byte. All write commands are followed by 16 bit data words with the lower 8 bits to be sent first. Programming is started by a sixth data word and stopped 4 ms later by a seventh data byte.
It is not possible to write a word into an EEPROM cell that is not empty. So it is not possible to fill up a word to OxFFFF (additional measure to protect from unauthorized reading).

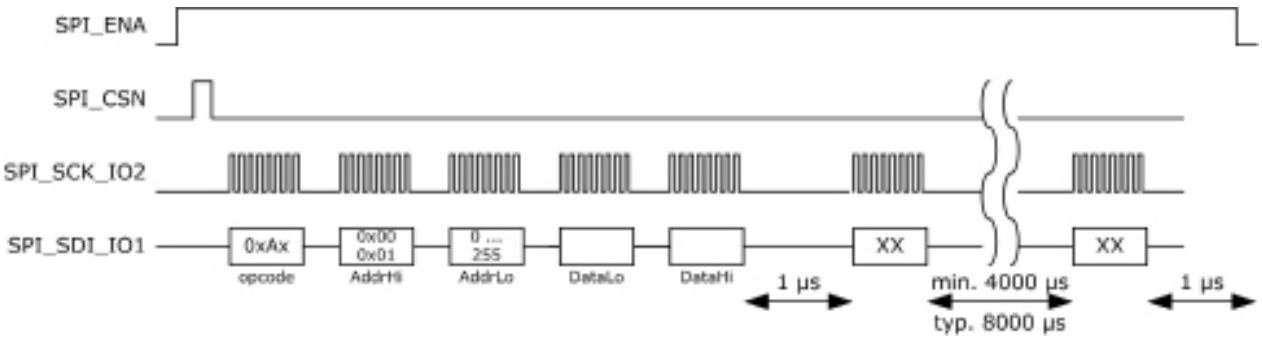


Figure 4.18 EEPROM write access

EEProm_berase:

Erase the complete block addressed in AddrHi. Looks the same as EEProm_write but the AddrLo and data bytes are ignored. For erasing the complete EEPROM this command has to be sent four times (AddrHi = 0, 1, 2 & 3).

4.5.4 Run PS081 with external µC via SPI (non-steady configuration)

Before starting: Please make sure the EEPROM is empty.

Sequence of commands to configrue the chip and start the measurement:

Power Reset (0xF0)

Watchdog off (0x9E)

Configure PS081 in RAM:

RAM Write (0x00) + add + 3x8Bit

Write RAM address 48..66

Important: all epr_xxx_xxx bits to 0 (configreg_1, bit 0-2)

Optional: Control read of RAM config

RAM Read (0x40) of RAM address 48..64

Configuration done.

Init Reset (0xC0)

Start New Cycle (0xCC)

Poll / Interrupt SPI_DO: New measurement value is indicated by SPI_DO 1 -> 0

Either poll it with µC or use interrupt pin of µC.

When SPI_DO goes from 1 to 0, toggle SPI_CS from 0 -> 1 -> 0

(this way SPI_DO is enabled for SPI communication):

Read HBO result at RAM address 0 (RAM Read, 0x40)

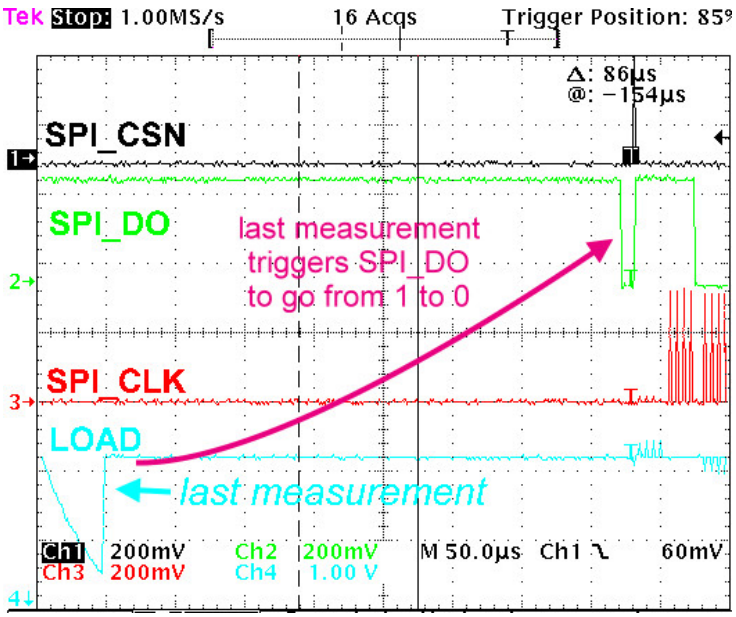
Annotations:

- You should make sure to erase the EEPROM before starting the sequence
- SPI_ENA must be HIGH. This enables the SPI-Slave mode and the SPI interface.
- When the measurement is started, new data is indicated by SPI_DO. Connect this wire to an input of your microcontroller and poll this pin. Alternatively connect it to an interrupt pin.
- When the interrupt is triggered please toggle the SPI_CSN pin in order to switch the SPI_DO wire from interrupt to communication mode (see pictures below)
- The hex-numbers in braces are the SPI opcodes. Please see chapter 4.5 SPI-Interface of the PS081 data sheet for an overview.
- There is also a way to store the configuration steadily in the EEPROM and only activate it by a power reset. Ask us for further information if you want to realize this way.
- Erasing the EEPROM, if required, can be done the following way:

Power reset (0xF0)
Erase EEPROM: Bgap on (0x87)
EEPROM enable on (0x91)
EEPROM block erase 0 (0xA4, AddHi = 0)
EEPROM block erase 1 (0xA4, AddHi = 1)
EEPROM block erase 2 (0xA4, AddHi = 2)
EEPROM block erase 3 (0xA4, AddHi = 3)
Bgap off (0x86)
EEPROM enable off (0x90)

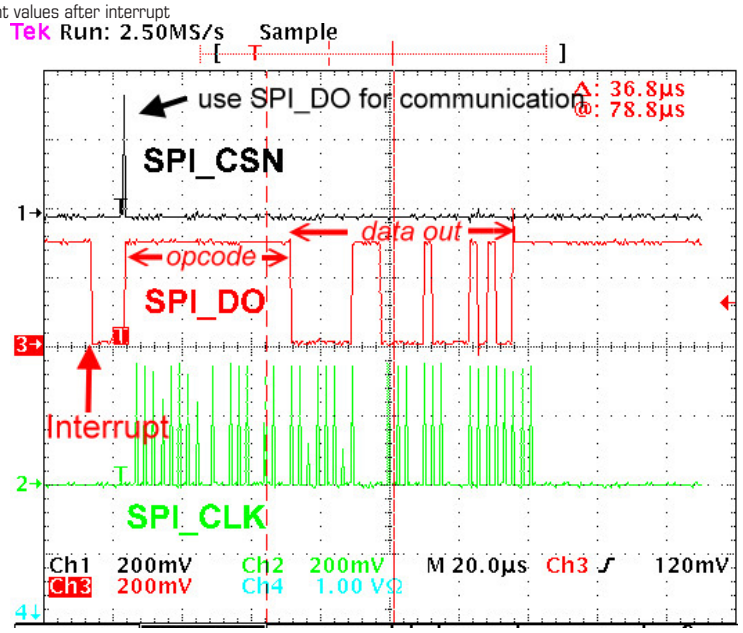
It is recommended to read the HBO result from RAM address 0. Reading from RAM address 20 (also HBO result) can result in a adress pointer conflict which is avoided when reading on address 0. Please note, that the HBO result is automatically copied to RAM address 0 as long as there is no program in the EEPROM (pure Front-End converter operation). **If you have an additional EEPROM program (e.g. pre-processing) you need to copy the HBO result from address 20 to address 0 manually!**

Fig. 4.19: Oscillograph of LOAD and SPI signals



In Fig. 19 you can see that SPI_DO gives an interrupt after the last discharging cycle of the measurement was done.

Fig. 4.20: Reading of measurement values after interrupt



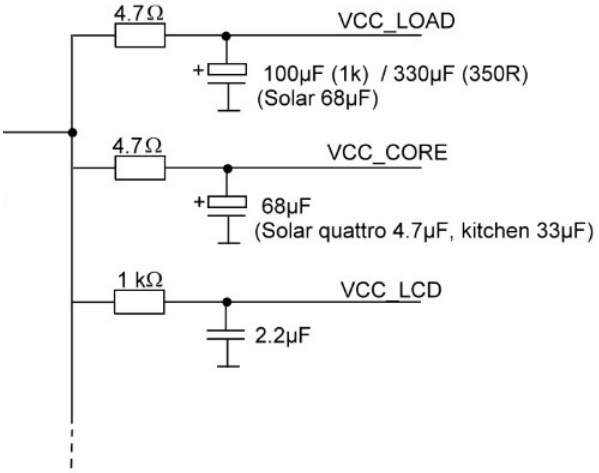
This oscillograph shows again the SPI_DO interrupt from 1 -> 0 and the response from the external microcontroller with SPI_CSN. This short pulse on SPI_CSN switches SPI_DO to communication mode. Then the opcode is sent to PS08 (not in the chart, this appears on line SPI_DI) and after 2 bytes the measurement results (3 bytes) is transmitted via SPI_DO (marked in the graph by ‚data out').

4.6 Power Supply

For a good measurement quality it is mandatory to follow some rules for the power supply.

There are several supply areas in the chip, VCC_LOAD, VCC_CORE and VCC_LCD. It is necessary to feed them with voltages decoupled by low-pass filters. Furhter, those pins need sufficient blocking capacitance mounted close to the chip.

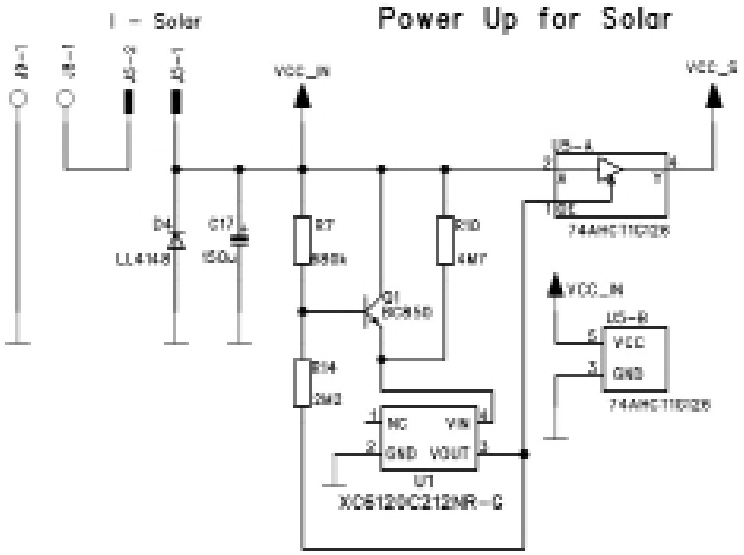
Figure 4.21 Power supply



In case of solar applications without any additional battery it is necessary to implement a power-up circuit. It provides a good start-up behaviour when the scale comes from total darkness. A solar panel delivers only a few microampere at poor light conditions and still has to start up the circuit.

The following figure shows a power-up circuit that is optimized for kitchen scales. To start up the scale it needs only about 3 µA @ 3.6 V. For other applications it might be necessary to change some values of the components.

Figure 4.22 Power-up Circuit



Short description of the function:
Coming from total darkness, all capacitors are discharged and the output of U5 is high-Z. The input voltage of U4 is zero. PS081 is not supplied by voltage. If light is switched on, the current from the solar panel charges C17 and supplies the voltage detection. The voltage detection (R7,R14,Q1,U3) is dimensioned so that U1 switches when the voltage at C17 passes 3.5 V. At that moment, the output of U5 leaves high-Z and goes to the voltage of C17. U4 is supplied with 3.5 V and regulates to 2.5 V for the PS081. PS081 begins to work. Because all capacitors behind VCC_R have now to be charged to the voltage at C17, this voltage drops down as only C17 can supply the necessary current. The solar panel is to weak for such a high current pulse. The voltage at C17 must not be lower than 2.55 V. Otherwise U4 cannot regulate 2.5 V for the PS081. C17 is also the buffer capacitor for low light situation. With the selected dimension under very bad light condition (20 Lux) the scale can operate for minimum 1 minute if it is well charged before. Therefore 1000 µF is the recommended value for C17 (minimum 680 µF).

Avoid to go below 680 µF. The circuit cannot start up if C17 is too low. The circuit will work with higher values, too, but the start-up time from darkness will increase because the charging time for C17 increases. The circuit is evaluated in detail and it is strongly recommended to follow the recommended values for a correct operation. For further information, see application note AN022 and design guide DG_Solar_POR.

4.6.1 Filtering / Recommendations LDO

In most circuits the voltage is regulated by a voltage regulator (LDO, low drop-out regulator). Of course this component has a noise which basically influences the measurement quality. Therefore it is crucial to choose suitable LDOs with a low-noise behavior, still keeping in mind that some applications need a low-current regulator as well. In this section we will give some recommendations which LDOs to choose.
The critical factor to watch out for is the ‚output noise'. The noise figures can normally be found in the datasheet of the LDO and is given as a summary value over the whole frequency range, e.g. 500µV RMS or in dependency of the frequency in µV / √Hz or as a diagram. A low output noise is desired, the figures can easily vary by factor 10 (e.g. Linear LT1761-BYP has 20µV RMS (with bypass capacitor) vs. TI TPS71501 which has 575µV RMS).

NOTE : We do NOT recommend switching regulators in any case, please use linear voltage regulators only as per the following recommendations.

Recommendations:

Table 4.4 LDO Recommendations

LDO	Features	Low-pass filter	Applications
Torex XC2206	medium noise, low current	use good low-pass filter	Solar
Micrel MC5205	medium noise, cheap solution	use medium low-pass filter	Low-cost solutions
Linear LT1761-BYP	very low noise, costly solution	use standard low-pass filter	High-end applications
TI TPS71501	very high noise	-	**NOT RECOMMENDED!!!**

Of course this list is far away from being complete. It shall just give some recommendations according to our experience in practical tests. Basically every low-noise, low-current LDO is suitable to use.

Caution: Please do NOT use the TI TPS71501 LDO as we saw major problems due to the noise of this regulator!

The additional low-pass filtering can help to reduce the noise getting through to the chip. Different types of low-pass filters can be used before decoupling the voltages:

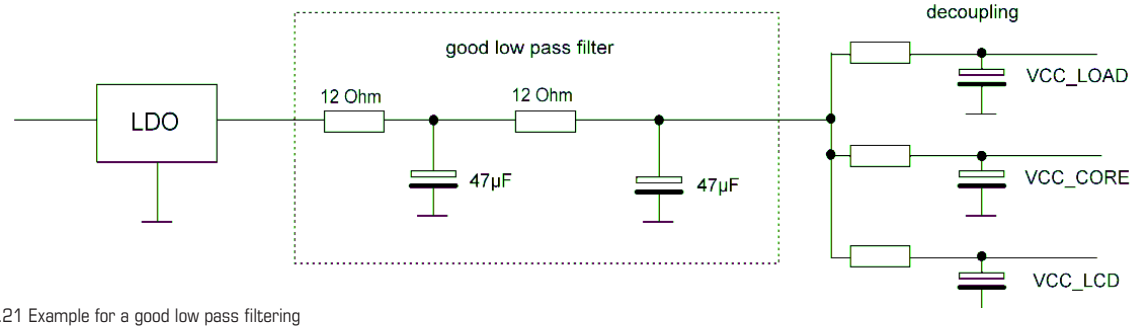


Figure 4.21 Example for a good low pass filtering

The good low-pass filter can reduce the noise coming from the LDO so that in combination with the decoupling of the voltages supplying the PS081 are widely noise-free or at least minimized. If the noise from the LDO is lower (like with the Linear or Micrel type i.e.) a simpler lowpass filter can be used, like the following:

Figure 4.22 Example for medium and standard low pass filtering

In the acam-circuits like those of the evaluation-kit or the examples for solar-quattro scales these insights are already put to practice. When building up your own circuit please make sure you follow the recommendations as they will contribute to a good overall measurement quality.

4.6.2 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 protects the EEPROM against corrupt data.

Caution: If the supply voltage goes below 2.1 V the voltage measurement will be wrong (the displayed value is too high) and the measured values cannot be used. In 1.5 V systems there is no possibility to measure the supply voltage with PSØ81.

Table of Contents Page

5 Configuration Registers

PSØ81 has 18 configuration registers of 24 Bit width, to be addressed in the RAM from address 48 to 66. The configuration registers control the whole chip including the strain measurement and the LCD controller. The configuration settings are mirrored in the EEPROM and the configuration is made by configuring EEPROM bytes 0 to 47 (48 bytes of configuration).

It is possible to write into the configuration registers

- By the internal microprocessor during operation
- Through the SPI interface from an external processor
- During the Power-on reset transferring a basic configuration from the EEPROM

5.1 Overview

Table 5.1 Overview of configuration registers

Configuration Register	RAM address	EEPROM bytes			PSØ8 vs. PSØ81 change
Configreg_00	48	2	1	0	no
Configreg_01	49	5	4	3	no
Configreg_02	50	8	7	6	no
Configreg_03	51	11	10	9	no
Configreg_04	52	14	13	12	no
Configreg_05	53	17	16	15	no
Configreg_06	54	20	19	18	no
Configreg_07	55	23	22	21	no
Configreg_08	56	26	25	24	no
Configreg_09	57	29	28	27	no
Configreg_10	58	32	31	30	no
Configreg_11	59	35	34	33	no
Configreg_12	60	38	37	36	no
Configreg_13	61	*not mirrored in EEPROM*			no
Configreg_14	62	*not mirrored in EEPROM*			no
Configreg_15	63	*not mirrored in EEPROM*			no
Configreg_16	64	41	40	39	YES
Configreg_17	65	44	43	42	NEW
Configreg_18	66	47	46	45	NEW

Internal microprocessor:

Configuration of registers is done in EEPROM and then mirrored to RAM. LCD segements are not configured in the EEPROM but written directly to the RAM address 61-63.

External microprocessor:

Don't use the EEPROM. Configure directly in the RAM addresses and set the paranetrs epr_pwr_cfg, epr_pwr_prg and epr_usr_prg to 0 (Configreg_01, Bits[2:0]).

5.2 Alphanumeric listing of configuration parameters

Table 5.2 Alphanumeric listing of configuration Parameters

Parameter	Register	Bits	Recommended Value
abgl_hr	1	19...22	5
auto10k	2	0	1
avrate	2	14...23	>1
bridge	3	0,1	
calcor	10	16...23	20
clkdivepr	16	14,15	1
clksyc	17	0,1	1
con_comp	11	0,1	
cpu_speed	0	1,2	2
cytime	2	4...13	
c10_div_lcd	17	11...15	
dis_haltpp_ps	0	0	0
dis_noise4	3	14	1
dis_osc_startup	0	3	1
dis_pp_cycle_mod	11	14	1
dis_startdel	11	15	0
en_avcal	1	9	0
en_fullbr_p	16	17	0
en_io4	17	2,3	2
en_io5	17	4,5	2
en_noise_c10_lcd	16	23	1
en_wheatstone	3	21	
ena_nonlin	16	16	0
epr_pwr_cfg	1	2	
epr_pwr_prg	1	1	
epr_usr_prg	1	0	
fha_en	16	11	0
force_unused_port	11	13	0
high_res	16	1	1
io_a	0	10...13	
io_a	17	6,7	
io_en_0_sdo	11	16,17	
io_en_1_sdi	11	18,19	
io_en_2_sck	11	20,21	
io_en_3_mio	11	22,23	
lcd_duty	1	15,16	

Parameter	Register	Bits	Recommended Value
lcd_charge	16	20,21	
lcd_directdrive	16	19	
lcd_dis_chargem	1	11	
lcd_fastld	11	8,9	1
lcd_freq	1	12...14	
lcd_pos	12	0...23	
lcd_pulsed	17	16...19	
lcd_r_const	11	6,7	2
lcd_r_fastld	11	3,4	1
lcd_segment	13	0...23	
lcd_segment	14	0...23	
lcd_segment	15	0...7	
lcd_spi_ena	16	22	0
lcd_spi_out	17	8...10	0
lcd_standby	11	12	
lcd_swload1k	11	5	
lcd_vlt	11	10,11	3
low_batt	1	3...5	
messb2	1	18	1
mfake	3	2,3	2
mod_caltdc	3	23	0
mod_rspan	1	6	
mult_en_pp	1	7	
mult_en_ub	1	10	
Mult_Hb1	4	0...23	
Mult_Hb2	5	0...23	
Mult_Hb3	6	0...23	
Mult_Hb4	7	0...23	
Mult_NLK	18	0...23	0
multio_sel	16	0...3	
Mult_PP	10	0...7	
Mult_TkG	8	0...23	
Mult_TkO	9	0...23	
Mult_Ub	10	8...15	
neg_sense	3	15	0
osz10khz_fsos	0	4...9	50
portpat	2	1	1
pptemp	2	3	1
ps_dis_phaseshift	3	11	0

Parameter	Register	Bits	Recommended Value
ps081adjust	3	4...9	33
ps_sel_calper	3	10	0
ps_speed	16	12,13	0
rspan_by_temp	1	8	
run_la_cont	16	4	0
sel_compint	11	2	
sel_compr	0	14,15	0
sel_qha	16	7...9	2
sel_speed_mh	16	5	0
sel_start_osz	3	17...19	
sense_discharge	16	10	1
single_conversion	2	2	
speed_talu	1	23	0
spi_delay	17	21...23	3
stop_osz	3	20	0
stretch	3	12,13	
tdc_conv_cnt	0	16...23	
tdc_sleepmode	1	17	
use_10kHz_lcd	16	18	1

5.3 List of configuration registers

Bit number →	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
parameter →			param1				param2									
Recommended value →							1	1	0	0	1	0	1	0	1	0

gray_labels = acam internal bits, use recommended settings (line below)

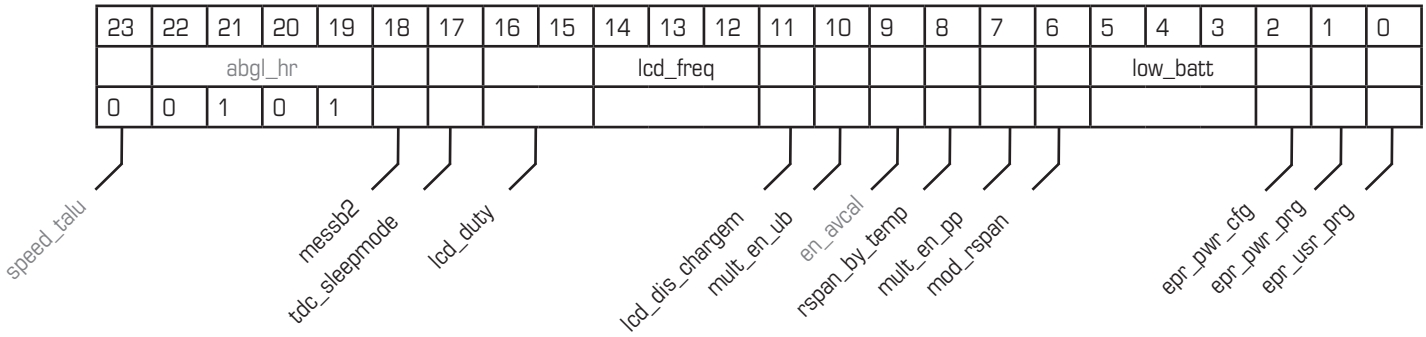
Configreg_00: RAM address 48 EEPROM bytes 0 - 2

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tdc_conv_cnt										io_a				osz10kHz_fsos									
														1	1	0	0	1	0	1	1	0	0

sel_compr, dis_osc_startup, cpu_speed, dis_haltpp_ps

Parameter	Recommended Value	Description	Settings
tdc_conv _cnt	-	Single Conversion Timer based on 10 kHz/64 = 156.25 Hz	
sel_compr	00	Selects comparator working resistor	00 = 10k 01 = 10k 10 = 7k 11 = 4.1k
io_a	-	Setting/reading MULT_IO0 to MULT_IO3 Output: output value, can be read back Input: read input value	
dis_osc_startup	1	Reduce current when starting the oscillator	1= on 0 = off

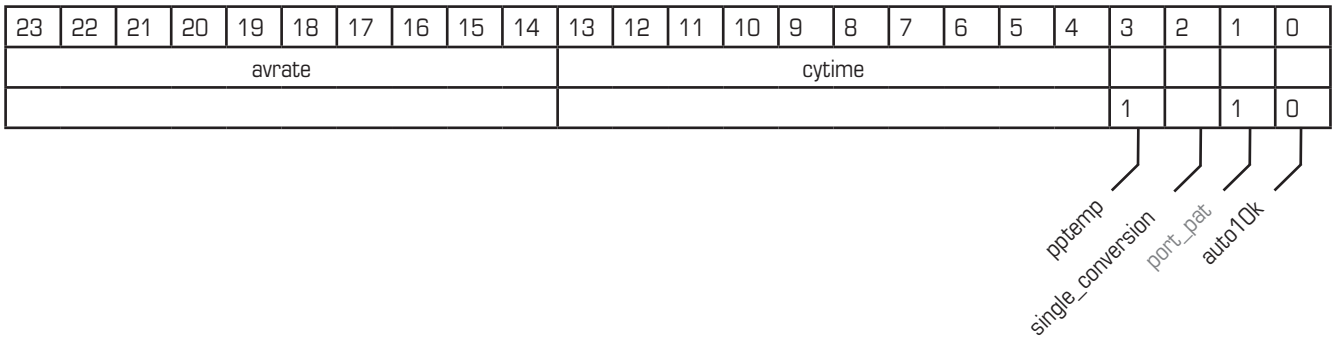
Configreg_01: RAM address 49 EEPROM bytes 3 - 5



Parameter	Recommended Value	Description	Settings
messb2	1	Set TDC measurement range 2	1 = on
tdc_sleepmode	-	Mode without TDC or strain gage measurement, to be used for scanning buttons in case the scale is off, same as avrate=0	1 = on 0 = off
lcd_duty	-	LCD duty cycle definition	0 = off 1 = 1/2duty 2 = 1/3duty 3 = 1/4duty
lcd_freq		Select LCD frequency (switch-on time of pixels)	
lcd_dis_chargem		Switch off LCD charge pump during the measurement	
mult_en_ub		Enable multiplications for supply voltage correction	1 = Enabled
rspan_by_temp		Use temperature measurement instead of Rspan for temperature compensation	1 = active
mult_en_pp		Enable multiplications in gain correction	1 = Enabled
mod_rspan		Enable internal multiplication of gain compensation resistor Rspan	1 = Enabled
lowbatt		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)
epr_pwr_cfg		Configuration in the EEprom is used after a power-on reset	as frontend := 0 stand-alone := 1
epr_pwr_prg		Start user code at EEPROM address 48 after a power-on reset	as frontend := 0 stand-alone := 1
epr_usr_prg		Start user code at EEPROM address 48 after a measurement	as frontend := 0 stand-alone := 1

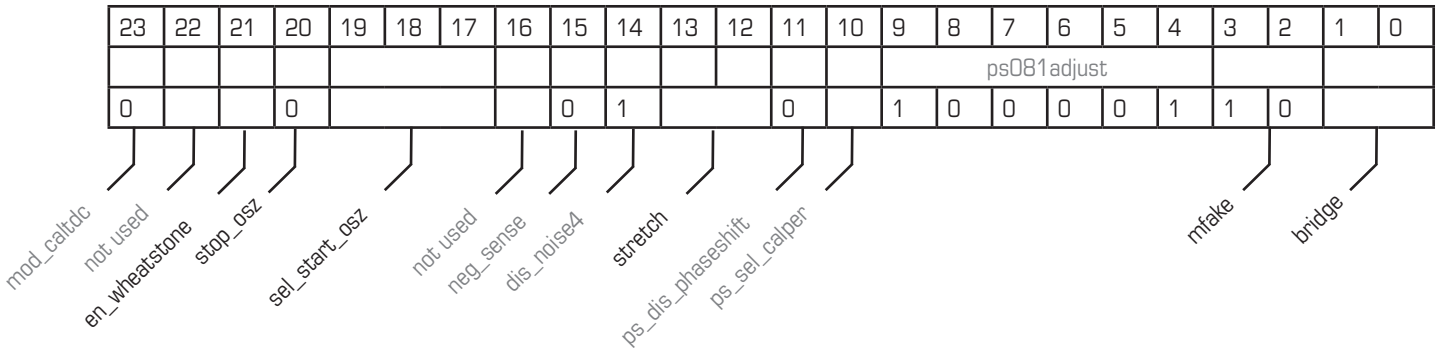
Pixel	Time	Multiplex mode			
		1/4	1/3	1/2	Hz
0	8.0ms	15	20	31	Hz
1	4.8ms	26	34	52	Hz
2	4.0ms	31	42	62	Hz
3	3.2ms	30	52	78	Hz
4	2.4ms	52	69	104	Hz
5	2.0ms	62	83	125	Hz
6	1.6ms	78	104	176	Hz
7	1.2ms	104	138	208	Hz

Configreg_02: RAM address 50 EEPROM bytes 6 - 8



Parameter	Recommended Value	Description	Settings
avrate	-	sample size of internal averaging	> 1
cytime	-	Cycle time in multiples 2 µs (8 * 4 M Hz period, stretch = 0) or of 100 µs (10 kHz period, stretch = 1)	
pptemp	1	Enable gain error and temperature measurement	1 = Enabled
single_conversion	-	Select operation mode	0 = Continous mode 1 = Single conversion mode
auto10k	-	Automatic calibration of the 10 kHz oscillator every 0.6 s by means of the 4 MHz quartz oscillator	May not be used in stretched single mode

Configreg_03 RAM address 51 EEPROM bytes 9 - 11



Parameter	Recommended Value	Description	Settings
en_wheatestone	-	Enable Wheatestone mode	1 = enabled
stop_osz	-	Stop the oscillator by command (e. g. if there is no interrupt after AutoOn)	1 = active
sel_start_osz	-	Sets delay from start of 4 MHz oscillator to start of measurement	0 = off 1 = continously on 2 = 100µs 3 = 200µs 4 = 300µs 5 = 400µs 6 & 7 are not supported

Parameter	Recommended Value	Description	Settings
stretch	-	Select stretch mode	0 = off 1 = single R measurement 2 = 2xR (half bridge), 200 µs delay 3 = 2xR (half bridge) 300 µs delay
mfake	10	Sets the number of fake measu-rements	
bridge	-	Sets the number of half bridges that are measured	0 = one half bridge (not reasonable) 1 = 2 half bridges 2 = not supported 3 = 4 half bridges

Configreg_04 RAM address 52 EEPROM bytes 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																							

Parameter	Recommended Value	Description	Settings
Mult_Hb1	-	Multiplication factor for HB1 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20	

Configreg_05 RAM address 53 EEPROM bytes 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																							

Parameter	Recommended Value	Description	Settings
Mult_Hb2	-	Multiplication factor for HB2 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20	

Configreg_06 RAM address 54 EEPROM bytes 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																							

Parameter	Recommended Value	Description	Settings
Mult_Hb3	-	Multiplication factor for HB3 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20	

Configreg_07 RAM address 55 EEPROM bytes 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																							

Parameter	Recommended Value	Description
Mult_Hb4	-	Multiplication factor for HB4 result h1 := h1 * [-2^23 to 2^23-1]/ 2^20

Configreg_08 RAM address 56 EEPROM bytes 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																							

Parameter	Recommended Value	Description
Mult_TkG	-	Multiplication factor for Rspan correction Rs := Rs * [-2^23 to 2^23-1]/ 2^20

Configreg_09 RAM address 57 EEPROM bytes 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																							

Parameter	Recommended Value	Description
Mult_Tk0	-	Offset value for Rspan, directly substracted

Configreg_10 RAM address 58 EEPROM bytes 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							
0	0	0	1	0	1	0	0																

Parameter	Recommended Value	Description
Mult_Ub	-	Multiplication factor for gain compensation by means of voltage measurement. hb := hb/(1 + ub*[-128 to 127]/2^21)
Mult_PP	-	Multiplication factor for gain correction. g := g * [0 to 255]/2^7

Configreg_11 RAM address 59 EEPROM bytes 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
								0	1	0		1	1	0	1	1	0		0	1			

io_en_3_mio, io_en_2_sck, io_en_1_sdi, io_en_0_sdo, dis_startdel, dis_pp_cycle_mod, force_unused_port, lcd_standby, lcd_vlt, lcd_fastld, lcd_r_const, lcd_swload1k, lcd_r_fastld, sel_compint, con_comp

Parameter	Recommended Value	Description	Settings
io_en_0_sdo	-	Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input
io_en_1_sdi		Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input
io_en_2_sck		Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input
io_en_3_mio		Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input
lcd_standby		LCD status	0 = LCD active 1 = LCD voltage supply in stand-by
lcd_vlt		LCD supply voltage	0 = 2V 1 = 2.5V 2 = 3V 3 = 2V without pump
lcd_fastld		Configures the number of fastload periods (10ms) with low-resistance voltage divider	
lcd_r_const		Defines the cross resistance of the LCD voltage divider	0 = 30 kΩ 1 = 50 kΩ 2 = 200 kΩ 3 = 800 kΩ
lcd_swload1k		LCD driver's voltage doubler uses 1 kΩ instead of 200 Ω to charge capacitor	

lcd_r_fastld		Selects the resistor for fast charging the LCD pixels	0 = 30 kΩ 1 = 50 kΩ 2 = 200 kΩ 3 = 800 kΩ
sel_compint		Select internal comparator	1 = Selected
con_comp		Controls the comparators switch-off mode.	00 = off 01 = off between single measurements 10 = off during loading and between single measurements 11 = always on

Configreg_12: RAM address 60 EEPROM bytes 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																							

Parameter	Recommended Value	Description	Settings
lcd_pos [23:0]	-	Position of lcd segments	

Configreg_13: RAM address 61 NOT mirrored in the EEPROM !

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																							

Parameter	Recommended Value	Description	Settings
lcd_segment [23:0]	-	Display segment digits 2 to 0	

Configreg_14: RAM address 62 NOT mirrored in the EEPROM !

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																							

Parameter	Recommended Value	Description	Settings
lcd_segment [47:24]	-	Display segment digits 5 to 3	

Configreg_15: RAM address 63 NOT mirrored in the EEPROM !

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																							

Parameter	Recommended Value	Description	Settings
lcd_segment [55:48]	-	Display segments special characters (1/3 duty ans 1/4 duty)	

Configreg_16: RAM address 64 EEPROM bytes 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
														selqha						multio_sel			
1	0				1	0	0	0	1	0	0	0	1	0	1	0	1	0	0				

ien_noise_c10_lcd
lcd_spi_ena
lcd_charge
lcd_direcrt_drive
use_10kHz_lcd
en_fullbr_p
ena_nonlin
clkdivepr
ps_speed
fha_en
sense_discharge
lmultihit_en
sel_speed_mh
run_la_cont

Parameter	Recommended Value	Description	Settings
en_noise_c10_lcd	1	In case the 10 kHz for the LCD charge pump is derived from the 4 MHz, this bit adds noise to the divider factor	1 = activated
lcd_spi_ena	0	Uses integrated LCD driver as SPI Interface for an external LCD driver Circuit	1 = enables external LCD driver control via SPI 0 = use internal LCD driver
lcd_charge[1:0]	-	Selects number of LCD cycles before recharging	0 = recharging each cycle 1 = recharging each 2nd cycle 2 = recharging each 3rd cycle 3 = recharging each 4th cycle
lcd_direct_drive	-	Drive LCD directly from supply voltage	1 = enabled
use_10kHz_lcd	1	Clock selection for LCD charge pump	0 = Charge pump clock derived from 4 MHz, divider can be set in the range 480 to 512. The selection avoids a fixed relation between charge pump and measurement and therefore reduced distortions of the measurement 1 = Use the internal 10kHz oscillator

sense_discharge	1	Set fast discharge of comparator's low pass capacitor	1 = enabled
multio_sel[3:0]	-	Use mMULTIO03 pin for diagnoses	0 = multiple I/O 1 ... 11 = internal use 12 = interrupt

Configreg_17: RAM address 65 EEPROM bytes 42 - 44

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
spi_delay				lcd_pulsed				c10_div_lcd					lcd_spi_out			io_a		en_io5		en_io4		clkssyc	
0	1	1											0	0	0			1	0	1	0	1	0

n. c.

Parameter	Recommended Value	Description	Settings
spi_delay[2:0]	3	Sets the timing for the LCD SPI master interface	0 = fast (about 500ns period) . . 7 = slow (about 10µs period)
lcd_pulsed[3:0]	-	Automatically pulsed display, especially for stand-by in solar applications	00xx = 300 ms blinking period 01xx = 600 ms blinking period 10xx = 1 s blinking period 11xx = 2 s blinking period xx00 = of on-time xx01 = 100 ms on-time xx10 = 200 ms on-time xx11 = 400 ms on-time
c10_div_lcd[5:0]	-	5 bit divider factor for deriving the 10kHz from the 4 MHz. Divider factor = 480+c10_div_lcd	
lcd_spi_out[2:0]	1	Set the LCD pins as SPI outputs, 6 pins in parallel each	xx1 = switch on LCD_COM1... LCD_COM4,LCD_SEG1, LCD_SEG2 as SCK x1x = switch on LCD_SEG3... LCD_SEG8 as SSN 1xx = switch on LCD_SEG9... LCD_SEG14 as SDO
io_a[1:0]	-	Setting/reading MULT_IO4 and MULT_IO5. Output: output value, can be read back Input: read input value	

en_io4[1:0]	2	Port definition for MULT_IO4	00 = output 01 = input with pull-up 10 = input with pull-down 11 = input
enio5[1:0]	2	Port definition for MULT_IO5	00 = output 01 = input with pull-up 10 = input with pull-down 11 = input
clksyc[1:0]	1	Speed of key Synchronization	

Configreg_18: RAM address 66 EEPROM bytes 45 - 47

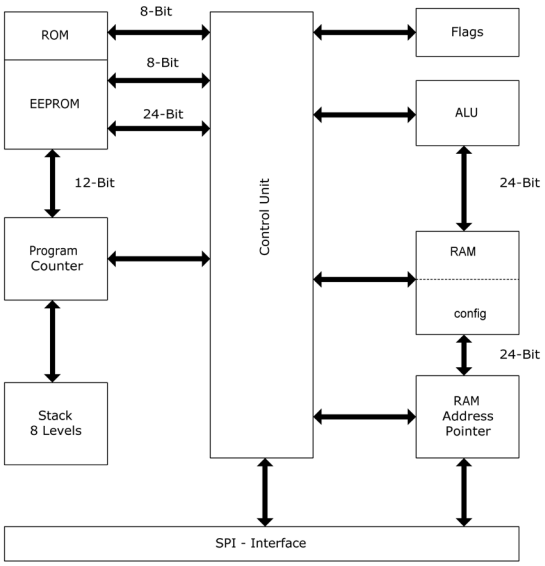
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_NLK																							

Table of Contents Page

6 Central Processing Unit (CPU)

6.1 Block Diagram

Figure 6.1 Block Diagram



6.2 Memory Organization

6.2.1 ROM and EEPROM Organization

Table 6.1

5119 … 2048	ROM Program Memory
2047 … 2000	Program Memory User EEPROM 48 bytes
1999 … 48	Program Memory EEPROM 1.952 bytes Program entry
47…45 44…42 41…39	Config Reg 18 (mirrored) Config Reg 17 (mirrored) Config Reg 16 (mirrored)
38…35 5…3 2…0	Config Reg 12 (mirrored) Config Reg 1 (mirrored) Config Reg 0 (mirrored)

The ROM area is starting at address 2048. All computation routines needed for the PICOSTRAIN measuring method reside here. There are also further helpful routines that are frequently needed in weight scale applications, e.g. decimal to 7-segment code conversion, averaging, median filter, rounding and hysteresis. 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 2048 bytes big, split in 4 blocks of 512 bytes. The program memory occupies 1952 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 PSØ81 during a power-on reset. 3 successive bytes are added to a 24 bit word. So there are 16 words of 24 bit that can be read by the program code. They are used for configuration register 0 to 12 and 16 to 18. During a power- on reset they are copied into RAM address 48 to 66.

EEPROM cells 2000 to 2047 are freely accessible. The processor can write to and read from those cells during operation (putepr and getepr, using addresses 0 to 15). They can be used for saving calibration data.

6.2.2 RAM Organization

Table 6.2

127 … 112	User RAM 127 … User RAM 112	Temporarily used by some ROM-routines like median or sinc filter (starting from address 112)
111 … 100 99…98 97 … 67	Shadow RAM (used by converter and ALU) Content of LCD segments (external) Shadow RAM (used by converter and ALU)	
66 … 48	Config Reg 18 … Config Reg 0	
47 … 32	User RAM 47 … User RAM 32	
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	- - Timer I/O status Pressed I/O status Falling edges I/O status Rising edges UBATT CAL HB1+ Flags (p1-p2)/p2 HB0 = (A-B)+(C-D)+(…)/(A+B)+(C+D)+(…) * 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	

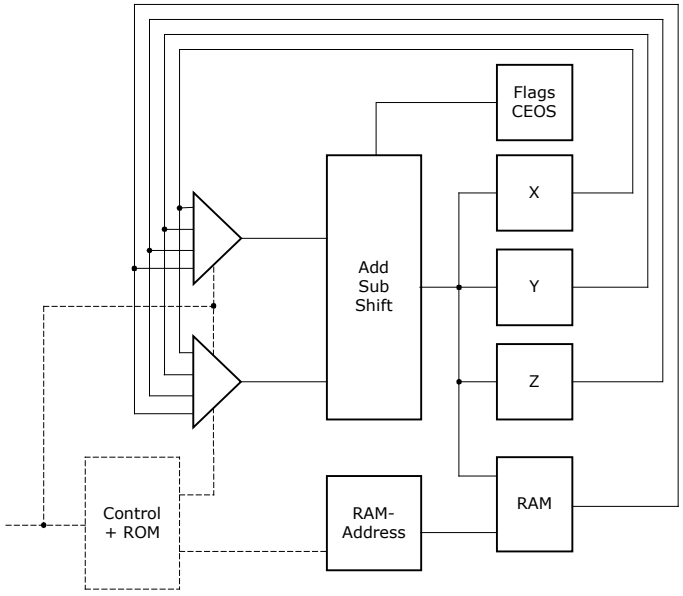
Parameters A..F represent the discharging times at the different ports, see section 6.2.4 Result Registers for more details

6.2.3 RAM Address Pointer

The RAM has its own address bus with 127 addresses. The width of 24 bit corresponds to the register 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,…)

6.2.4 Arithmetic Logic Unit (ALU)

Figure 6.2 ALU block diagram



6.2.5 Accumulators

The ALU has three 24-Bit accumulators, X, Y and Z. The RAM is addressed by the RAM address pointer and the addressed RAM cell is used as forth accumulator. A single RAM address is mapped into the ALU by the ram address pointer. So in total there are 4 accumulators. All transfer operations (move, swap) and arithmetic–operations (shift, add, mult24…) can be applied to all accumulators.

6.2.6 Flags

The processor controls 4 flags with 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.

6.2.6.1 Carry

Shows the carry over in an addition or subtraction. With shift operations (shiftL, rotR etc.) it shows the postponed bit.

6.2.6.2 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.).

6.2.6.3 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.

6.2.6.4 Overflow

Indicates an overflow during an addition or subtraction of two numbers in the meaning of two's complement.

6.3 Status and Result Registers

6.3.1 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)+(C-D)+(..)/ (A+B)+(C+D)+(..)	HB0 compensated sum
ram=21	: TMP=(p1-p2) / p2	Temperature
ram=22	: Status	See 6.3.2
ram=23	: HB1+	Time measurement TDC at SG_A1, Pin11
ram=24	: CAL	Resolution TDC
ram=25	: UBATT	Measured supply voltage
ram=26	: Status_F	Indicates I/Os and pairs of I/Os with HIGH-to-LOW
ram=27	: Status_R	Indicates I/Os and pairs of I/Os with LOW-to-HIGH
ram=28	: Status_P	Indicates I/Os and pairs of I/Os with HIGH
ram=29	: Timer	
ram=30,31	: NC	Free, can be used temporarily, will be overwritten during measurement

x-Accu	: HB0	Value of ram=20
y-Accu	: Temp	Value of ram=21
z-Accu	: Flags	Value of ram=63
ramadr	: 0	Value of RAM address pointer

Descriptions:

A :	Discharge time measurement at SG_A1
B :	Discharge time measurement at SG_A2
C :	Discharge time measurement at SG_B1
D :	Discharge time measurement at SG_B2
E :	Discharge time measurement at SG_C1
F :	Discharge time measurement at SG_C2
G :	Discharge time measurement at SG_D1
H :	Discharge time measurement at SG_D2
P1:	Discharge time measurement at TMP_1
P2:	Discharge time measurement at TMP_2

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:	Result(Tmp) = $TMP/2^{24}$
Status:	See above
HB1+:	Result (HB1+)/ns = 250 * HB1+ /2^{14} [4MHz clock]
CAL:	Result (Cal)/ps = 250,000 / CAL [4MHz clock]
UBATT:	Result (UBATT)/V = 2.0+1.6*UBATT/64

HB1, HB2, HB3, HB4, HB0 and TMP are given as two's complement. MSB = 1 indicates a negative value. To get the positive value calculate $X - 2^{24}$.

Explanation:

Based on a standard extension of a load cell (2 mV/V) the resistance variation is 0.2 %, e.g. 2 Ω at a 1000 Ω load cell. The change of 0.2 % corresponds to 2000 ppm. For reasons of internal calculations and accuracy, the result is given in x100 of 2000 ppm (= 200,000 ppm). Please note, that the value in this register depends not only on the load cell's sensitivity but also on the Mult_HBx setting in PS081. This explanation is based on Mult_HBx = 1.

Examples:

1.5 mV/V load cell, PICOSTRAIN wiring, Mult_HBx = 1:

1.5 mV/V = 1500 ppm → result in PS081 at maximum strain: 150,000 (0x0249F0)

2 mV/V load cell, Wheatstone wiring, Mult_HBx = 1:

2 mV/V means 1.333 mV/V in Wheatstone = 1333 ppm (due to a reduction in strain) → result in PS081 at maximum strain: 133,333 (0x0208D5)

1 mV/V load cell, Picostrain wiring, Mult_HBx = 4:

1 mV/V = 1000 ppm → result in PS081 at max. strain: 400,000 (0x061A80)

6.3.2 Status Register

Table 6.3 Status Register (RAM Address 22)

Bit	Description
Status[23]= flg_io3_mio	Pin26(32)
Status[22]= flg_io2_sck	Pin20(25)
Status[21]= flg_io1_sdi	Pin19(24)
Status[20]= flg_io0_sdo	Pin18(23)
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]= error_dspclk	1 = Collision between TDC and DSP
Status[07]= flg_io3_mio_r	1 = Rising edge at Pin26(32), 0 = no edge
Status[06]= flg_io2_sck_r	1 = Rising edge at Pin20(25), 0 = no edge
Status[05]= flg_io1_sdi_r	1 = Rising edge at Pin19(24), 0 = no edge
Status[04]= flg_io0_sdo_r	1 = Rising edge at Pin18(23), 0 = no edge
Status[03]= flg_io3_mio_f	1 = Falling edge at Pin26(32), 0 = no edge
Status[02]= flg_io2_sck_f	1 = Falling edge at Pin20(25), 0 = no edge
Status[01]= flg_io1_sdi_f	1 = Falling edge at Pin19(24), 0 = no edge
Status[00]= flg_io0_sdo_f	1 = Falling edge at Pin18(23), 0 = no edge

Pin numbers in brackets = dice

The status of the inputs can be queried from the status regsiters at RAM addres 26 to 28. Please see chapter 4.4.4 on page 4-17 for more details

6.4 Instruction Set

The complete instruction set of the PSØ81 consists of 94 core instructions that have unique op-code decoded by the CPU. Further there are emulated instructions like no2lcd that are replaced automatically by the assembler and call a subroutines in the ROM code.

The variety of the instruction set allows to write comprehensive programs that cope with the 2 K size of the EEPROM.

6.4.1 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 goto-instructions.

It is possible to jump into subroutines only by means of absolute jumps and without any condition.

6.4.2 Arithmetic operations

The RAM is organized in 24 Bit words. All instructions are based on two's complement operations. An arithmetic command combines two accumulators and writes back the result into 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.

Table 6.4 Instruction set

Simple Arithmetic	Complex Arithmetic	Shift & Rotate	RAM access
abs	div24	clrC	clear
add	divmod	rotl	decramadr
compare	mult24	rotR	incramadr
compl	mult48	setC	move
decr		shiftL	ramadr
getflag		shiftR	swap
incr			
sign			
sub			

Logic	Bitwise	LCD display	EEPROM access
and	bitclr	clrLCD	equal
eor	bitinv	dez2lcd	getepr
nor	bitset	newlcd	putepr
invert		no2lcd	useEprInit
nand		no2LcdAccu	
nor		setLCD	
or		spi2lcd	

Unconditional jump	Skip on Flag	Miscellaneous
goto	skip	clk10kHz
gotoBitC	skipBitC	clrwdt
gotoBitS	skipBitS	gr2st
gotoCarC	skipCarC	hysterises
gotoCarS	skipCarS	initAvg
gotoEQ	skipEQ	initTDC
gotoNE	skipNE	median
gotoNeg	skipNeg	newcyc
gotoOvrC	skipOvrC	nop
gotoOvrS	skipOvrS	rollAvg
gotoPos	skipPos	round
jsub		ssnPulse
jsubret		ssnSet
		stop

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 arithmetic

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 arithmetic

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 registers 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

clrLCD	Clear LCD
Syntax:	clrLCD
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	Subroutine call
Description:	Clear LCD registers 61 & 62. Use this opcode in combination with ‚newlcd' to switch off all LCD segments. In real a subroutine in the ROM code is called. The assembler converts this command to the corresponding jump command.
Category:	LCD Display

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 program 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 constant 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 RAM address pointer by one
Category:	Ram Access

dez2lcd	**Decimal 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 op-code no2lcd is not sufficient dec hgfe dcba 0 --> 0b00111111=0x3F 1 --> 0b00000110=0x06 2 --> 0b00111011=0x3B 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| < |p2/2|)
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 bit combination 0 / 0 and 1 / 1 returns 0 bit combination 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 bit combination 0 / 0 and 1 / 1 return 1 bit combination 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 op-code is simply used 16 times in the beginning of the assembler listing, fed 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 RAM address 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 RAM address pointer. This opcode needs tempo-rarily 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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
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 jump label. See examples section for how to introduce a jump label.
Category:	Goto on flag

gotoPos	Jump on positive
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 positive. Program counter will be set to target address if the foregoing result is positive. The target address is given by using a jump label. See examples section for how to introduce a jump label.
Category:	Goto on flag

gr2st	Gram to Stone Conversion
Syntax:	gr2st p1
Parameters:	p1 = ACCU [x]
Calculus:	p1 = stone (p1)
Flags affected:	-
Bytes:	3
Cycles:	185
Description:	Converts weight in multiples of 100mg to stone. The result is given with 2 positions in stone, the rest in pounds. The value can be directly used with no2lcd.
Category:	Miscellaneous

hysteresis	**Hysteresis**
Syntax:	hysteresis p1,p2,p3
Parameters:	p1 = ACCU [x] p2 = NUMBER [half scale division] p2 = NUMBER [half scale division + half hysteresis]
Calculus:	p1 = hysteresis (p1)
Flags affected:	-
Bytes:	10
Cycles:	subroutine call
Description:	This opcode provides a hysteresis function. By parameter p3 you can define a value range which is considered as „tolerance" for the variation of the actual measurement value. In other words, although another division normally would be displayed already, the current division is kept. This opcode contains also the round opcode. The round function is set by parameter p2. See also the explanation further down on this page.
Category:	Miscellaneous

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

initAvg	**Initialize rolling average**
Syntax:	initAvg p1,p2
Parameters:	p1 = ACCU[x] p2 = number from 3 to 17
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	Subroutine call
Description:	Initialization of the rolling average subroutine. p1 sets the default value for the calculus. p2 defines the number of data points for the rolling average. In real a subroutine in the ROM code is called. The assembler converts this command to the corresponding jump command.
Category:	Miscellaneous

initTDC	**Initialize TDC**
Syntax:	initTDC
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	Initialization reset of the TDC (time-to-digital converter). Should be sent after configuration of registers. The initTDC preserves all configurations .
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 program counter is loaded by the address given through the jump label. 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

median	**Median Filter**
Syntax:	median p1,p2
Parameters:	p1 = ACCU [x] p2 = NUMBER [3 to 15, uneven]
Calculus:	p1 = median [p1+r(32)+r(33)+...+r(32+p2-2)]
Flags affected:	S Z (of result p1)
Bytes:	7
Cycles:	subroutine call
Description:	Applies median filter with configured depth. The number to be added is given by the Accu X. The number defines the depth of the filter, only uneven numbers are valid. To use this opcode the corresponding RAM-cells need to be initialized, this can be done by initAvg
Category:	Miscellaneous

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

mult48	**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 multiplication 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 bit combination 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 initialization of the PSØ81 to start a new measurement cycle. Normally this is done by the PSØ81 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 p1,p2
Parameters:	p1 = ACCU[x] p2 = number 1 to 6
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	subroutine call
Description:	Converts the 6 digits of the decimal number in register x into 7-segment code. The position of the decimal point is determined with p2. 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'.
Category:	LCD display

no2lcdAccu	**Covert 6 digits to LCD code**
Syntax:	no2lcd p1,p2
Parameters:	p1 = ACCU[x] p1 = ACCU[Y]
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	subroutine call
Description:	Converts the 6 digits of the decimal number in register x into 7-segment code. The position of the decimal point is determined with p2. 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'.
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 bit combination 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 bit combination 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 RAM address 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 RAM address pointer). EEPROM bytes 2000 to 2047 are accessible via ‚putepr', setting the RAM address pointer to addresses 0 to 15. 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 RAM address (range: 0..65)
Category:	RAM access

rollAvg	Calculate rolling average
Syntax:	rollAvg p1,p2
Parameters:	p1 = ACCU[x] p2 = number from 3 to 17
Calculus:	ACCU[y] = ACCU[x] old p1 = rolling average result
Flags affected:	-
Bytes:	2
Cycles:	Subroutine call
Description:	Feeds p1 as new value into the rolling average subroutine. p2 defines the number of data points for the rolling average. The value falling out of the rolling average is stored on RAM address 13. The result is stored in the x ACCU. The previous result is stored in the y ACCU. In real a subroutine in the ROM code is called. The assembler converts this command to the corresponding jump command.
Category:	Miscellaneous

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

round	Rounding
Syntax:	round p1,p2
Parameters:	p1 = ACCU [x] p2 = NUMBER [half scale division]
Calculus:	p1 = round (p1, p2)
Flags affected:	
Bytes:	7
Cycles:	subroutine call
Description:	Rounds the number in x. Depending on the configured ‚half scale division' the number stored in x will be rounded down or up (down <5, up >= 5).
Category:	Miscellaneous

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

setLCD	**Set LCD**
Syntax:	setLCD
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	Subroutine call
Description:	Sets all LCD register 61 & 62 bits to 1. Use this opcode in combination with ‚newlcd' for showing all LCD segments. In real a subroutine in the ROM code is called. The assembler converts this command to the corresponding jump command.
Category:	LCD Display

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 positive
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 positive
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

spi2lcd	Send LCD display value to SPI
Syntax:	spi2lcd p1,p2
Parameters:	p1 = ACCU [y] p2 = NUMBER [number of bits to transmit]
Calculus:	-
Flags affected:	-
Bytes:	7
Cycles:	subroutine call
Description:	This opcode sends the content of accu y to the SPI interface for external LCD controller. The second parameter defines, how many bits are transmitted. This function is mainly needed when operating an external LCD driver via SPI. Please note: This opcode requires a special configuration of PSØ81 and also needs suceeding opcodes to work properly. Please see chapter 4.3 “Support of an external LCD driver” for mor deteils.
Category:	LCD display

ssnPulse	Send Pulse on SSN for external LCD controller
Syntax:	ssnPulse
Parameters:	-
Calculus:	SSN -> 0 -> 1 -> 1
Flags affected:	-
Bytes:	3
Cycles:	
Description:	Generates positive pulse on SSN (pins LCD_SEG3...8), ends in logic 0
Category:	Miscellaneous

ssnSet	Set SSN to HIGH for external LCD controller
Syntax:	ssnSet
Parameters:	-
Calculus:	SSN -> 1
Flags affected:	-
Bytes:	3
Cycles:	
Description:	Sets pin SSN (pins LCD_SEG3...8) to HIGH
Category:	Miscellaneous

stop	Stop
Syntax:	stop
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	1
Cycles:	1
Description:	The DSP and clock generator are 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,r) p2 = ACCU (x,y,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. Not possible with ACCU(z)
Category:	RAM Access

useEprInit	Initialize User - EEPROM
Syntax:	usrEprInit p1, p2
Parameters:	p1 = EEPROM CELL (0 to 15) p2 = NUMBER (24-bit)
Calculus:	EEPROM (p1) = p2
Flags affected:	-
Bytes:	1
Cycles:	3
Description:	This opcode initializes the User programmable EEPROM space from address 2000 to 2047. In total there are 16 EEPROM cells á 24-bit programmable. Please note: This opcode is only for INITIALIZATION. For writing / reading from the user EEPROM cells from the program you have to use putepr / getepr opcodes
Category:	EEPROM access

6.5 System Reset, Sleep Mode and Auto-configuration

ALU activity is requested by a reset (power-on, watchdog), the end of measurement or in sleep mode the end of the conversion counter. A reset has priority over the last two items. First the ALU jumps into the ROM code starting with address 2048. There a first check is done whether the ALU was activated after a reset or not.

In case of a reset the flag epr_pwr_cfg is checked to decide whether the auto-configuration 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. In stand alone operation this is reasonable and epr_pwr_cfg bit should be 1. In front end operation this is unlikely and with epr_pwr_cfg = 0 the µP is stopped.

In case the ALU is started not by a reset the TDC unit starts a measurement or, in sleep mode, the conversion counter is started without a measurement. Afterwards the flag epr_usr_prg is checked to decide about a jump into the EEPROM (address 48). Again, in stand-alone operation epr_usr_prg =1 is reasonable, in front-end operation epr_usr_prg = 0 will be more likely.

In the EEPROM code first the flag flg_rstpwr should be checked to see whether the reason for the jump was a reset. If yes, a detailed check is recommended to see whether the reset comes from a power-on reset, a pushed button, the watchdog interrupt.

Otherwise a check of flag flg_intav0 will indicate if the chip is still in sleep mode or if an active strain measurement is running.

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

6.5.1 Power On Reset

When applying the supply voltage to the chip a power-on reset is generated. The whole chip is reset, only the RAM remains unchanged.

In case epr_pwr_prg = 1 the user code at EEPROM address 48 is started.

6.5.2 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 epr_pwr_prg = 1 the user code at EEPROM address 48 is started.

6.5.3 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

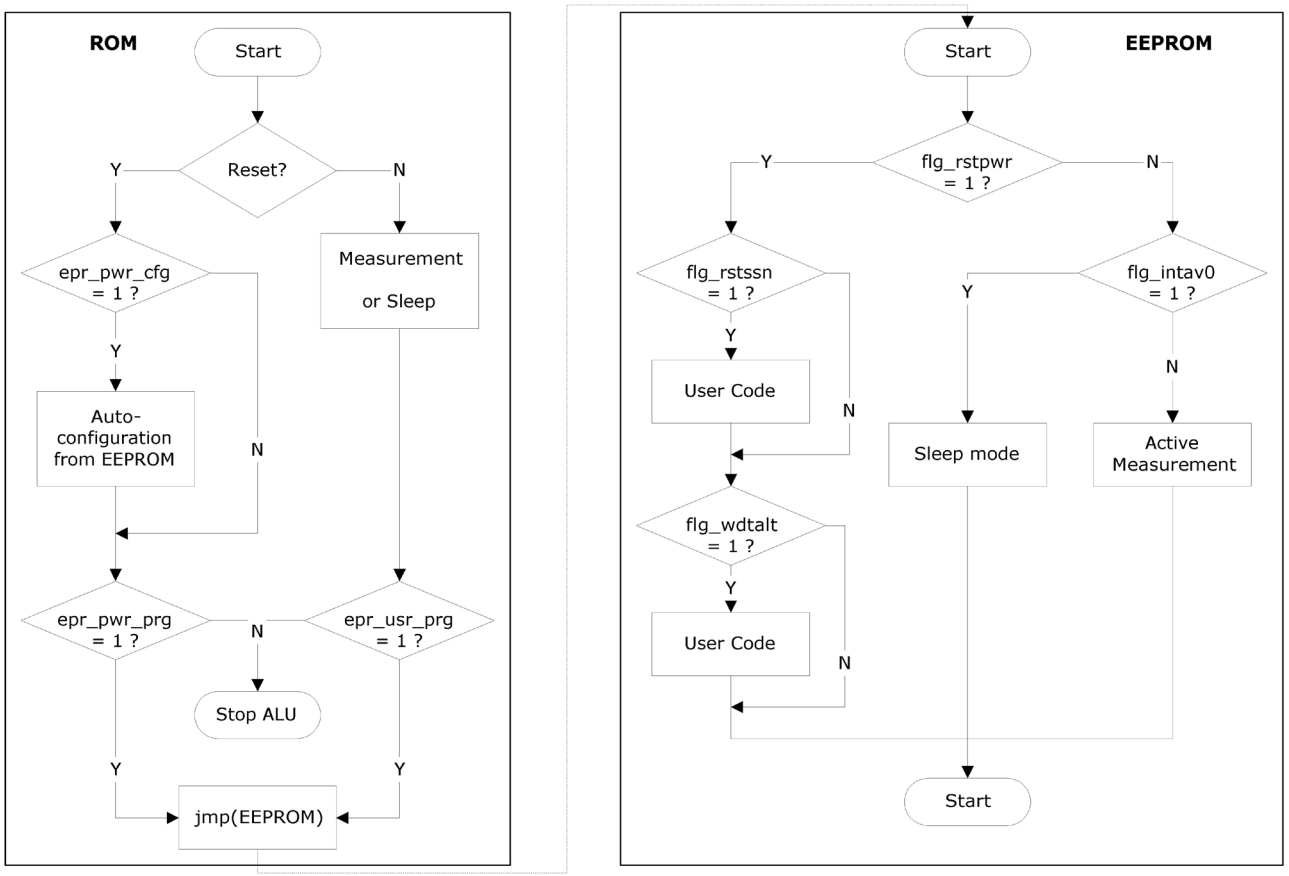


Figure 6.3 Flow chart

flg_rstssn in register 22, bit 18.

In case epr_pwr_prg = 1 the user code at EEPROM address 48 is started.

6.5.4 Sleep Mode

In 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:	tdc_sleepmode	Register 1, Bit 17
	tdc_conv_cnt[11:0]	Reg0, Bits 23 to 14

Sleep mode is activated by setting tdc_sleepmode = 1. This is equivalent to set avrate = 0.

In sleep mode the conversion counter tdc_cnv_cnt () is running to the end and then immediately starting the user program beginning at address 48 in the EEPROM.

After running in sleep mode the TDC has to be reinitialized for measurements.

6.5.5 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
- for the TDC unit in measurement range 2 as pre-counter
- as basis for the cycle time in stretched modes

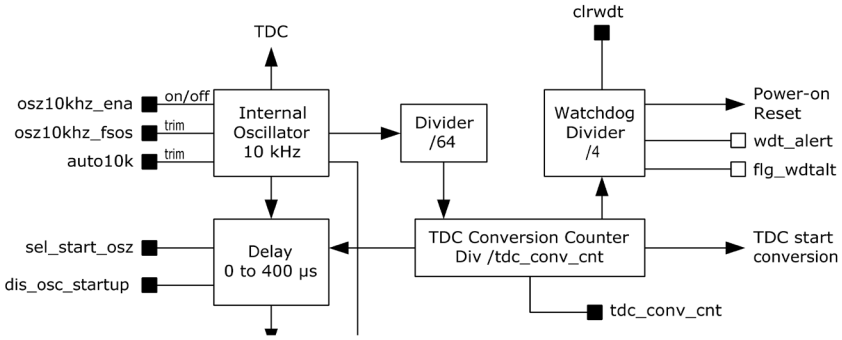


figure 6.4 Clock Generation

6.5.6 Watchdog Counter and Single Conversion Counter

The TDC conversion counter starts a measurement in single conversion mode. It is running continuously. The single conversion rate is given by 10 kHz / 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 has to be reset by CLRWDT 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 controlling the PSØ81 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.

6.5.7 Timer

PSØ81has a real time counter that counts autonmatically after a power-on reset in periods of 10 ms. The actual position of this counter can be read from RAM address 29. The counter rolls over at 2^{24} bit, which corresponds to a period of 46 hours. The counter can be reset by opcode clearTimer.

Table of Contents Page

7 Miscellaneous

7.1 Migration from PS08 to PS081

The following table shows the differences between PS08 and PS081:

	PS08	PS081
EEPROM Program Space:	1Kx8 bit	2Kx8 bit
EEPROM User Space:	2x24 bit	16x24 bit
RAM User Space:	32x24 bit	48x24 bit
Inputs:	max. 5	max. 21
GPIOs (Mult_IO):	1	3
Configuration Registers:	17	19
External LCD via SPI:	no	yes
Opcodes:		same as PS08 plus: - Anti vibration filter - Hysteresis - Round - Gramm to Stone conversion
Package:	QFN56	QFN56
Dice:	2770x2520µm² Pad: 116µm x 90µm	2370x3470µm² Pad: 116µm x 90µm

Hardware changes:

From a hardware point of view the changes are rather small. If the packaged version is used (QFN56), PS08 can be replaced by PS081 on existing PCBs. Only thing to pay attention to is that Pin 13 and 15 have changed, this is Pin 13 from VCC --> Mult_IO5 and Pin 15 GND --> Mult_IO4.

So to replace PS08 by PS081 on an existing PCB, there are 2 possibilities:

- don't connect pin 13 and 15 with PS081
- or: connect them, but configure them both as inputs (configreg_17, en_io4 and en_io5)

Software changes:

PS08 has 17 configuration registers (configreg_00 to configreg_16). PS081 has 2 more and configreg_16 has changed. In other words, you have to put a configuration in the additional registers, in the first instance you can just take the default values recommended in the data sheet.

Some very few opcodes have changed, e.g. no2lcd in the way the parameters are given. In PS081 there are additional opcodes available such as median, hysteresis or round. A description of the new opcodes and the changes you find in the online help of PS081 Assembler version. There's also an updated version for the evaluation software.

Other changes:

SPI opcode „Read Access“ has changed. In PS081 the 4 EEPROM cells are addressed subsequently.

The parameter „PS081Adjust“ (configreg_03, bit [9:4]) needs to be set in PS081, if the AV-rate setting is <10. PS081adjust is thereby to set two times the AV-rate and a minimum of 8. E.g. the AV-rate is 6, then PS081adjust would be 12.

7.2 Bug Report

Date	Reported Bug	Solution (if any)
Dec 2009	Cancelation of Full bridge parallel mode (zero drift optimized)	Use standard full bridge mode
	Background: This mode was introduced to reduce the zero drift of PS081 by paralleling RDSON resistors on the chip. Intensive tests of several bonding options did unfortunately not show any better behavior, in some cases even worse. Please use standard full bridge mode instead, the systematic offset drift lies here in the range of ±6nV/V/K and can be cross matched on the PCB.	
July 2010	If any I/O pin is used as output, a maximum of 4 inputs can be configured. In other words, to increase the number of inputs up to 21 is only possible if no output is configured.	
	Background: To increase the number of inputs an internal logic was added to connect not only the 6 physical I/Os directly, but also combination of the lines. This works fine as long as only inputs are configured. In July 2010 it was observed that as soon as 1 pin is configured as output, this internal logic does not work properly any longer. This means, that the register 26, 27 and 28 which are normally used to indicate the rising, pressed or falling flags of the buttons are not updated properly. So if an output is configured, there are still inputs available, but only 4 as maximum and their status need to be checked in status register 22 instead of 26 to 28. In this sense, the maximum number of inputs and outputs when used mixed, is input = 3 and output = 3. If you have an application with an update rate of >= 100 Hz there is another work-around option, please contact the support team of acam in this case.	

May 2012	When exactly two of the six MultIOs are simultaneously '1' and the rest four are '0', then a multi input key(Section 4.4.4) is automatically detected. The condition of two MultiIO pins being at '1' can happen due to any of the following reasons: 1. For e.g. when two output ports are configured simultaneously to 1 from the program. 2. When two inputs ports are set simultaneously to '1' externally. 3. When two inputs are configured simultaneously (not one after another) with pull up resistors. 4. When an input is switched simultaneously with an output.
	Regardless of how the condition occurs, as a result the outputs function normally. But the status registers showing the input status of MultI00 – MultI05 (RAM address 22 and 26-28) are blocked by the falsely detected multi input. If it occurs during configuration then further configuration of the 4 other IOs are blocked. Once occurred, this condition can be resolved only by setting each of the previously '1' MultIOs to 0. Workaround: Configure one MultIO port by default to 1 (either as output '1' or as pulled up input) permanently. This constant '1' on one MultiIO port ensures the suppression of the unwanted multi input detection from all the error sources listed. This however leaves only 5 MultiIOs for effective usage.

7.3 Known issues and solutions

Issue: (known since July 2012)

Supply voltage reduction below 0.8V can cause an undefined state of PS081 („brown-out effect") where the chip does not react to the I/Os or interfaces anymore.

Solution:

When supply voltage is reduced, make sure it is reduced to true 0V and not only to a level < 0.8V.

7.4 Literature Guide

Datasheets

Title	Document-No	Date
PSØ81-EVA Evaluation System for PSØ81	DB_PS081_EVA V1.0	September 2009
ALCS-350 V2 Load Cell Simulator	DB_ALCS_V2 V0.1	August 2009
PicoProg Ø81	DB_PicoProg_Ø81_en_V0.1	January 2010

Whitepapers

Title	Document-No	Date
Construction Guideline for solar driven Scales	WP001 V2.0	October 2008
How to Lower Gain and Offset Drift Drift of a Load Cell by using TGGain and TKOffset Factors of PSØ81	WP002 V1.1	February 2010
Temperature Measurement with PS08	WP003 V1.0	August 2009
How to build Digital Load Cells with PICOSTRAIN conveniently	WP004 V1.0	March 2010

Application Notes

Title	Document-No	Date
Metrological Investigations of PSØ81 Determining Zero Drift and Gain Drift	AN018 V1.0	July 2008
Strain Gauge Wiring with PICOSTRAIN	AN012 V1.0	August 2005
Rspan by Temp Compensation Compensation of Gain error for uncompensated Load Cells	AN021 V0.5	November 2009
Design Guidelines for Building a Solar Kitchen Scale	AN022 V1.3	March 2010
Design Guidelines for Building a Solar Body Scale	AN023 V1.3	September 2009
EMI Countermeasures for a Digital Load Cell	AN025 V1.0	June 2010

All available documents can be downloaded from the acam website at:

http://www.acam.de/download-section/picostrain

7.5 Document History

22.06.09 First release of preliminary version

09.07.09 Second release of preliminary version

31.08.09 Release of version V0.3. New Mult_pp factors, new POR circuit, minor corrections

04.09.09 Section 3.3, new figures for load cell wiring, section 5, register numbering

18.12.09 Release of Version V0.5, updating section 3.3 Connecting the SG, cancellation of fullbridge paralled mode

15.01.10 Section 4.6.1 added, section „Contacts/Distributors“ moved to the end of the

document, literature guide completed

12.02.10 Relase version V0.6. Adding new sections 3.6.3 and 3.6.4

24.06.10 3.4.6 Resolution updated, 6.3.1 status register corrected, new section 7.1

03.08.10 Chapter 4, I/O revised. Chapter 7, bug report.

04.07.12 Chapter 4, Figures 4-13 and 4-17 corrected. Bug added to bug report in Chapter 7, changed Section 7.3.

Table of Contents Page

8 Appendix

8.1 High-Resolution Investigations

To investigate the highest possible resolution of the converter we picked a high quality load cell from HBM and recorded the noise figures. The set up was as follows:

- HBM-SP4C3 load cell, modified to Picostrain wiring
- Only 1 Rspan was used, the other one shortened
- The supply voltage was 4.5 V
- The PCB was the high resolution plug-in module of the evaluation kit
- Median filter was used

Two figures were determined, the noise behavior and the stabilization time. Please see the following table for an overview:

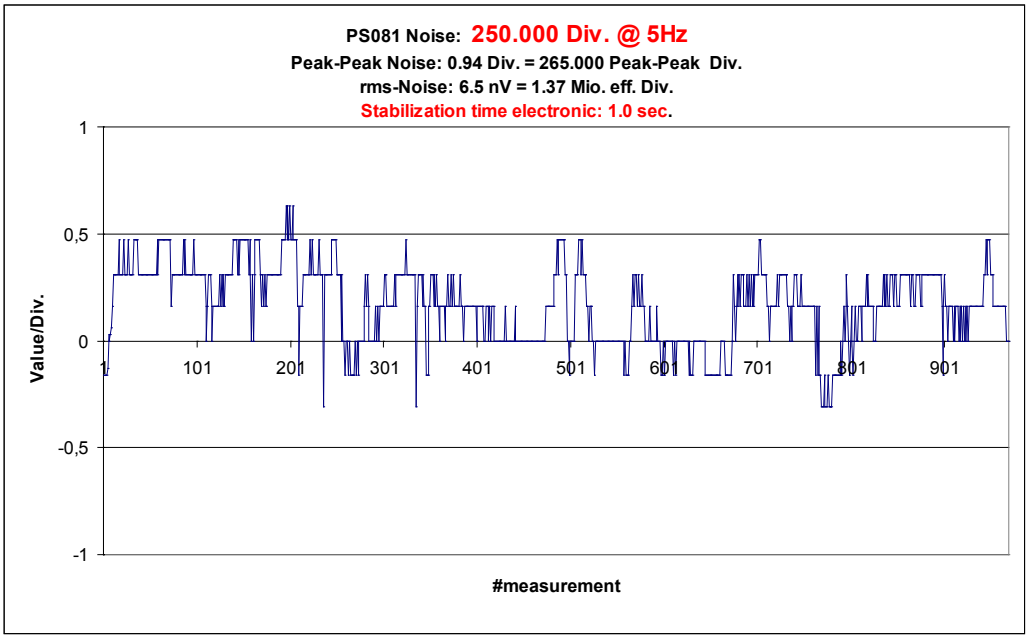
Table 8.1 Resolution

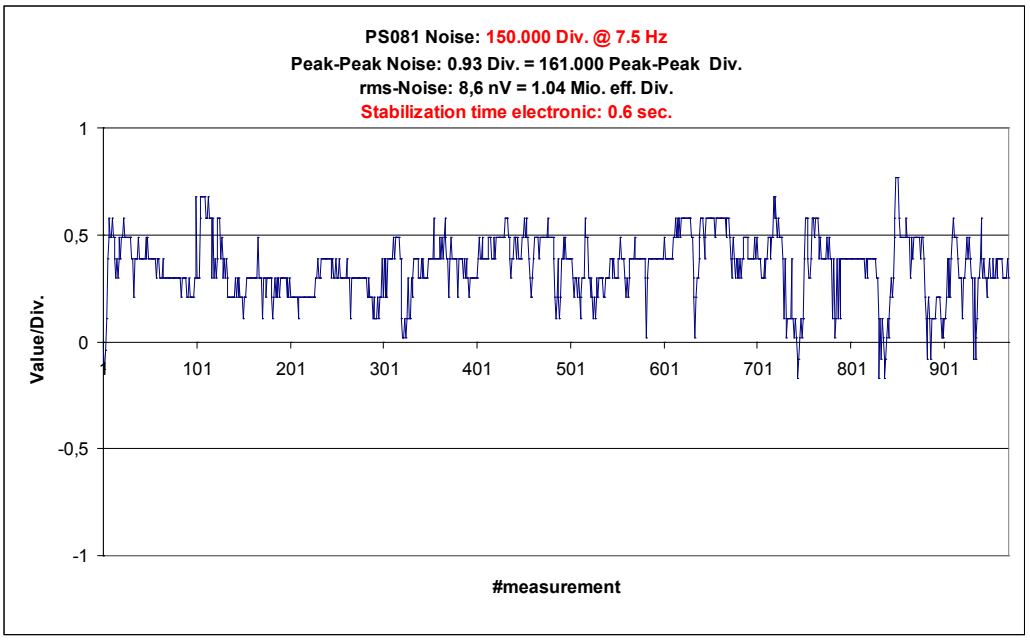
Divisions					
peak-peak	internal (eff.)	eff. Bits	Upate rate	Stabilization time	RMS-noise [nV]
250.000	1.370.000	20.3	5 Hz	1.0 sec.	6.5 nV
150.000	1.040.000	19.9	7.5 Hz	0.6 sec.	9 nV
100.000	680.000	19.3	12 Hz	0.35 sec.	13 nV
80.000	595.000	19.1	16.6 Hz	0.27 sec.	15 nV
60.000	385.000	18.5	30 Hz	0.15 sec.	24 nV

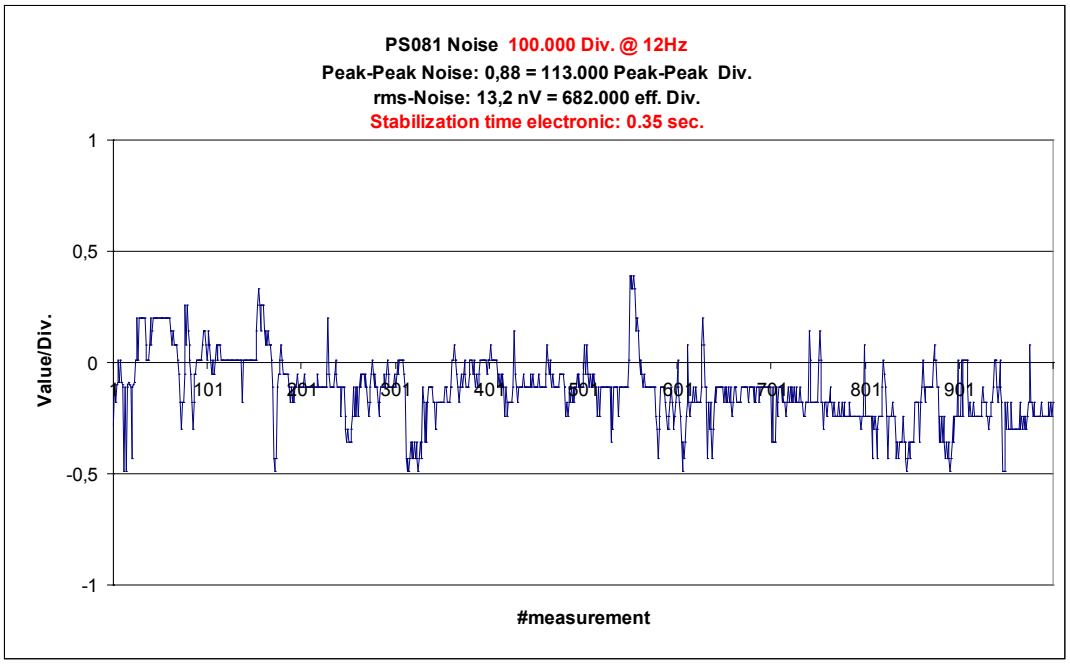
The following 3 noise diagrams show the noise behavior within 1 scale division. In the diagram title you can see the number of stable scale divisons set, in all cases the noise is deeply within 1 division.

Figure 8.1 Achievable Resolution

Annotations:

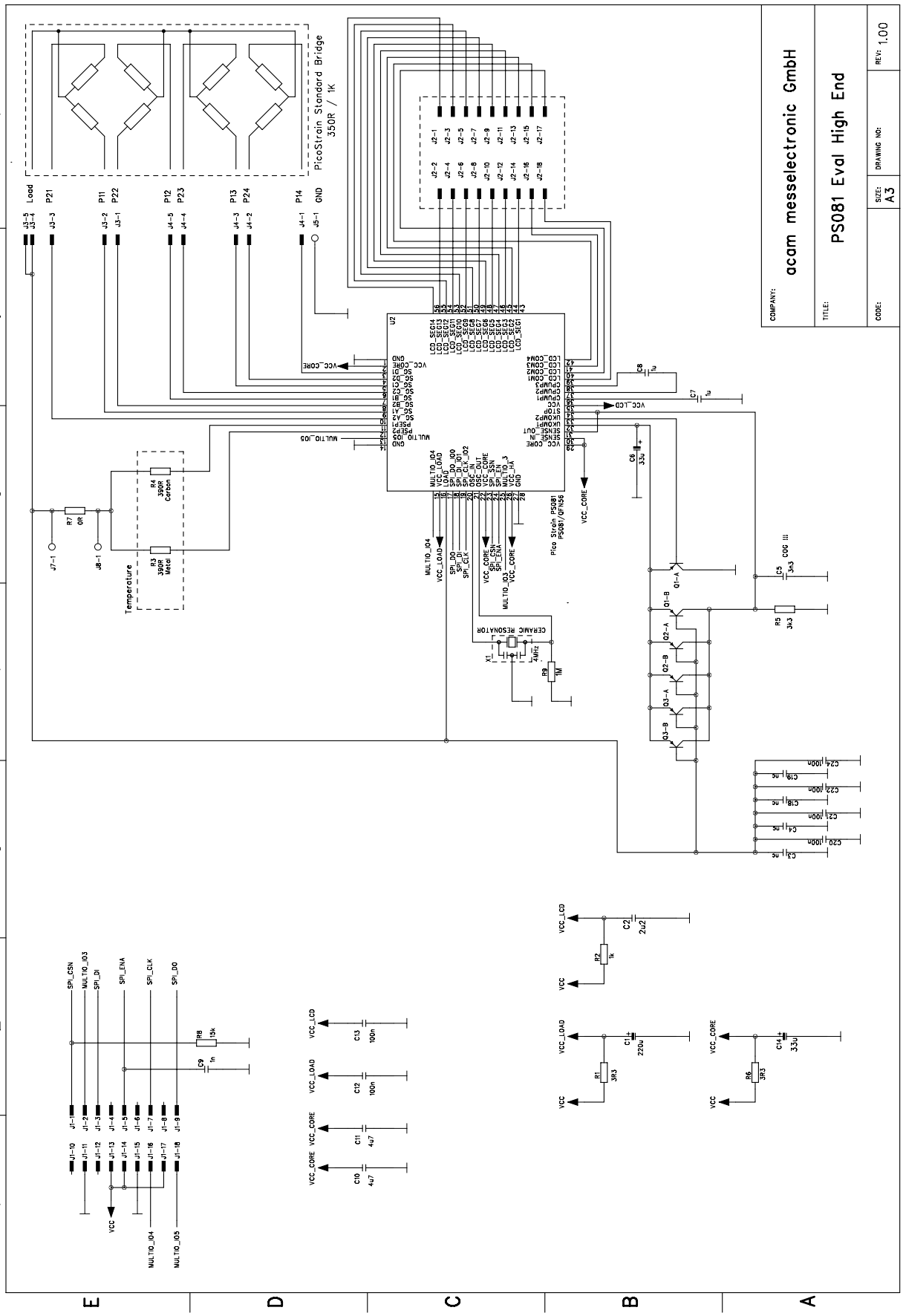






A high resolution as achieved with this set up requires always a good sensor, adapted software and hardware settings. Please contact the acam team in order to get more details when setting up such a high resolution application.

8.2 Schematics



8.3 Code snippet for external LCD driver

In chapter 4 the support of an external LCD driver is explained. The following code snippet shows basically how to program the SPI access to the external LCD driver. A simple measurement example will be available in the assembler software soon.

Configuration:

```
equal 0x480540          ; Config Register 64 ←
equal 0xE00700  ; Config Register 65 ←
equal 0x000000          ; Config Register 66 ←
```

Constant declarations:

```
CONST   spi_chip_select  9
CONST  spi_data_out              10
CONST  spi_clock         8
CONST   lcd_reg_low_read         98
CONST  lcd_reg_mid_read 99
CONST   spi_bus          65
```

Initialization of Holtek HT1621 (realized as subroutine):

```
init_holtek:
        ;set initial state
        ramadr 65
        or r, 0x000700          ;set pins lcd_spi_out to 1
        ;set duty cycle and number of common segments (Holtek)
        move y, 0x0004A1
        jsub send_command    ;LSB of y is MSB to Holtek!!!
        ;turn on display  (Holtek)
        move y, 0x000601
        jsub send_command
        ;turn on bias generator (Holtek)
        move y, 0x000401
        jsub send_command
        ;select crystal oscillator (Holtek)
        move y, 0x000141
        jsub send_command
        jsubret
```

```
        ;send command (fixed to 9+3 bits), supporting routine
send_command:
        ramadr 65
        bitinv r,spi_chip_select
        spi2lcd y, 0x00000C
        ramadr 65
        bitinv r,spi_chip_select
        jsubret
Sending data to Holtek, for ¼ MUX LCD:
;-------------------------------------------------------------------------
; 1/4 Multiplex Driver
;-------------------------------------------------------------------------
;1/4 duty
driver_4:
        ramadr spi_bus
        bitinv r, spi_chip_select
        move y, 0x000005   ;opcode send data for Holtek, set adr 0
        spi2lcd y, 0x000009
        ;first 24 bits
        ramadr lcd_reg_low_read      ;reg 98
        move     y, r
        spi2lcd y, 0x000018
        ;next 24 bits
        ramadr lcd_reg_mid_read  ;reg 99
        move     y, r
        spi2lcd y, 0x000018
        ;send last 8 bits, special chars
        ramadr lcd_reg_high
        move     y, r
        spi2lcd y, 0x000008
        ramadr spi_bus
        or r, 0x000700 ;set SPI wires back to high
        jsubret
```

Call the subroutine(s):

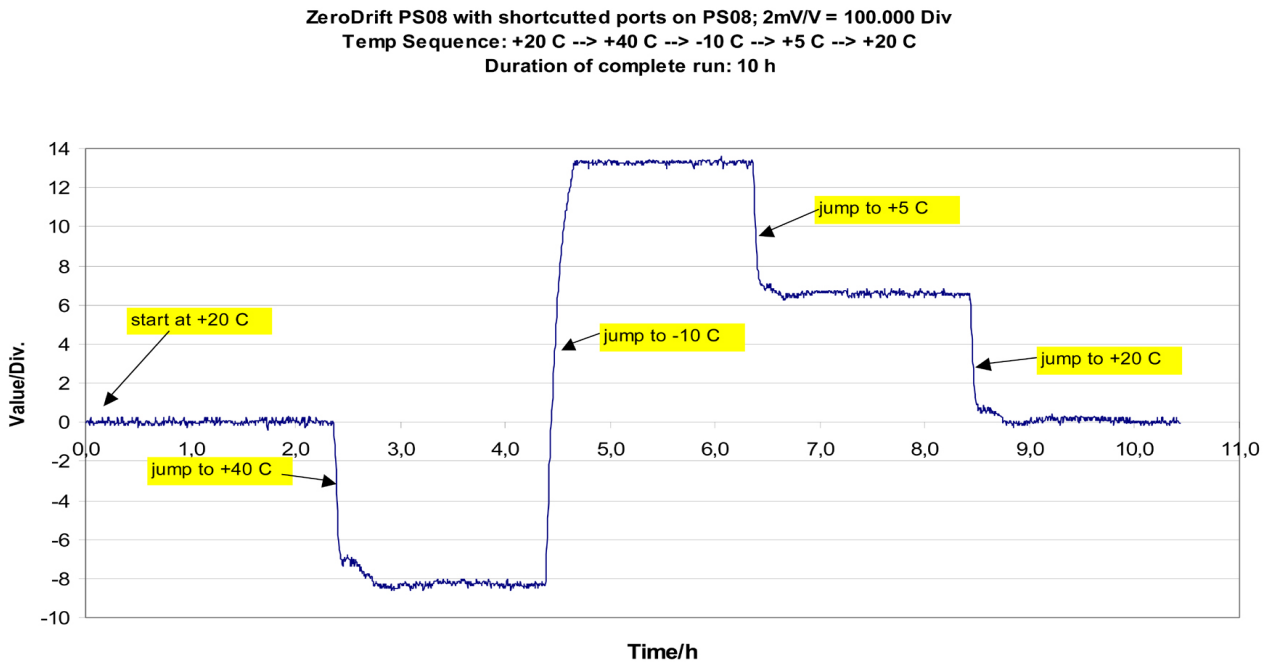
```
ramadr  20      ;HBO result
move    x, r
no2lcd  x, 1
;------- displaying on external LCD -------
newlcd
jsub            driver_4
```

Note:

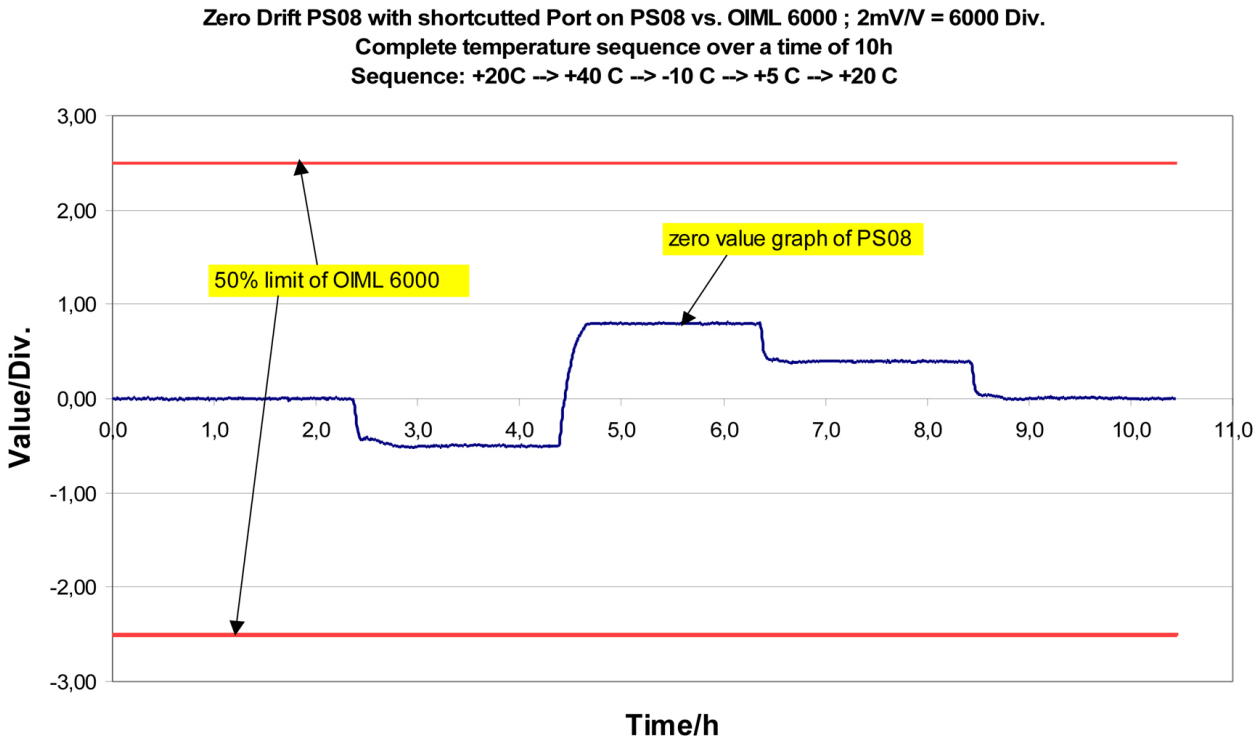
This code snippet show the relevant code to drive an external LCD via SPI. It's not a complete example. Please make sure that the LCD driver is connected properly (connect LCD pins in parallel as shown in chapter 4).

8.4 Long Term Offset-Drift Investigations

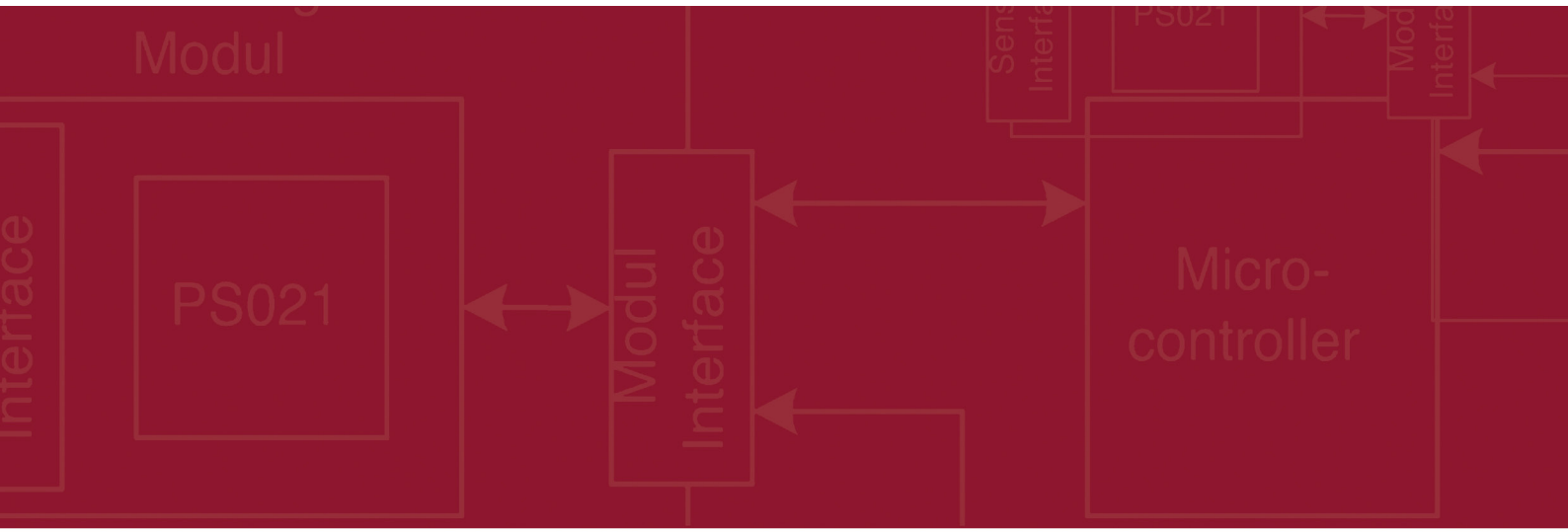
In August 2009 acam did some long-term offset drift investigations. In the setup the ports of the PS08 were short-cutted (by a resistor) and the board put into the temperature chamber. In a 10 hours run the temperature cylce +20°C -> +40°C -> -10°C -> +5°C -> +20°C was run. Please see in the following diagrams the results:



The offset drift is very low and lies deeply within OIML 6000 limits as shown in the following diagram:



PICOSTRAIN



acam-messelectronic gmbh
Friedrich-List-Strasse 4 ,
76297 Stutensee-Blankenloch
Germany / Allemagne
ph. +49 7244 7419 - 0
fax +49 7244 7419 - 29
e-mail: support@acam.de
www.acam.de