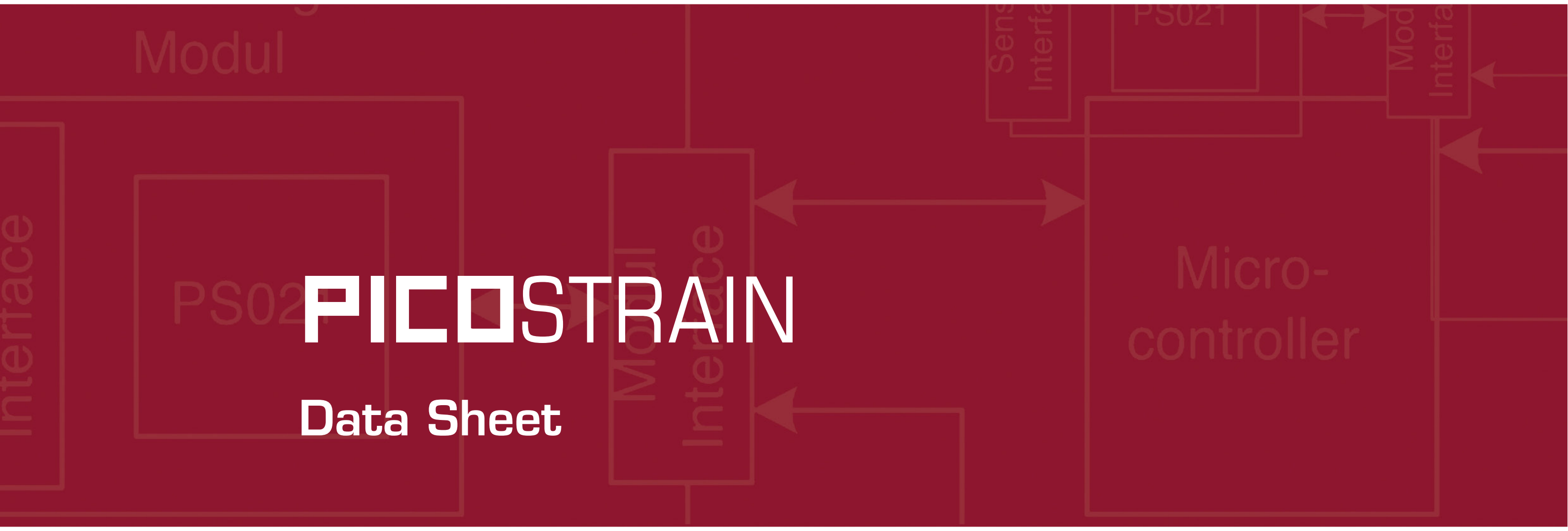


PICOSTRAIN

Data Sheet

PSØ9

Single Chip Solution for Strain Gauges

27th May 2011
Document-No.: DB_PSØ9_en V0.3

Published by acam-messelectronic gmbh

Disclaimer / Notes

„Preliminary“ product information describes a product which is not in full production so that full information about the product is not available yet. Therefore, acam messelectronic GmbH („acam“) reserves the right to modify this product without notice. 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

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

1 Overview

1.1 Features

- RMS noise: 31 nV fast settle, 5 Hz
  19 nV SINC3, 5 Hz
  11 nV SINC5, 5 Hz
- Upto 80,000 peak-peak divisions in weighing applications (2 mV/V strain) when operating in pure PS09 mode.
- Up to 120,000 peak-peak divisions in weighing applications (2 mV/V strain) when operating in PS081 mode at 4.5 V.
- Resolution: 28 bit ENOB (RMS) or 25.8 bit noise-free (peak-to-peak)
- Scalable update rate from < 1 Hz to 1000 Hz
- Current consumption:
  ~0.39 mA PSØ9 itself (at maximum speed)
  ~0.007 mA PSØ9 itself (at low current configuration)
  ~0.002 mA standby current
- Power supply voltage: 2.1 V to 3.6 V
- Converter type: Time-to-digital converter (TDC)
- 24-Bit internal microprocessor with 8 KB one-time programmable (OTP) ROM
- 4-wire serial SPI interface
- 2-wire serial I²C interface
- UART (RS232)
- 128 Byte User EEPROM
- 160x 24Bit RAM
- Interface to drive external LCD driver circuits
- 8 GPIOs pins, up to 24 inputs possible
- 4 capacitive inputs
- Analog switches integrated for direct driving of Wheatstone bridges
- Embedded temperature measurement and temperature compensation with < 0.01 °C accuracy (peak-to-peak)
- Very low gain and offset drift
- Embedded bandgap voltage reference
- Watchdog timer

1.2 Advantages

- Small and compact solution for weighing applications Converter and microcontroller in one chip
- Perfectly suited for building Digital Load Cells and Consumer Scales
- Extreme low total system current (down to 15µA including strain gages)
- Very low self heating of the sensor
- Gain and offset correction of the load cell

1.3 Packages

- Available as dice (1.98x1.7mm², 120 µm pitch, Pad opening 85x85µm²)
- Packaged (QFN40, 7x7 mm²) for development and mass production

1.4 Applications

Industrial

- Digital Load Cells
- 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

1.5 Application fields PS09/PS081

	PS09	PS081
Solar bathroom scale		X
Low cost bathroom scale	X	
Solar kitchen scale	X	X
Kitchen scale with cap.switches	X	
Digital load cell	X	
High end scales	X	X
(OIML) calibrated scales	X	X

1.6 General Description

The PSØ9 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 **PICO**STRAIN. Thus, it combines the performance of a 28-Bit signal converter with a 24-Bit microprocessor.

By connecting an external LCD driver and some very few external components you can build a complete weighing electronic. Overall a very compact design is feasible with PS09. Therefore it is extremely well suited for building Digital Load Cells (DLC), an unit where even no LCD driver is needed and the result is just given digitally to the outside world (by SPI, IIC or UART). The powerful and unique **PICO**STRAIN temperature compensation allows temperature adjustment of the sensor without mechanical trimming and simplifies production significantly. Again this feature is very helpful when it comes to DLCs, but also ordinary analog load cells can use this feature and improve quality thereby.

Another outstanding feature helping to build latest kitchen or portable scales are the capacitance based inputs. Using capacitance keys allow very flat scale designs and reflect the latest trend in the area of consumer products – and this at a current ad on of approx. 1µA which is much less in comparison with traditional solutions by an external driver.

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 1 million internal divisions (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 Σ-Δ-A/D converters, but at a much lower total current consumption and with integrated microprocessor.

Equipped with these features, a variety of scale electronics can be served with PSØ9. On the resolution side, it allows to build scales with up to 120,000 stable peak-peak divisions (at 2mV/V)! On the other hand, a sophisticated power management and the special features of the **PICO**STRAIN measuring principle can reduce the total current of the system down to 15 µA, including the sensor current. This way, with **PICO**STRAIN it is the 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 but low current scale such as a C3 legal for trade scale that runs more than 20,000 operating hours with 2x AA batteries.

1.7 Functional Block Diagram

Figure 1-1: PS09 block diagram

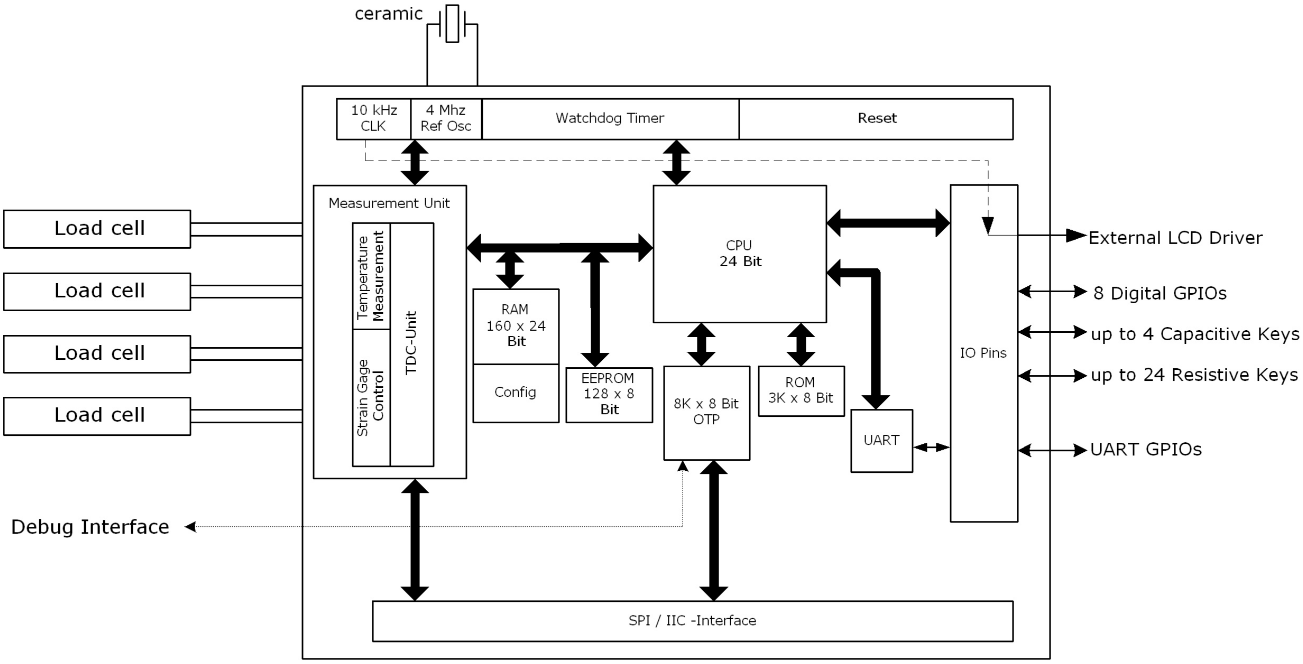


Table of Contents

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	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_RC	Supply voltage	Vcc vs. GND	2.1	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 of a weight scale (-10°C ... +40 °C)

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		400 280 225 150		mV
Voh	Output high voltage		0.8Vcc			V
Vol	Output low voltage				0.2Vcc	V
Vlbat	Low battery voltage detect		2.2		2.9	V
Iq	Quiescent current	No oscillator, no TDC		1.5		µA
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 and external oscillator

	ENOB dR/R strain resistance		
Frequency (Hz)	**No filter**	**SINC3**	**SINC5**
500	23.3	24.3	24.7
250	23.9	24.7	25.2
100	24.7	25.3	25.6
50	25.0	25.7	26.0
20	25.5	26.3	26.5
10	26.1	26.9	27.2
5	26.7	27.4	27.7

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

	Resolution @ **2 mV/V** max. out, **Fast settle[1]**			
Frequency (Hz)	**ENOB**	**Divisions**	**Noise nV**	**Noise nV**
		effective	rms	peak-peak[2]
500	14.3	20000	327	1963
250	14.9	31000	216	1295
100	15.7	53000	124	744
50	16.0	66000	101	604
20	16.5	93000	71	427
10	17.1	140000	47	282
5	17.7	213000	31	186

** Fast settle = without filter*

Table 2.6: Performance at Vcc = 3.3V with **external** comparator and external oscillator, 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[2]		effective	rms	peak-peak[2]
500	15.3	40000	164	982	15.7	53000	124	744
250	15.7	53000	124	744	16.2	75000	88	526
100	16.3	81000	82	491	16.6	99000	66	399
50	16.7	106000	62	372	17.0	131000	50	302
20	17.3	161000	41	245	17.5	185000	36	214
10	17.9	245000	27	162	18.2	301000	22	132
5	18.4	346000	19	114	18.8	456000	14	100

[1] Fast settle = without filter

[2] Peak-to-peak data in PSØ81-compatible mode. In PSØ9 mode the peak-to-peak data will be higher

Table 2.7: Performance at Vcc = 3.3V with **internal** comparator and internal RC oscillator

	ENOB dR/R strain resistance			**ENOB 2mV/V, Fast settle***
Frequency (Hz)	**No filter**	**SINC3**	**SINC5**	
500	22.5	23.5	23.9	13.5
250	23.1	23.9	24.6	14.1
100	23.9	24.5	24.8	14.9
50	24.2	24.9	25.2	15.2
20	24.7	25.5	25.7	15.7
10	25.3	26.1	26.4	16.3
5	25.9	26.6	27.0	16.9

* 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	95 @1.8V	115 @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 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	20 µA 60 µA	Postal, Body, Kitchen , Pocket	3,000 hours (1xCR2032)
5,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	40 µA 120 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, Counting	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)*

2.6 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 2.1: SPI - Write access

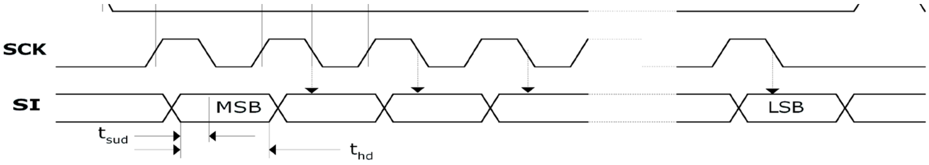
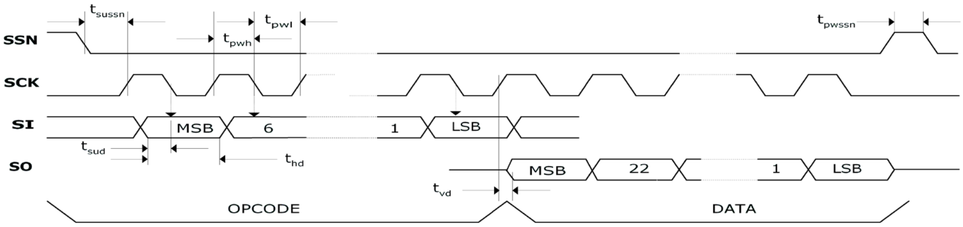


Figure 2.2: SPI-Read access



2.7 Pin Assignment

PSØ9 is available as Die or in QFN40 package. The following pictures and tables show the pin assignment and description.

Remark: The terms "multiple purpose input/output" (Mult_IO) or "GPIO" or simply "I/O" are used interchangeably.

QFN40

Figure 2.3: Pin assignment QFN 40

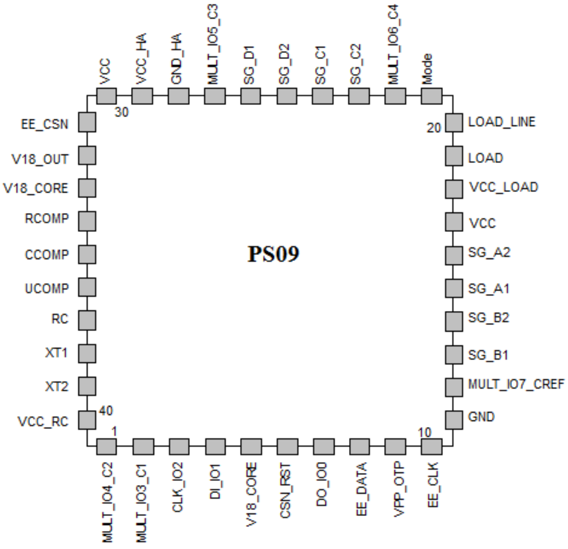
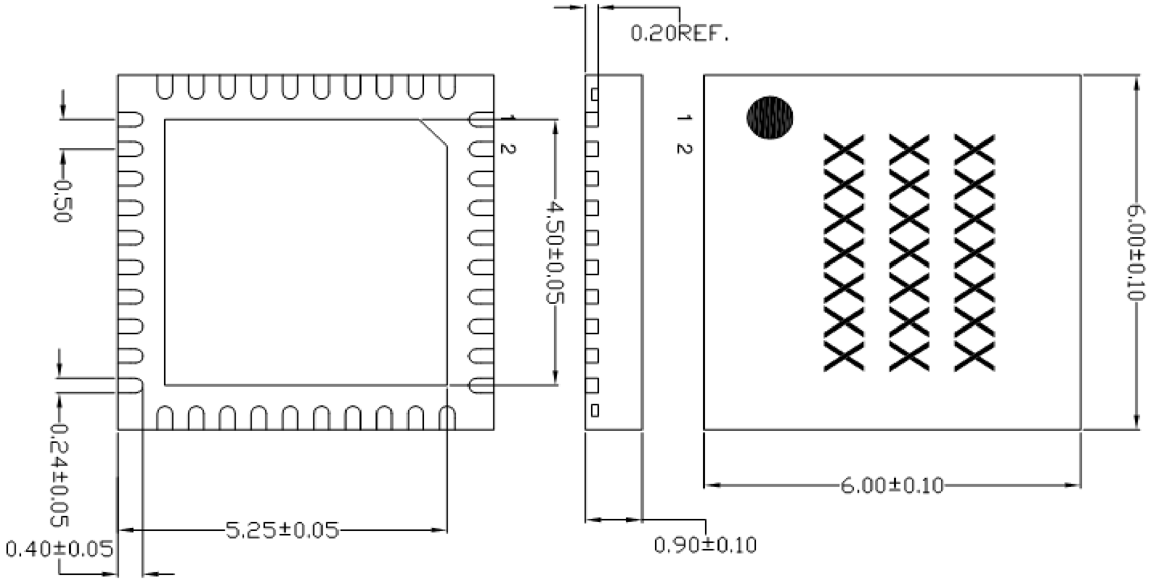


Table 2.12: Pin Description QFN 40

#QFN	Name	Description	Type
1	MULT_IO4_C2	Multiple purpose input/output #4 or capacitive input #2	Digital IN/OUT
2	MULT_IO3_C1	Multiple purpose input/output #3 or capacitive input #1	Digital IN/OUT
3	CLK_IO2	SPI or IIC clock or multiple purpose input/output #2	Digital IN/OUT
4	DI_IO1	SPI data in (MOSI) or multiple purpose input/output #1	Digital IN/OUT
5	V18_CORE	Supply voltage of digital core	
6	CSN_RST	SPI chip select (SSN) / IIC reset	Digital IN/OUT
7	DO_IO0	SPI data out (MISO) or IIC data in/out or multiple purpose input/output #0	Digital IN/OUT
8	EE_DATA	EEPROM data (external program developmenr EEPROM)	Digital IN/OUT
9	VPP_OTP	Programming voltage OTP	
10	EE_CLK	EEPROM clock (external program development EEPROM)	Digital IN/OUT
11	GND	Ground	
12	MULT_IO7_CREF	Reference capacitor for capacitive switches or multiple purpose input/output #7	Digital IN/OUT
13	SG_B1	Strain gage port 1, half bridge B	SG - Port Driver
14	SG_B2	Strain gage port 2, half bridge B	SG - Port Driver

#QFN	Name	Description	Type
15	SG_A1	Strain gage port 1, half bridge A	SG - Port Driver
16	SG_A2	Strain gage port 1, half bridge A	SG - Port Driver
17	VCC	Power supply voltage	
18	VCC_LOAD	Power supply of LOAD pin	
19	LOAD	LOAD pin to connect load capacitor	
20	LOAD_LINE	connect to LOAD	
21	MODE	Select between SPI, IIC interface or stand alone mode	Analog Input
22	MULT_IO6_C4	Multiple purpose input/output #6 or capacitive input #4	Digital IN/OUT
23	SG_C2	Strain gage port 2, half bridge C	SG - Port Driver
24	SG_C1	Strain gage port 1, half bridge C	SG - Port Driver
25	SG_D2	Strain gage port 2, half bridge D	SG - Port Driver
26	SG_D1	Strain gage port 2, half bridge D	SG - Port Driver
27	MULT_IO5_C3	Multiple purpose input/output #5 or capacitive input #3	Digital IN/OUT
28	GND_HA	Ground	
29	VCC_HA	Power supply TDC	
30	VCC	Power supply voltage	
31	EE_CSN	EEPROM chip select	Digital IN/OUT
32	V18_OUT	Regulated 1.8V voltage	Analog Output
33	V18_CORE	Supply voltage of digital core	
34	RCOMP	Comparator resistor	Analog IN/OUT
35	CCOMP	Comparator capacitor	Analog IN/OUT
36	UCOMP	Threshold voltage comparator	Analog Output
37	RC	RC oscillator	Analog Input
38	XT1	Ceramic oscillator	Analog output
39	XT2	Ceramic oscillator	Analog input
40	VCC_RC	Power supply RC oscillator	

Figure 2.4: QFN40 Dimensions



Die

Figure 2.5: Pad assignment PS09 dice

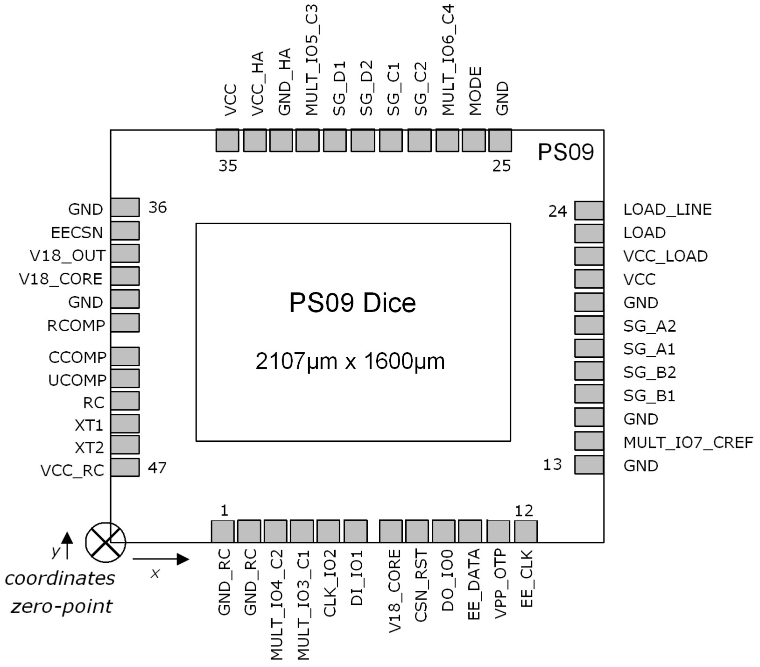


Table 2.14 Pad description PS09 dice

Pad	Name	X-Pos(µm)	Y-Pos(µm)	Type
1	GNDRC	442	45	Ground
2	GNDRCa	593	45	Ground
3	MULT_IO4_C2	742	45	Digital IN/OUT
4	MULT_IO3_C1	862	45	Digital IN/OUT
5	CLK_IO2	982	45	Digital IN/OUT
6	DI_IO1	1102	45	Digital IN/OUT
7	V18_CORE	1222	45	Supply voltage of digital core
8	CSN_RST	1342	45	Digital IN/OUT
9	DO_IO0	1462	45	Digital IN/OUT
10	EE_DATA	1582	45	Digital IN/OUT
11	VPP_OTP	1702	45	Programming voltage OTP
12	EE_CLK	1821	45	Digital IN/OUT
13	GND	2058	130	Ground
14	MULT_IO7_CREF	2058	240	Digital IN/OUT
15	GND	2058	360	Ground
16	SG_B1	2058	482	SG - Port Driver
17	SG_B2	2058	602	SG - Port Driver
18	SG_A1	2058	723	SG - Port Driver
19	SG_A2	2058	843	SG - Port Driver
20	GND	2058	963	Ground
21	VCC	2058	1080	Supply voltage
22	VCC_LOAD	2058	1200	Supply voltage

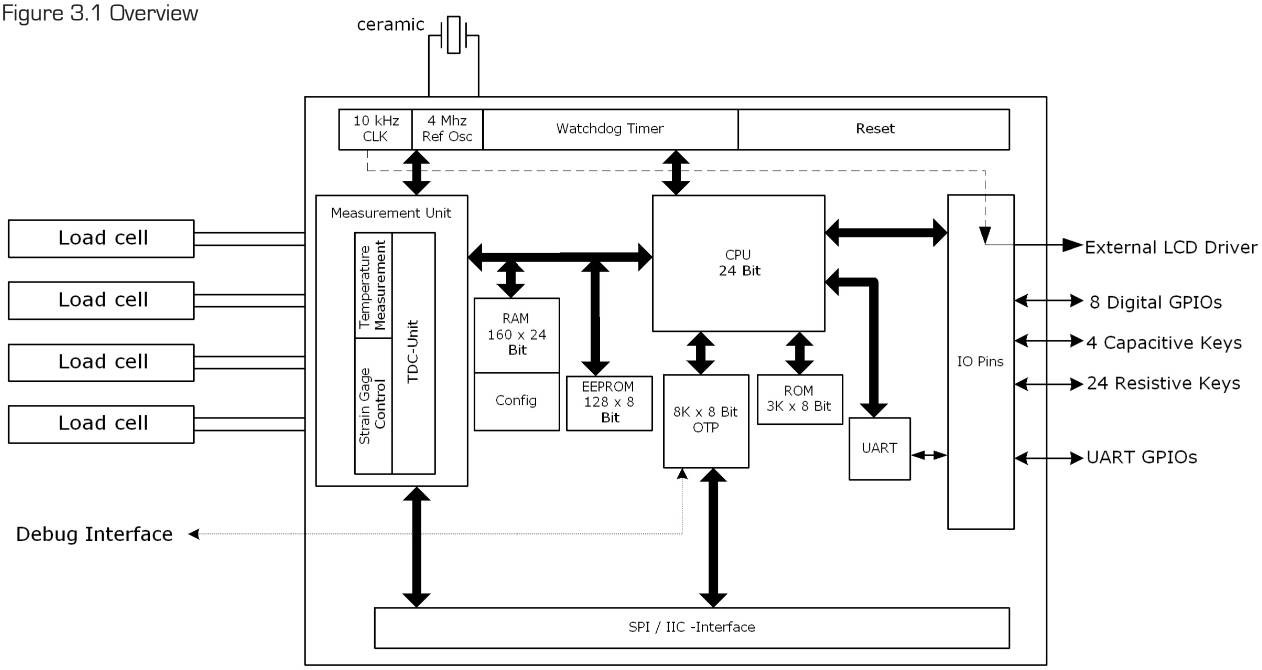
Pad	Name	X-Pos(µm)	Y-Pos(µm)	Type
23	LOAD	2058	1320	
24	LOAD_LINE	2058	1434	
25	GND	1936	1556	Ground
26	MODE	1822	1556	Digital IN/OUT
27	MULT_IO6_C4	1702	1556	Digital IN/OUT
28	SG_C2	1448	1556	SG - Port Driver
29	SG_C1	1328	1556	SG - Port Driver
40	SG_D2	1207	1556	SG - Port Driver
31	SG_D1	1087	1556	SG - Port Driver
32	MULT_IO5_C3	672	1556	Digital IN/OUT
33	GND_HA	527	1556	Ground Hardmacro
34	VCC_HA	407	1556	Supply voltage
35	VCC	279	1556	Supply voltage
36	GND	45	1454	Ground
37	EE_CSN	45	1320	Digital IN/OUT
38	V18_OUT	45	1200	Analog Output
39	V18_CORE	45	1080	Supply voltage digital core
40	GND	45	960	Ground
41	RCOMP	45	840	Analog IN/OUT
42	CCOMP	45	720	Analog IN/OUT
43	UCOMP	45	600	Analog IN/OUT
44	RC	45	480	Analog Input
45	XT1	45	360	Analog Output
46	XT2	45	240	Analog Input
47	VCC_RC	45	126	Power supply RC oscillator

Table of Contents Page

3 Converter Front End

3.1 Overview

Figure 3.1 Overview



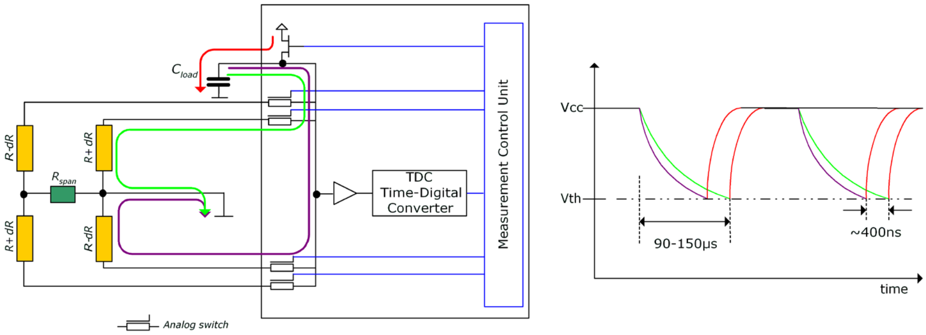
The PICOSTRAIN based converter has the strain gage ports (SG_Ax to SG_Dx) to measure:

- 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



NEW with PS09 Please note, that the control of the strain gage resistors was changed from PS081 to PS09. By means of internal analog switch the middle connection of the bridge is now directly to GND. So the way of how the resistors are discharged has changed. This saves one wire to the load cell and additionally improve EMI behavior of the sytem

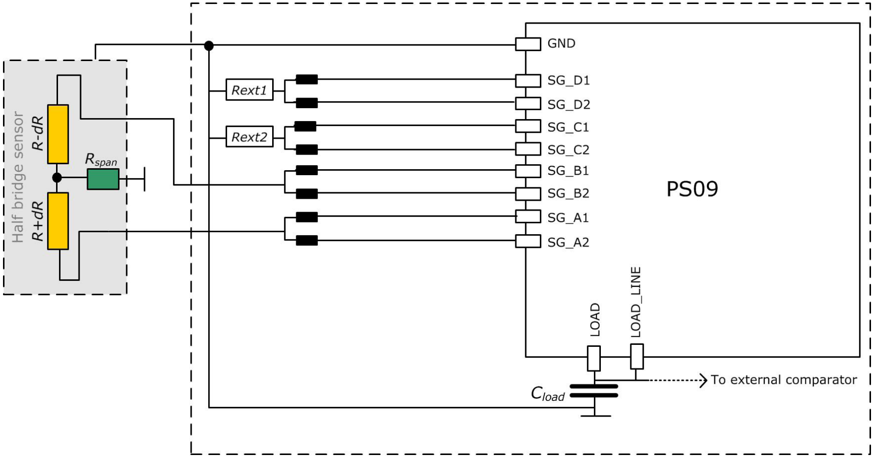
This chapter will explain the components of the front-end, the parameters to set and how to dimension external components.There are many ways to connect your strain gage sensor to PSØ9. In this section we show how to connect them.

3.3 Connecting the Strain Gauges

Caution: For best results it is recommended to generally connect the load cell body to GND of the electronic. A simple standard wire is sufficient.

3.3.1 Half Bridge

Figure 3.3 Classical halfbridge connection

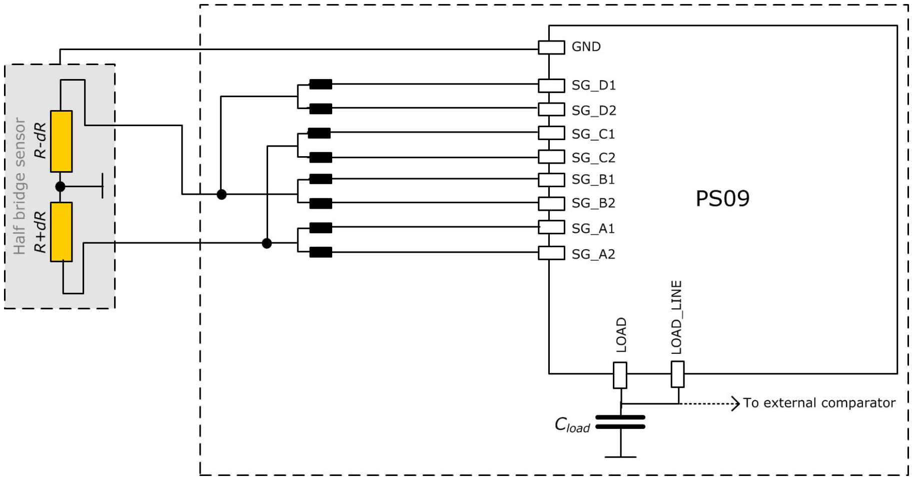


Note: The half bridge is connected to port A and B, while both pins of one port are used, as illustrated in the picture.

This requires the bridge setting = 0 (register 3, bridge[1:0] = 0)
The multiplication factors should have positive sign, e.g. Mult_Hb1 = +1, Mult_Hb2 = +1.

This classical half bridge connection in combination with a low average rate (AVRate = 2) allows maximum speed where up to 1 kHz can be reached. This connection also permits enabling the special EMI suppression transistors. (Section 3.7).
The external resistor values at port SG_C and SG_D must be the same as the strain gage resistor values. Special attention has to be be paid to the temperature coefficient of these resistors when temperature measurement is to be done in this configuration. (Refer Section 3.5.10)

FIgure 3.4: Alternative connection of a halfbridge



Alternate to the classical half bridge connection shown in figure 3.3, the external resistors can be totally eliminated and the unused ports can be connected to the existing half bridge, as shown in the figure. However, this configuration is not useful when high measurement speeds are required. Additionally the EMV protection transistor at the ports (See Section 3.7) must not be switched on in this connection.

NOTE : In this connection en_emi_noise_reduction = 0 in Config Reg 12.

3.3.2 Full Bridge

Figure 3.5: Connecting a full bridge with one Rspan resistor

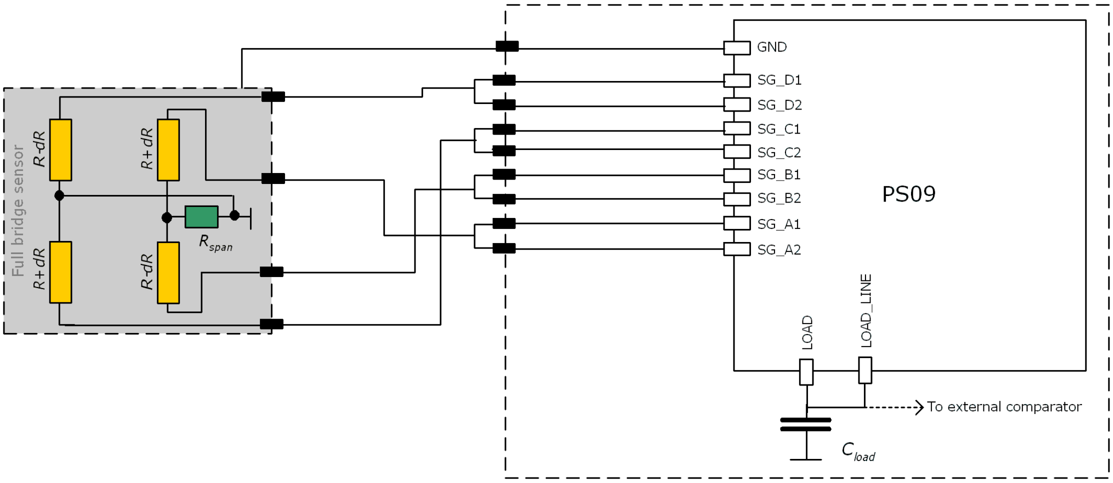
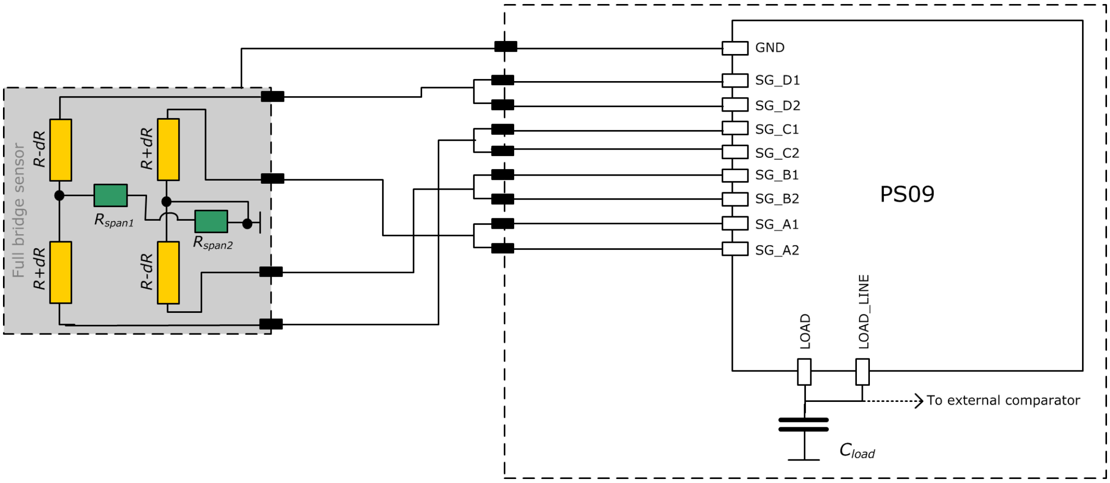


Figure 3.6: Connecting a fullbridge with two Rspan resistors

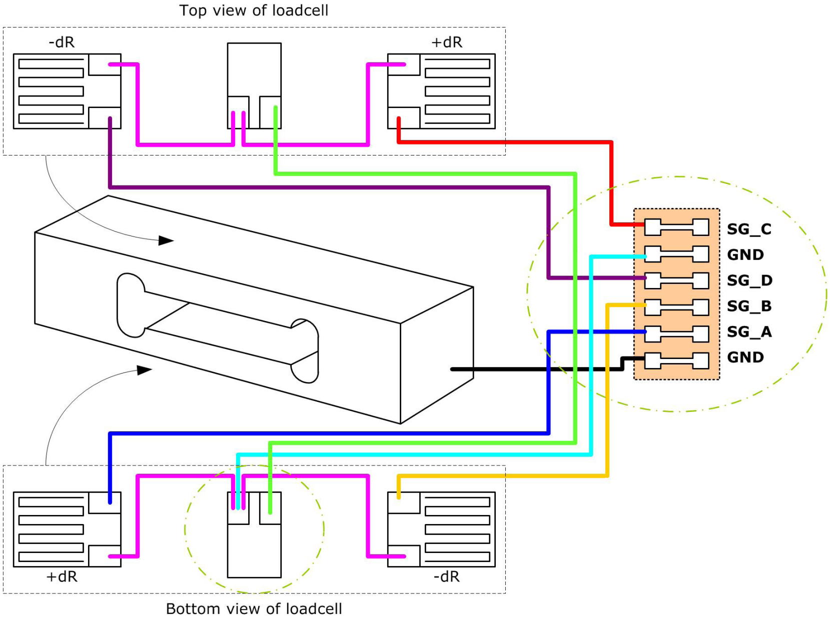


Note:

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). If the full bridge has two Rspan resistors they should be switched in series. Then the value for TKgain has to be doubled compared to bridges with one Rspan.
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:

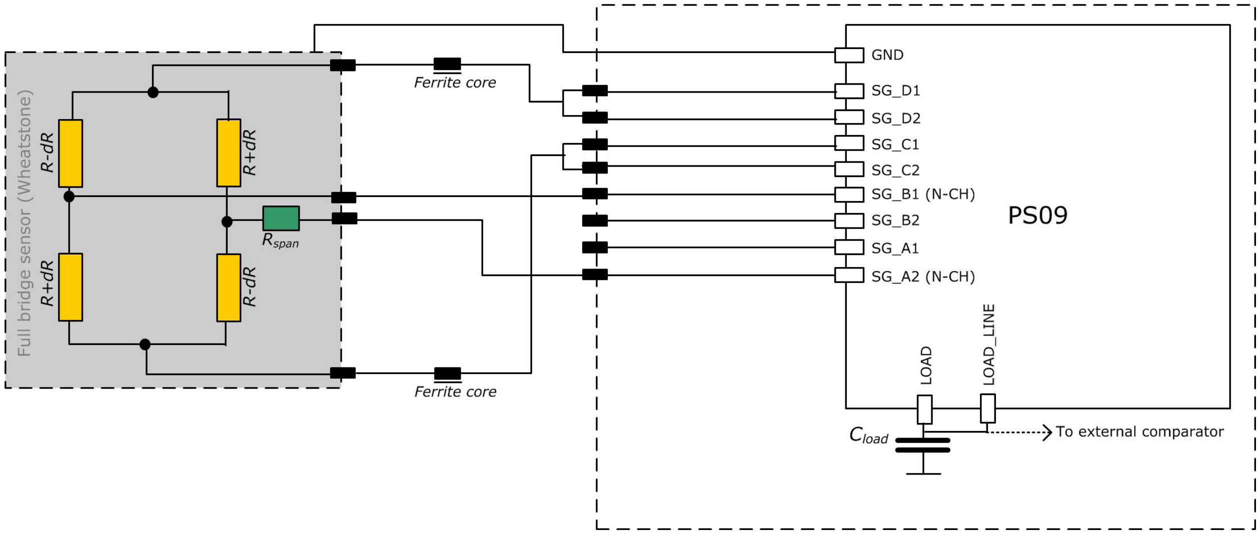
Figure 3.7: Adapted load cell wiring



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 Wheatstone Bridge

Figure 3.8: Connecting a Wheastone bridge



NEW with PS09 Unlike with PS081, the PS09 doesn‘t need an external analog switch to connect the Wheatstone bridge. Here the integrated analog switch saves the external component now.

Note: In Wheatstone mode the system looses 0.6 bit of resolution because of the reduced strain. Because of this, we recommend to use Wheatstone connection only in 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.
To enable Wheatstone mode please set en_wheatstone to 1 (register 3, bit 21).

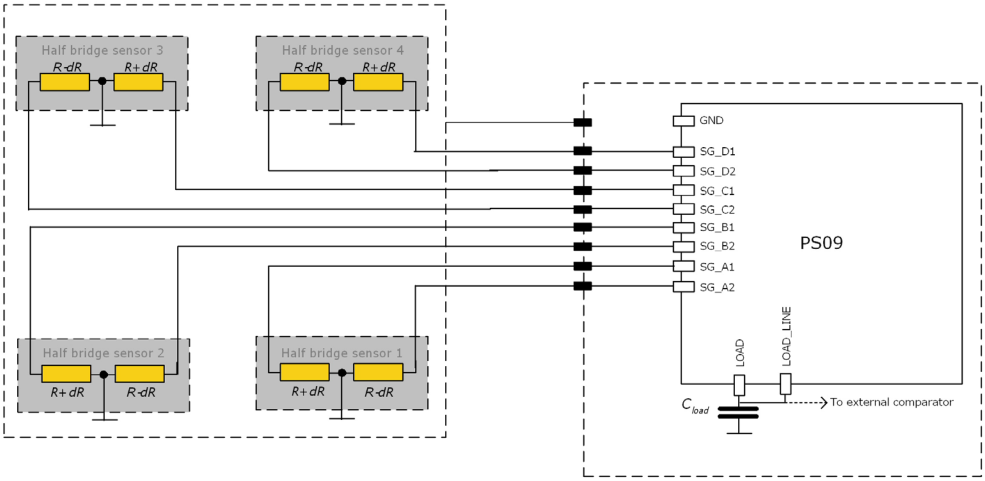
If the Wheatstone bridge has a gain compensation resistor (Rspan) the standard setting for TKGain is 0.75 (Configreg_08). The factor 0.75 doesn’t modify the natural span compensation behavior of the load cell. In any case “mod_rspan” has to be set to 1 (Configreg_01, Bit 6).
To avoid reflections in the Wheatstone bridge we recommend the use of ferrite cores. They are placed in the two lines which are connected directly to PS09. Ordinary (SMD-)ferrite cores with a damping of 100Ohm @ 100MHz with a low DC resistance (<0.1Ohm) 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.

Caution: Wheatstone bridges in combination wit Continuous mode and IIC read out show increased noise. See the bug report at the end of this datasheet.

3.3.4 Quattro Bridge (4 sensors)

Figure 3.9: Connecting a quattro bridge



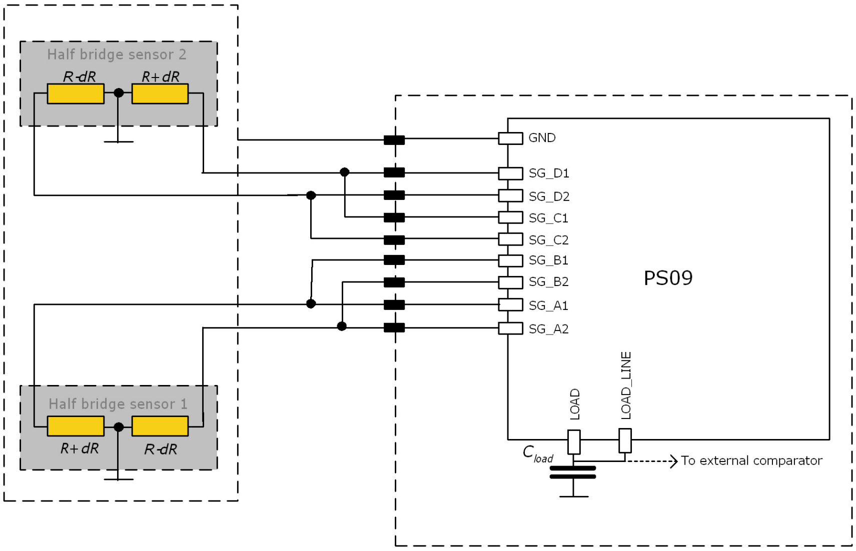
In some cases four sensors are used. Then each half bridge is connected to one port. This is a typical connection e.g. for quattro body scales. The result of each half bridge can be read but also the overall result.

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.

2 Half Bridges separately

Figure 3.10: 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.

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.

3.3.5 PS081 Compatible Mode

As seen in section 3.2, the principle of controlling the strain gage resistors in PS09 is different from the earlier PS081 chip. Analog switches are used to control the ports in PS09 instead of the N-channel transistors of PS081. Hence in PS09, the discharge path during the measurement cycle is through the analog switch which leads to very good EMI behavior in PS09 mode (please see chapter 3.7 for more details).
However, the measurement mode of PS081 is also supported in PS09. This PS081 compatible mode has some distinct advantages and disadvantages which are outlined in this section.
To support the PS081 compatible mode in the PS09 chip, the following changes need to be made in the PS09 connection:

1. Connect the LOAD_LINE pin of the PS09 chip directly to Ground
2. Connect the strain gage as in PS081 with the middle tap of the bridge(s) connected to the LOAD pin.

No other configuration changes are necessary. PS09 automatically detects the different connection and switches to the PS081 compatible mode. This means that internally only the N-Channel transistors in the analog switches in PS09 are activated so that the discharging is principally performed the same way like in PS081.

Note: The PS081 compatible mode is supported for Half Bridge, Full Bridge and Quattro modes only.

Wheatstone Mode is not supported in PS081 compatible mode!

The figure 3.11 shows the connections for a full bridge using PS09 in the PS081 compatible mode.

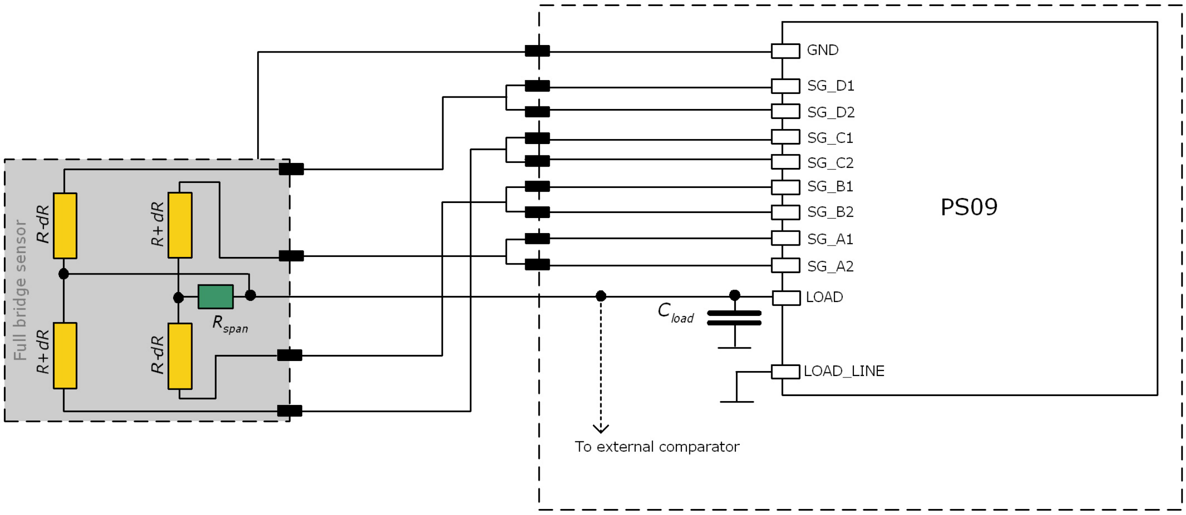
Once this connection is performed, the measurement mode of the PS081 is automatically adopted.

Advantages of PS081 compatible mode:

The PS081 control mode has some very clear advantages over the PS09 control mode.

1. Resolution: is higher than the PS09 mode by about 0.5 Bit.

Figure 3.11 PS09 in PS081 compatible mode with full bridge



2. Power Supply Rejection Ration (PSRR): is about 25 dB higher than in PS09 mode.
3. Gain drift over supply voltage is better by a factor of 5 compared to PS09 mode.
4. Offset drift is about 2-3 times better in comparison to the PS09 mode.
5. Maintains good measurement quality at voltages lower than 2.7V, e.g. in battery driven applications without a voltage regulator.

This mode can be employed in high quality weighing scales without the need for any voltage stabilization as for example in battery driven scales.

The strength of the PS081 compatible mode lies in the absence of the analog switch in the discharge path which is present in PS09. The analog switches in PS09 mode introduce some additional noise – which is avoided in the PS081 compatible mode by using only the N-channel transistors. This explains why the measurement quality in PS081 mode is relatively better than the PS09 mode.

Disadvantage of the PS081 compatible mode:

The shortcoming of the PS081 compatible mode is in its EMI behaviour at 50 Hz, mainly in applications with long unshielded cables, like for quattro scales. This 50 Hz EMI disturbance is strongly suppressed in the PS09 mode of operation. In PS09 mode, due to the use of the analog switches, there is a possibility to disconnect the unused strain gage ports internally from the measurement path. This can be configured by setting the en_emi_noise_reduction bit (Bit 7) in Configreg_12. All the strain gage ports that are not being measured in the current measurement cycle are connected internally to ground and thus disconnected from the measurement path. This is mainly advantageous for connections with long unshielded paths, typical to Quattro scales.

Note: For Strain gages with high resistance like for e.g. 10 kOhm, the 50 Hz EMI noise is predominant, thus affecting measurement quality drastically. For such cases, the PS09 mode must be used in order to perform measurement of acceptable quality.

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 ≈ 70 µs – 150 µs

For Wheatstone wiring the effective strain is reduced (see chapoter 3.4.6) and therfore Cload needs to be adapted accordingly:

Rsg = 350R → Cload = 400nF

Rsg = 1000R → Cload = 150nF to 220nF

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

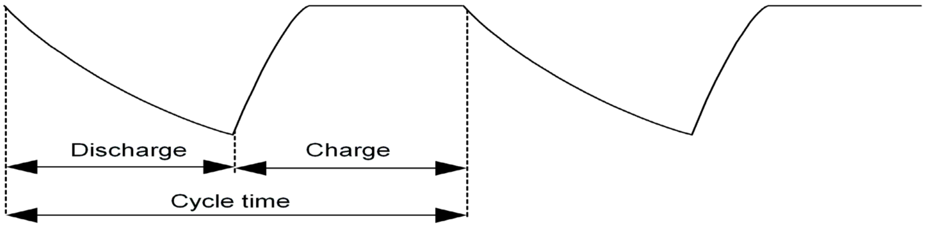
Note:

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 90 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.

Clock source for Cycle Time

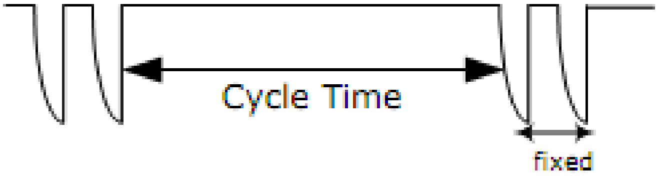
The cycle time is derived from the high speed clock which is a 4MHz ceramic resonator connected externally to PS09.

NEW with PS09 Furthermore PS09 offers in contrast to PS081 the possibility to use an internal RC oscillator as alternative clock source. This spares the external ceramic resonator therefore. However, as this internal clock source is not as accurate as an external component, the resolution decreases about 1.2 Bit. According to that the max. possible resolution is is limited to approx. 15.3 bit (3.3V, external comparator, 2Hz, median 5) when using the internal RC osciallator. For further details on how to use the internal RC oscillator please see chapter 4.1 Osciallators.

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



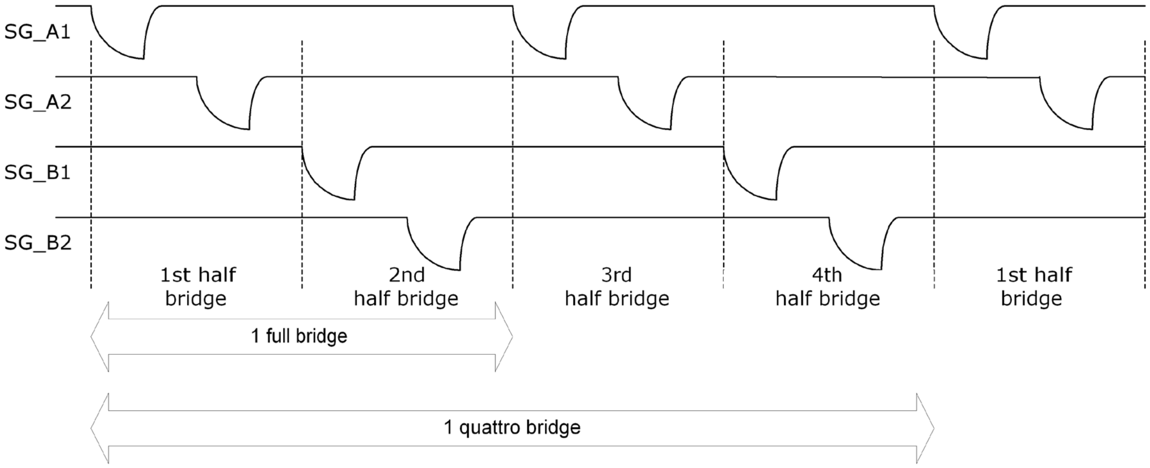
There are several parameters to adjust in stretched mode. Please have a look at Stretched Mode settings in section 3.5.3.

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

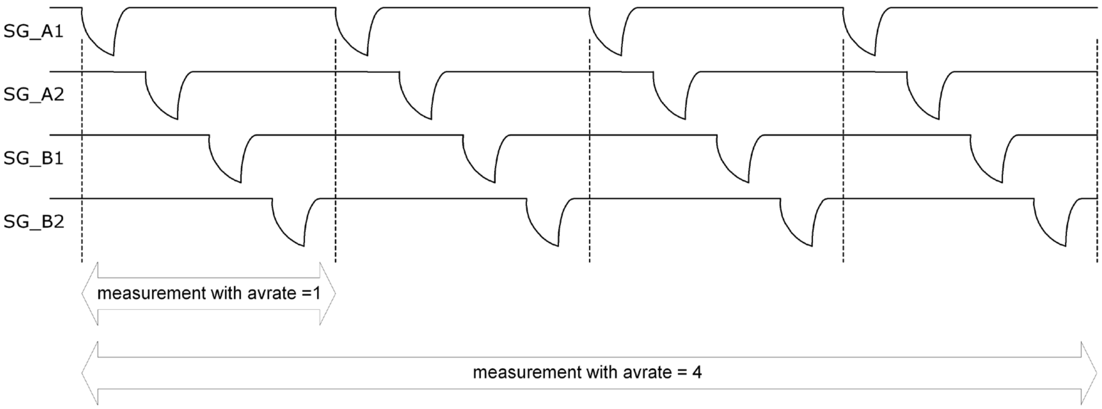


Those numbers of cycles for each mode together define 1 sample (avrate=1). This is also the minimum needed for one complete ratio measurement.

3.4.5 Better resolution by averaging

In PS09, 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 ps_tdc1_adjust (Configreg_03, Bit [9:4]) should be set to 2x avrate. Example: avrate = 8 –> set ps_tdc1_adjust 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 HB0 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 (0x800001) to +8388607 (0x7FFFFF). One LSB has thereby the valency of 10 nV/V (2 mV/V divided by 200,000).
  Example: at 3 V the valency of 1 LSB is 10 nV/V x 3 V = 30 nV.
- 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 PS09 an RMS noise as low as 11 nV at 3.3 V can be achieved, or expressed in ENOB up to19.5 Bits (related to 2 mV/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:

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

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

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

Please 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 **PICO**STRAIN 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

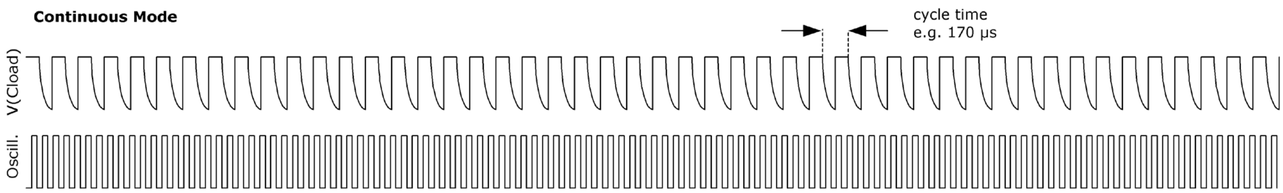
The PS09 has 3 basic operating modes as well as combinations of them. They are related to the sampling frequency and the active time of the 4 MHz oscillator. Therefore, the selection has influence on the stability of the result and the current consumption.

The basic modes are:

- Continuous Mode
- Single Conversion Mode
- Stretched Mode

3.5.1 Continuous Mode

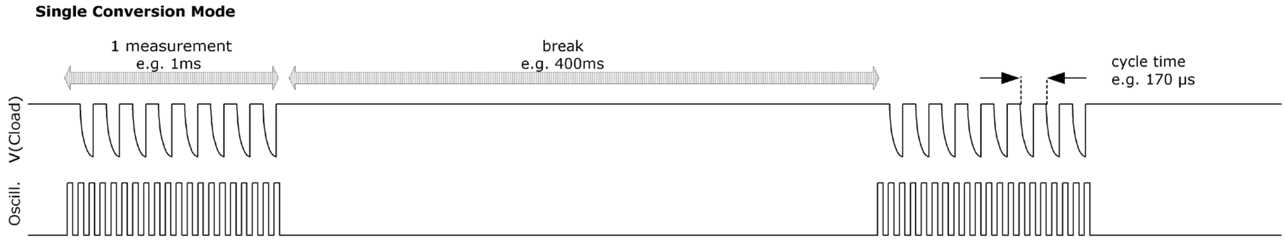
Figure 3.16: Continuous mode



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.

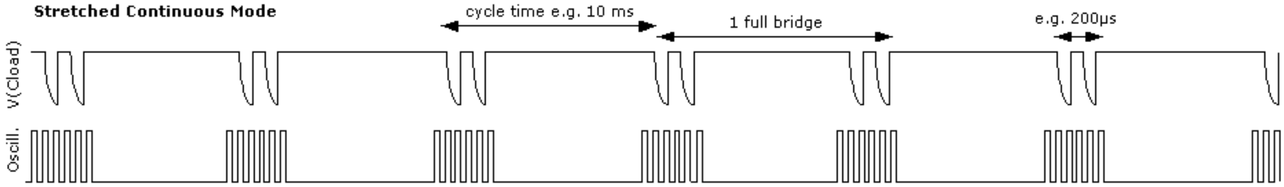
Pure Single Conversion Mode should be used only in mechanically stable systems like body scales, because it implies undersampling. The consequence of undersampling is that mechanical oscillations of the weighing system will end up in unstable data.

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

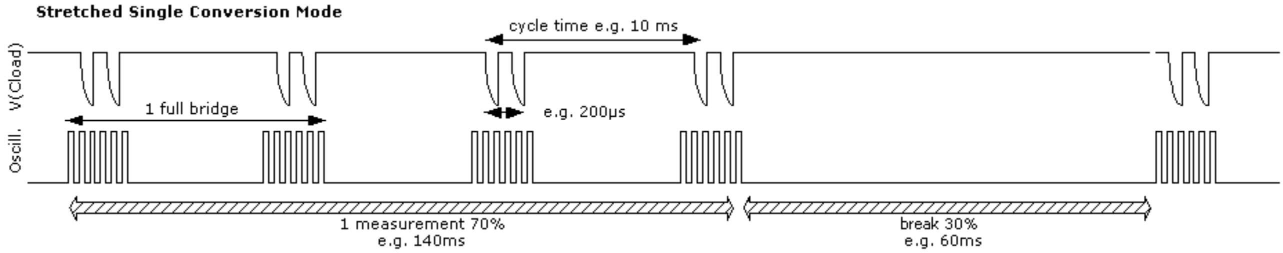


Stretched Conitnuous Mode combines stretched mode and continuous mode. There are longer intervals between the discharge time measurements for the half bridges. The oscillator is activated only for each half bridge measurement.

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.

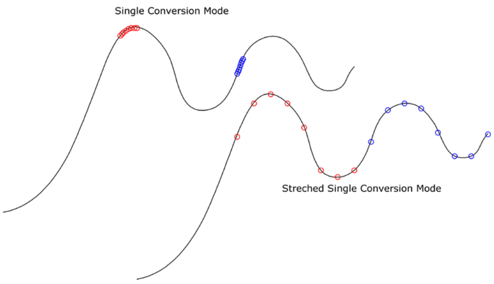
3.5.3.2 Stretched Single Conversion Mode

Figure 3.19: Stretched single conversion mode



For mechanically sensitive weigh scales like kitchen scales the PS09 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.

3.5.4 Configuration of Modes

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 (recom mended 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 measure ments that make a single measurement sequence (internal averaging).

3.5.4.1 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.

3.5.4.2 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 measurement. 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 measurement (two discharge time measurements). The cycle time (time between subsequent 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

3.5.4.3 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

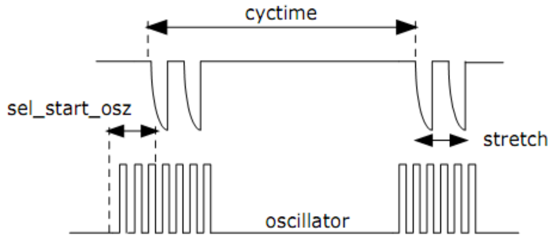
sel_start_osz[19:17] in Configreg_03

single_conversion [2] in Configreg_02

tdc_conv_cnt[23:16] in Configreg_00

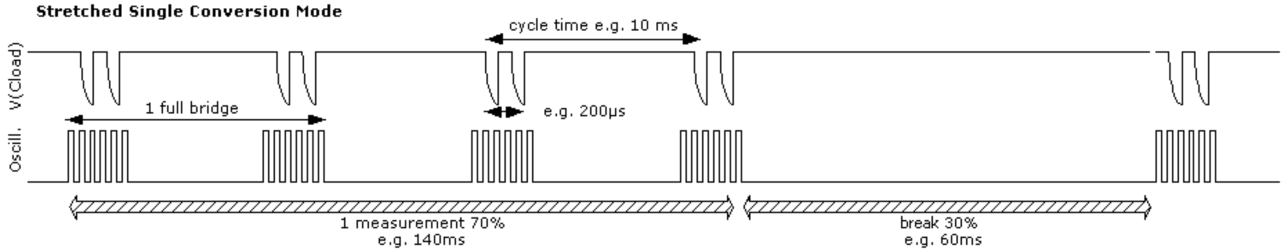
Those parameters set the stretch mode. The following 2 pictures show how the parameters are applied, one showing the continious stretched the other the single conversion stretched mode.

Figure 3.21: Parameters in continuous 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.

3.5.5 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	stretch = 0
			single_conversion = 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.

3.5.6 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 +2^MFake + 1)

Mfake = #Fake measurements, Temperature measurement on

Mfake-Register	#Fake Measurements
0	0
1	2
2	4
3	8

Example1: Cycle time = 110 µs
AVRate =12
Quattro bridge
Mfake = 1
Tconversion = 110 µs*(2*12*4 + 6 + 2^1 + 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 + 2^2 + 1) =199.21 ms
The maximum measuring rate is 5.02 Hz

3.5.7 Conversion Time / Measuring Rate (Single Conversion Mode)

If PSØ9 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 ms
→ measuring rate = 1 / 0.1344ms = 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).

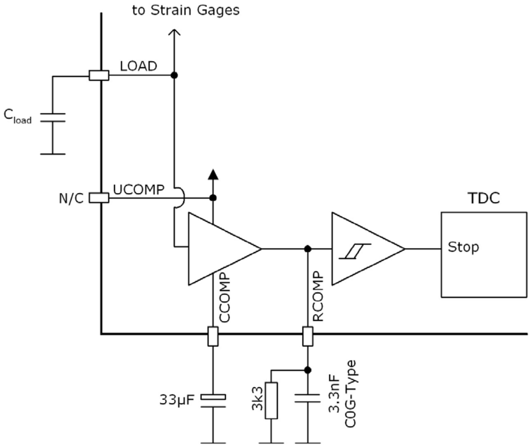
3.5.8 Comparator

The end of the discharge cycle is triggered by a comparator. PSØ9 offers an internal low noise comparator. Alternatively, an external comparator might be used for highest performance.

3.5.8.1 Internal Comparator

The internal comparator is selected by setting Configreg_12, sel_comp_int = 1. By means of the internal comparator it is possible to get about 50,000 divisions peak-peak at 2 mV/V, 5 Hz update rate and MEDIAN 5 software filter.

Figure 3.23: Internal comparator



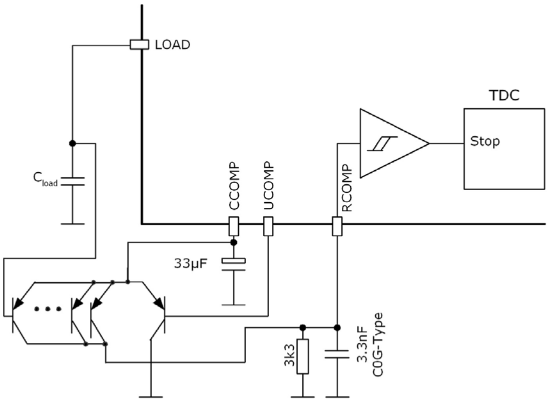
NEW with PS09

Please note: To test the internal comparator whilst the external comparator is populated on the board you just need to remove the transistors. Then, by switching the comparator control bit (Configreg_12, sel_comp_int) to internal enables you to run the measurement with internal comparator.

3.5.8.2 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 120,000 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.

Capacitors at CCOMP and RCOMP (both external and internal)

The capacitors at CCOMP and RCOMP are important for the low noise figure. For best performance we recommend 33 µF for CCOMP (ordinary electrolytic capacitor) and 3.3 nF for RCOMP (preferably COG / NPO capacitor).

For cost reduction, smaller values for the two capacitors are possible, however noise will sligthly increase by using smaller values:

Possible values:

CCOMP: > 1 µF

RCOMP: lower than CCOMP electrolytic capacitor divided by 3000

Example:

CCOMP = 1 µF → RCOMP < 1 µF/3000 → 330 pF selected.

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

When should an external comparator be used?

There are two 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.

3.5.8.3 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_comp_r[1:0]).

We recommend the following settings:

Con_Comp	= ‘b10	→	on during measurement
Sel_comp_r	= ‘b00	→	6k resistor selected
	= ‘b01	→	6k resistor selected
	= ‘b10	→	2.5k resistor selected
	= ‘b11	→	1.8k resistor selected

Use of higher comparator threshold (sel_compth2)

The use of the analog switch in PS09 also has some effects on the gain drift of PS09 over supply voltage. This is evident especially at low supply voltages (< 2.7V) with increasingly noisy behavior and very low PSRR. This can be associated with the behavior of the analog switch itself.

To counter this, there is a possibility to configure and select a second threshold for the comparator by the setting:

SEL_COMPTH_2 = 1   in Configreg_11 (Bit 12)

By setting this bit, a second threshold voltage is generated. This is about 20% higher than the default threshold voltage of the comparator (so the discharging gets shorter). This higher threshold voltage ensures fairly good gain stability over supply voltage, at very low supply voltages.

Hence this configuration is recommended for applications operating in PS09 mode with lower supply voltages (< 2.7 V) or when such a probability exists. A typical application for this setting is a battery driven consumer scale.

3.5.8.4 Correction of Comparator Delay (Source of gain drift of PS09 itself)

The focus of the comparator performance is on ultra low noise brings in a non-neglectable delay to the measurement. This delay depends on temperature and results in a gain error which is too high for precise weigh scale applications. To compensate for this gain error a correction measurement is done to eleminate the time delay caused by the comparator.

NEW with PS09

PS081 had two external resistors to realize the gain correction measurement, but PS09 makes the correction by making additional measurements over the strain gage resistors themselves. In other words, there are no external resistors necessary anymore (exception: if you operate a single half bridge, the 2 compensation resistor are still needed).

To activate the gain compensation set bit 7 in configreg_01 (En_gain)

The delay of the comparator mainly depends on some key components of the comparator circuit, mainly the capacitance of the low pass filter connected to the pin RCOMP (usually around 3.3nF, COG) and the middle capacitor connected to the pin CCOMP (usually around 33uF, elco). As these values can be selected by the user and vary in value therefore there is another factor to adopt for these changes to make the gain correction work with different hardware settings. This factor is called Gain_comp (configreg_10, bit[7:0]) and usually set to 1.28 (=0xA3). If the capacitor values change, the Gain_comp factor needs to be adapted accordingly (higher capacitor values means higher Gain_comp factor and vice versa).

At the correct value of Gain_comp 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.

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 PSØ9 the physical reasons are totally different. PSØ9 has a TD-Converter with no preamplifier. The gain drift of the PS09 electronics comes mainly from the delay time of the comparator. PS09 can determine the delay time by means of additional measurements over the strain gage resistors. As the delay is affected by hardware settings (see previous description of the influence of the capacitors) there is a factor to correct for these variations, called Gain_comp. If this factor is not adjusted properly, the gain error can be up to approx. 8ppm/K, if set appropriately this can be reduced to approx. 1ppm/K. Of course the Gain_comp factor only needs to be determined once during the development phase and can then later be used for the whole series production. In the first instance it is recommended to use the default value of 1.28 for Gain_comp, this is the proper value for the hardware designs given by acam. A detailed description of how to determine the proper Gain_comp factor in case the hardware differs significantly from the acam recommended circuits is found in the application note AN018.

3.5.9 Zero Drift of PS09 itself

Also the zero drift of the PS09 originates from a reason other than the drift of an AD-Converter. The reason of the remaining zero drift of our PICOSTRAIN products are (chip internal and external) parasitic resistor paths that are not or not perfectly compensated.

Because of the nature of the remaining zero drift, the value of this drift depends (externally) mainly 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. 10nV/V/K and lies therefore in the 50% limit of OIML 10000.
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, because of this it is recommended to lay them out as symetrically as possible. The cable to the load cell is not as critical because the wires have a much bigger diameter.

A special case is the Wheatstone mode. In this mode nearly 100% of the remaining parasitic resistances are compensated because of the kind of the wiring and measurement. Therefore, in Wheatstone mode the zero drift of PS09 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 PS09. To get an 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:

Table 3.2: Typ. offset drift values in different operation modes

Mode	350 Ohm SG	1 kOhm SG	OIML 10000
Fullbridge Standard *	±20 nV/V/K	±8 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, i.e. symetrical connection of port lines on the PCB

3.5.10 Internal Temperature Measurement (using Integrated Rspan)

NEW with PS09

PS09 has an integrated temperature measurement. For this measurement all needed components are integrated so that no additional external circuitry is required. Basically it is another ratio measurement between the external strain gage resistors (relatively temperature stable) to an internal aluminium resistor in the chip (high sensitivity to temperature).

The temperature information can be used to correct the gain drift of uncompensated load cells. Due to the integrated aluminium resistor as temperature dependent element we call this type of temperature compensation also „integrated Rspan" compensation.

Resolution of the temperature measurement:

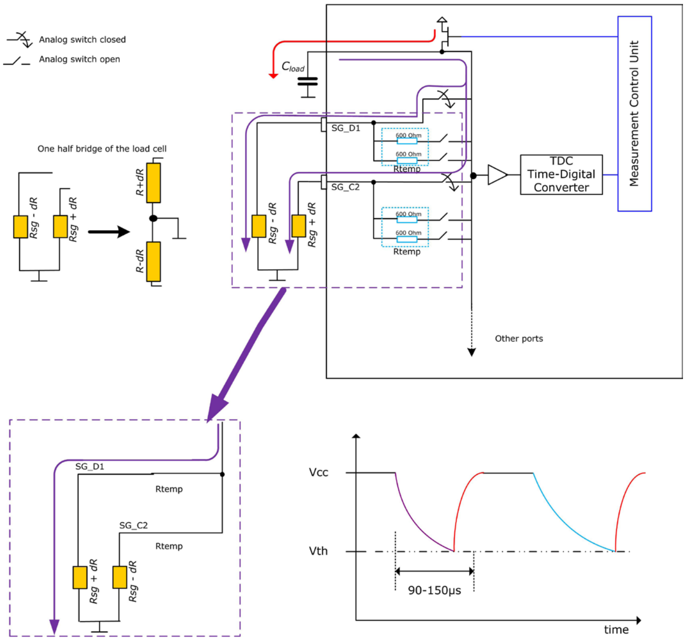
Because of the used high resolution TD-Converter and the metal resistor the resolution of the temperature measurement is typically 0.01 °C.

Background:

As already mentioned, in PS09, the integrated metal resistors Rtemp are used only for measuring the temperature. There is one Rtemp resistor connected at port SG_C2 and another at port SG_D1. Both the Rtemp are exactly identical and are controlled by the same configuration bit.
The value of Rtemp can be selected between 300 Ohm and 600 Ohm by the configuration bit sel_rtemp_300R in Register 14. The 300 Ohm setting is recommended for load cells with 350 Ohm strain gage resistors, when the load cell has 1 kOhm strain gage resistors, 600 Ohm setting for the Rtemp is recommended.
The temperature measurement principle uses the strain gage resistors at Ports C and D as the reference resistor, which are characterized by very low temperature coefficient of resistance (TCR) in the range of 5 ppm/K. Hence they are very stable over temperature. The integrated Rspan i.e. the Rtemp has a specified temperature coefficient of approx. 3500 ppm/K and it is the temperature sensitive element used in measurement.

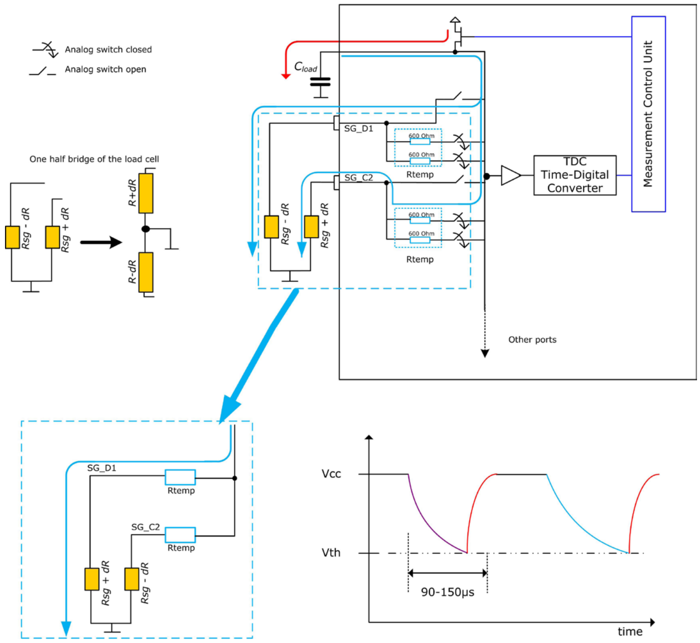
Principle:

Figure 3.24 Internal temperature measurement with integrated Rspan (Rtemp)



The principle of measuring the temperature involves performing one discharge cycle through a parallel combination of the temperature stable reference (strain gage) resistors at Strain gage ports SG_C2 and SG_D1. This is shown in the above figure indicating the discharge cycle in purple. The analog switches at the ports are controlled appropriately by the measurement control unit.

Fiogure 3.25 Integrated Rspan (Rtemp)



This is followed by a discharge cycle through a parallel combination of the effective series resistance of the strain gage resistor and the temperature sensitive resistor (Rsg + Rtemp) at Port C, and the effective series resistance of the strain gage resistor and the temperature sensitive resistor (Rsg + Rtemp) at Port D.
The discharge cycle is through the combination

(Rsg_D1 + Rtemp_D1) | (Rsg_C2 + Rtemp_C2) shown in the figure in blue.

Rsg_D1	→	Strain gage resistance at port SG_D1
Rtemp_D1	→	Integrated Rspan resistance at port SG_D1
Rsg_C2	→	Strain gage resistance at port SG_C2
Rtemp_C2	→	Integrated Rspan resistance at port SG_C2

The ratio of the 2 measurements contains the temperature information of the system and hence is used to compensate for the temperature drift of the system. This ratio can also be processed to calculate the absolute temperature of the system.

Value of Rtemp can be 600 Ohm or 300 Ohm (600 Ohm | 600 Ohm). Thus the effective resistance of this combination for e.g. with Rsg = 350 Ohm and Rtemp = 300 Ohm is

(350+300) | (350+300) = 325 Ohm

The measurements from the above 2 discharge cycles are processed internally and the ratio which is given out as the result of the temperature measurement is

(Rtemp / Rsg) @ the current temperature

Where, Rsg is the strain gage resistance. This ratio is a function of temperature and thus can be used for temperature compensation.

Note: The strain gage connections at ports C and D must be made such that the resistor experiencing the positive strain is connected at port C and the resistor with the negative strain is connected at port D. Only this can ensure that the result Rtemp / Rsg is free of effects of the strain on the load cell itself. Hence this must be ensured for correct temperature measurement.

Configuration:

- The temperature measurement on chip is enabled by setting the integrated_rspan bit (Bit 8) in Config_reg 1.
- Depending on the strain gage resistance used, the value of Rtemp is selected by using the bit sel_rtemp_300R in Register 14.

Temperature measurement result:

It must be noted that per measurement cycle, the temperature measurement is made only once at the end of the

cycle. The result is updated with the result registers at the end of the measurement cycle. The result ratio stated above can be read from **RAM address 245**, as part of the result registers at the end of the measurement. Interpretation of the result:
The result ratio is a function of temperature, and can be interpreted as follows.

Where T is the current temperature and TkRtemp = 3500 ppm/K is the temperature coefficient of resistance of Rtemp as per specification. 3500 ppm/K is a typical value and can vary from chip to chip slightly by +5%.
The temperature information can be directly used to adjust for the gain and offset drift of the load cell (See Chapter 3.5.12), fine adjustment of the correction can be done by setting Tk-gain and Tk-offset (Configreg_08 and 09) factors. From the above interpretation, the current temperature can also be calculated very accurately with the help of the result available from the PS09.

Note: The value of the Rtemp has a variation over different chips of approx. + 10%. Also TkRtemp has a small variation of a few percent. Hence the resulting ratio at a given temperature also changes from chip to chip by several percent. However the Rtemp of every chip in a system can be calibrated as part of the usual offset adjustment process itself, without any additional steps. This calibration of Rtemp is necessary for temperature drift compensation using the integrated Rspan (Rtemp). Please contact acam for further information on this compensation method.

Temperature Measurement using PS09:

With the method used in the PS09 for temperature drift compensation, the PS09 has adequate information to calculate the temperature of the system with high resolution. For standard temperature measurement (e.g. consumer scales) no additional measurements have to made, only the system has to be calibrated once at a nominal temperature at production. The calibration procedure has been explained in the next section.

The following are the formulae used to calculate various intermediate results and then the final result.

1. With the result ratio read out from the RAM address 245 of the PS09, a factor called current ratio at that temperature is calculated.

$$\text{Current ratio CR} = 1 + \left\{ \frac{\text{Content of RAM address 245}}{2\text{^}20} \right\} \longrightarrow (1)$$

This current ratio CR is basically used for all further calculations.

2. The value of the Result ratio at 0°C is back calculated using the measurement value at any known nominal temperature TN as

$$\text{Ratio @ 0°C} = \frac{\left\{\dfrac{\text{TkR}_{\text{temp}} * \text{T}_{\text{N}}}{10^6}\right\} + \text{CR @ T}_{\text{N}}}{\left\{\dfrac{\text{TkR}_{\text{temp}} * \text{T}_{\text{N}}}{10^6}\right\} + 1} \longrightarrow (2)$$

TN is the nominal temperature °C.

3. The TkRtemp of the Rtemp resistor is specified as 3500 ppm/K. However, when the Rtemp is connected in series with the Strain gage resistor, the temperature coefficient of the combination changes. The resultant temperature coefficient at a nominal temperature TN is thus denoted by TkRes and is calculated using the formula

$$\underset{\text{in ppm/K}}{\text{Tk}_{\text{Res}}} = \left\{\frac{(\text{CR @ T}_{\text{N}}) - (\text{Ratio @ 0°C})}{\text{Ratio @ 0°C} * \text{T}_{\text{N}}}\right\} * 10^6 \longrightarrow (3)$$

4. Finally the temperature of the system at any point of time can be calculated using the formula

$$\underset{\text{in °C}}{\text{Temperature T}} = \left\{\frac{\text{CR} - (\text{Ratio @ 0°C})}{\dfrac{\text{Tk}_{\text{Res}}}{10^6} * \text{Ratio @ 0°C}}\right\} \longrightarrow (4)$$

Note: The CR value in this equation must be calculated according to Formula 1, by reading the current value of the RAM address 245.

Calibration of a system for temperature measurement:

A system designed using the PS09 with the capability of temperature measurement must be calibrated once before it can be used. For the calibration, a nominal known temperature is needed. It is sufficient to use the room temperature itself as the nominal temperature.

1. Note down the nominal (room) temperature of the system (TN°C).
2. Read out the result ratio from the PS09 at RAM address 245 and calculate the Current Ratio CR using formula (1).
3. Next, calculate the Ratio @ 0°C using formula (2). Store this value in the OTP or data EEPROM for future use.
4. In the next step, calculate the effective temperature coefficient of resistance at the nominal temperature, TkRes using the formula (3). Store this value in the OTP or data EEPROM for future use.
5. The calibration process is complete without going into a temperature chamber at all. For future use, the stored parameters from Steps 3 and 4 can be substituted in the formula (4) and the temperature can be directly calculated.

Remark:

The inaccuracy of this procedure results from the variation of TkRtemp from chip to chip. TkRtemp is quite a stable value because of the behavior of the metal resistance but nevertheless vary by some percent from chip to chip. Therefore the typical error of the temperature measurement is 1°C at a temperature change of 30°C.

Example:

Calibration is done at 20°C at the factory. Hence 20°C is measured very accurately with close to no error. The error at 50°C (30°C more than the calibration temperature) is typically +1°C. For most purposes this accuracy should be high enough.
For highest accuracy i.e. for professional use, a calibration at 2 different temperatures has to be made.

Temperature measurement in Half bridge configuration:

In the classical half bridge connection (Section 3.3.1), at port C and D, external resistors are connected. The external resistor at ports C and D and their temperature coefficient of resistance will play an important role in the temperature measurement, when enabled in the half bridge configuration. The external resistors at port C & D would be used as the temperature stable reference resistors. The value of the resistors must be same as the strain gage resistor value. It is recommended for these resistors to have a temperature coefficient not more than 50 ppm/K.

3.5.11 Temperature Compensation (whole system)

Nowadays the gain and offset drift of the electronics is typically much lower than the one of the sensor (e.g. load cell). The drift of the electronics is approx. 5-10 times lower than the one of the sensor. Therefore a temperature compensation for the whole system is required. PICOSTRAIN introduced here with its unique temperature compensation concept a way to compensate for the gain and offset drift of the whole system very effectively and without manual trimming of the sensor.

Bascially there are two ways to compensate the sensor/load cell, they are:
1. Using the gain compensation resistor of the load cell (Rspan)
2. Using the integrated Rspan i.e. internal temperature measurement

The temperature correction is based on an integrated algorithm inside the chip. This algorithm has to be fed by two factors to make the compensation, namely TK-Gain and TK-Offset (Configreg_8 and Configreg_9). To employ those factors and generally the use of the temperature compensation the bit mod_rspan (Configreg_01) needs to be set. The factors in detail:

Table 3.3: Temperature compensation factors

Paremeter	Terminal	Description
mod_rspan	Configreg_01, bit 6	Set to „1" to enable temperature correction
integrated_rspan	Configreg_01, bit 8	0 = internal measurement is not used → Rspan as temp. sensor 1 = activate and use internal temperature measurement for compensation
tk-gain*	Configreg_08, bit [23:0]	Gain compensation factor
tk-offset*	Configreg_09, bit [23:0]	Offset compensation factor

* factors formerly named Mult_TKG and Mult_TkO

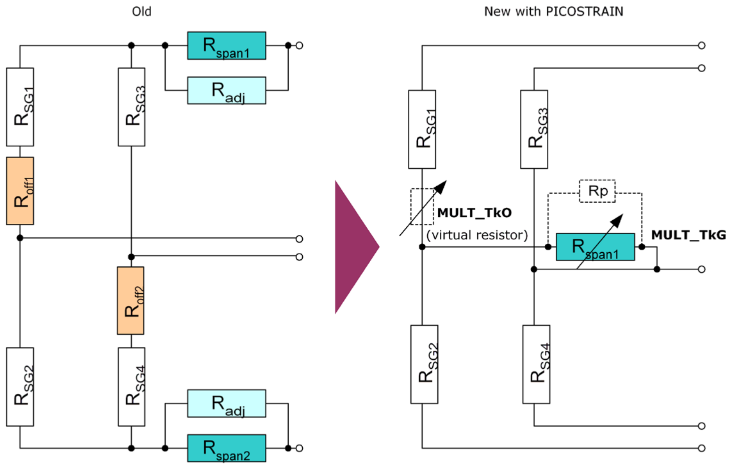
The most accurate compensation can be achieved by using the gain compensation resistor (Rspan) on the load cell. Please see the next section for a detailed description of the compensation of the load cell's gain and offset drift.

3.5.12 Gain and Offset Drift Compensation of the Load Cell (TK-Gain* and TK-Offset*)

* formerly named Mult_TkG and 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. Today's challenge is an optimized complete system (scale) and not only a very good electronic. Therefore, with the **PICO**STRAIN 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 one compensation resistor (Rspan).

Figure 3.25: Rspan-simplification with Rp



PS09 can measure this compensation resistor and correct it by an algorithm in the processor, based on the correction factors TK-Gain for the gain drift and TK-Offset 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 nor to trim

Rspan manually. A further advantage of this method is that it is no longer necessary to trim the load cell exactly to zero, no zero offset compensation resistors are needed (the zero offset is recognized by the chip internally!).

All these points together lead to a much simpler load cell circuit compared with the traditional approach:

With this temperature compensation method the gain and offset drift behavior of the load cell can be improved by PS09. This is a very comfortable method to improve the quality of the complete scale without modifying the load cell or the electronic. Using this method for example in a digital load cell, the production can be simplified at a higher quality level and lower cost.

Some examples how to use TK-Gain, TK-Offset:

- 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 (i.e. stable over production lot but wrong) it can be corrected directly by PS09 without going into the temperature drift chamber.
- If a run in the temperature drift chamber is done, the correction factors for TK-Gain and TK-Offset 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 comfortable to meet the requirements of high end scales (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 TK-Gain and TK-Offset. Also a screencast is available for temperature compensation by means of the internal temperature unit (http://www.acam.de/download-center/picostrain).

PS09 can furthermore correct uncompensated load cells (cells without a gain compensation resistor Rspan). It therefore uses the temperature measurement information instead of the Rspan value. The adjustments are also done by means of the two factors TK-Gain and TK-Offset. However, the accuracy of this compensation will not be as good as the use of an Rspan compenstion resistor. Improvements by a factor 6 to 8 (compared to the uncompensated load cell) can be expected. 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 TK-Gain and TK-Offset method.

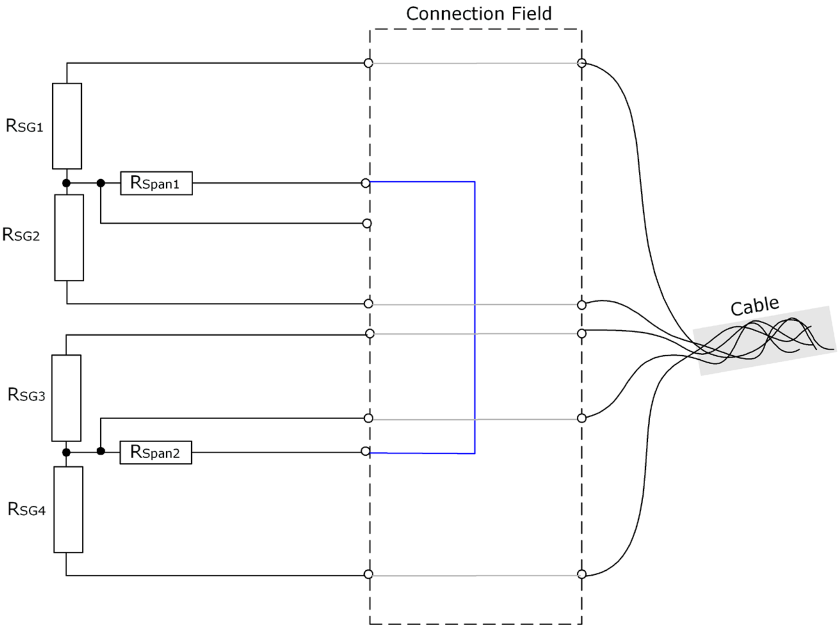
Please see foregoing chapter Temperature Compensation (whole system) to see how the activation of the temperature compensation and selecting the different available methods is done.

3.5.13 Annotations Rspan (gain compensation resistor)

- **PICO**STRAIN needs only one Rspan resistor.
- As the Common Mode Rejection Ratio (CMRR) of the **PICO**STRAIN products is very good (>135dB) there is no need to use two Rspan resistors.
- Indeed, **PICO**STRAIN can not handle bridge with two Rspan resistors.

So the easiest way of course is to use load cells which natively have only 1 Rspan resistor. Nevertheless if you have a load cell with two Rspan resistor, it is easy to connect it in the correct way, as shown in figure 3.24.

Figure 3.26: Two Rspans in a row



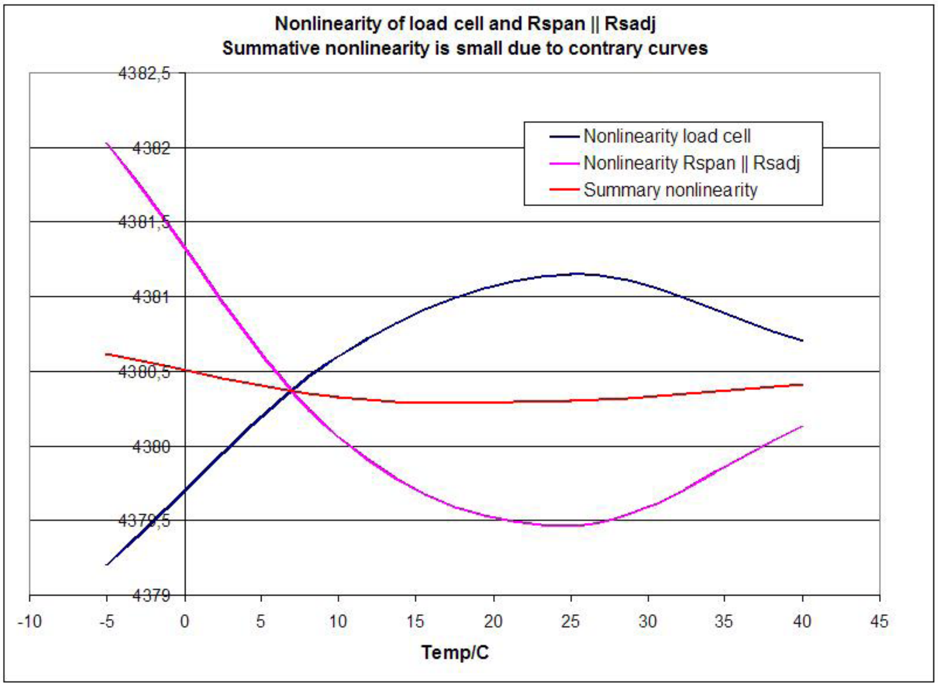
A possible way to change a load cell with 2 Rspans is to switch them in series. This is possible if the connections of the Rspan resistors are available as illustrated in the following picture:
you make any re-wiring proposed in the connection field of the load cell. This is true for the changes regarding Rspan as well as the change from Wheatstone-wiring to **PICO**STRAIN-wiring (please see also chapter 3.3.2**).**

3.5.14 Nonlinearity of gain drift over temperature

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

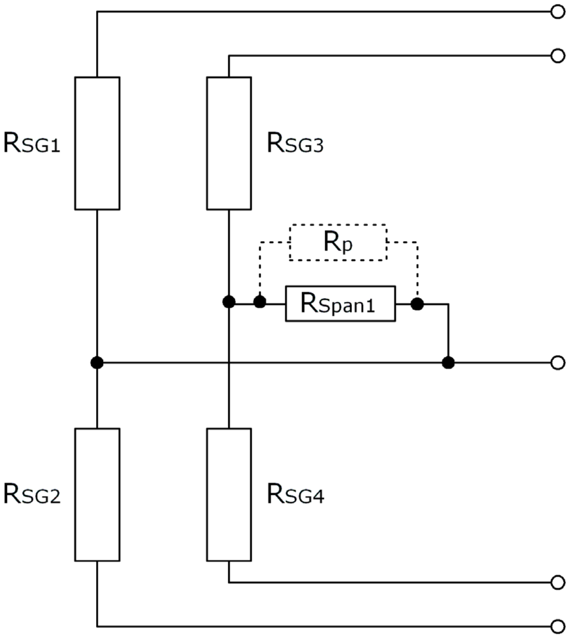
Independent from the the **PICO**STRAIN gain drift compensation there is always a nonlinearity over temperature coming from the load cell itself (mainly caused by material, glue, wiring, etc.). To compensate for that nonlinearity a resistor network, consisting of the Rspan and a paralleled adjustment resistor Rp (was called Rsadj in former picture) is used. The nonlinearity of the selected Rspan || Rp combination behaves contrarily to the load cell‘s nonlinearity so that overall nonlinearity can be reduced by this measure. The following picture illustrates the effect:

Figure 3.27: Nonlinearity of gain drift



NEW with PS09 Meanwhile in a classical load cell the Rp or Radj is added manually, PS09 offers the unique possibility of adding Rp virtually, i.e. by a register setting. In other words, a virtual resistor can be set by means of enabling it in the configuration and setting a register for the value. The following illustration shows the effect of the virtual Rp resistor, without being physically present:

Figure 3.28: Possible Rp



Not all load cells require this measure. First, because the load cell can show only a low nonlinearity so that Rp is not needed. Second, if the load cell does not need to match any specifications for legal for trade, some nonlinearity may be acceptable. However, if the nonlinearity cannot be neglected, Rp can be added and the overall nonlinearity reduced therefore. Good indicators to decide whether the nonlinearity is too high or not are:

a) In the first run of determining TK-Gain (temperature compensation) you get a value < 0.8

b) A temperature run with TK-Gain = 1.0 shows a gain drift > 100ppm/K

c) The load cell formerly had a parallel resistor for nonlinearity compensation

To use the virtual parallel resistor, enable the bit en_tkpar in configreg_01 (bit 11). By means of configreg_07 you can then set the Rp – value in multiples of the Rspan value (from -8.0 to +7.999). E.g. if you have an Rspan value of 40Ω and the desired value for Rp is 320 Ω then you would set Mult_tkpar to 7.9999.
For the size of Rp it's best to take a value which is close to the formerly used one (if there was one). Otherwise, by means of the ordinary temperature compensation (TK-Gain and TK-Offset determination) a rough estimation of the Rp – value can be derived. The Excel-Sheet for calculation can be obtained from acam.

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 otp_usr_prg =1, the program in the OTP is started.
Specialties of the post-processing are:

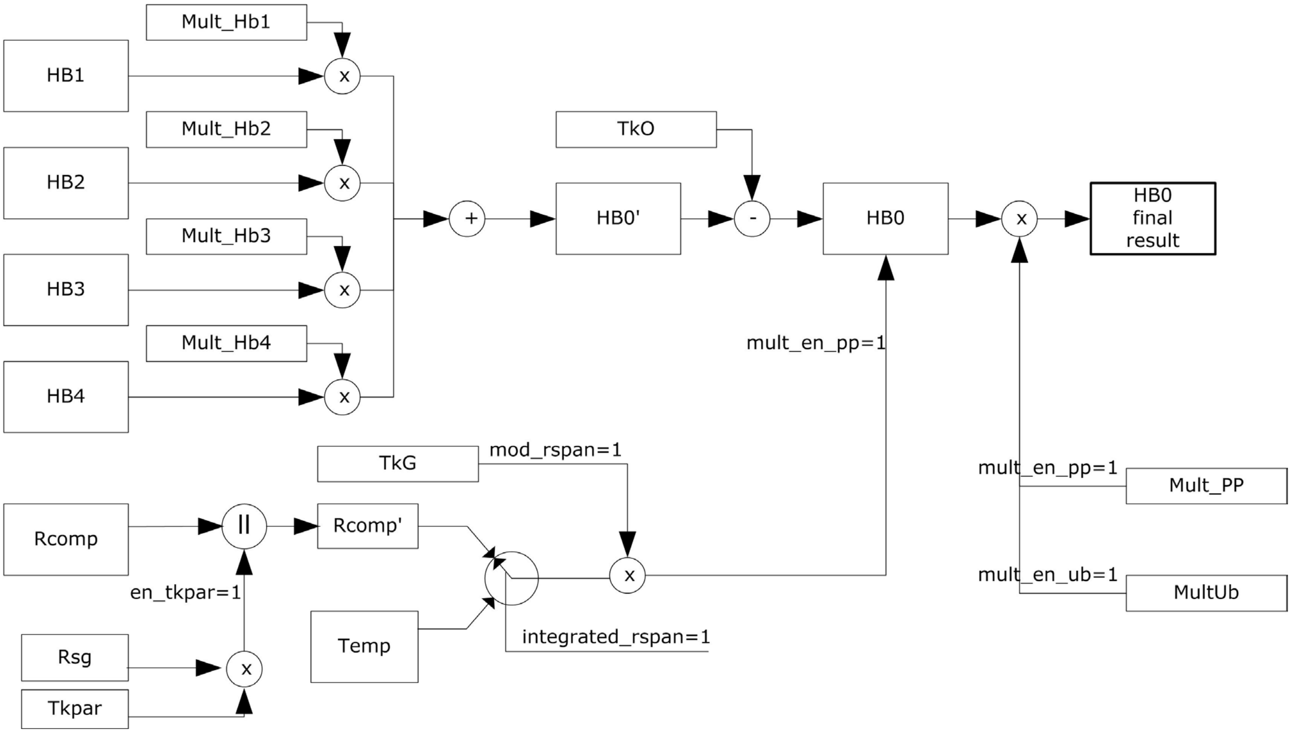
- The results of the four half-bridges have independent multiplication factors. This offers the possibility to do a software correction for off-center weights in quattro applications.
- The strain sensors and the span compensation resistor are separated. The gain compensation resistor can therefore be adjusted by software. Also 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.
- With **PICO**STRAIN the offset is not affected by the span compensation. The offset can be corrected by software.

The corrected result may be further multiplied by correction factors depending on the battery voltage. This supports power supply rejection and allows an operation directly from a battery without regulation.

A simple example program (extract) to display the results could be:

```
ramadr      224+20      ;HB0 result, 224 base address for results
move        x , r       ;Load x-Accu with the result
move   y , 2            ;Load y-Accu with the comma position
jsub   no2lcd           ;Convert into 7-segment display using a
                        ;subroutine (see software library)
jsub   send2display     ;Send the new value to display to the LCD
                        ;controller
clrwdt                  ;Clear the watchdog
stop                    ;Stop the µC
```

Figure 3.29 Post-processing



3.6.1 Off-center Correction for Quattro Scales

Some scales – for example body scales – have four load cells, usually a half bridge sensor. The indicated weight might vary with the position on the platform in case the load cells do not all have exactly the same sensitivity. PSØ9 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 doing 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. The multiplication factor is then derived from calculating the ratio between measured weight / expected or nominal weight. The factor is stored as a 24-bit value in the registers Configreg_04, 05, 06 and 07.

3.6.2 Mult_UB - Power Supply Rejection

PS09 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

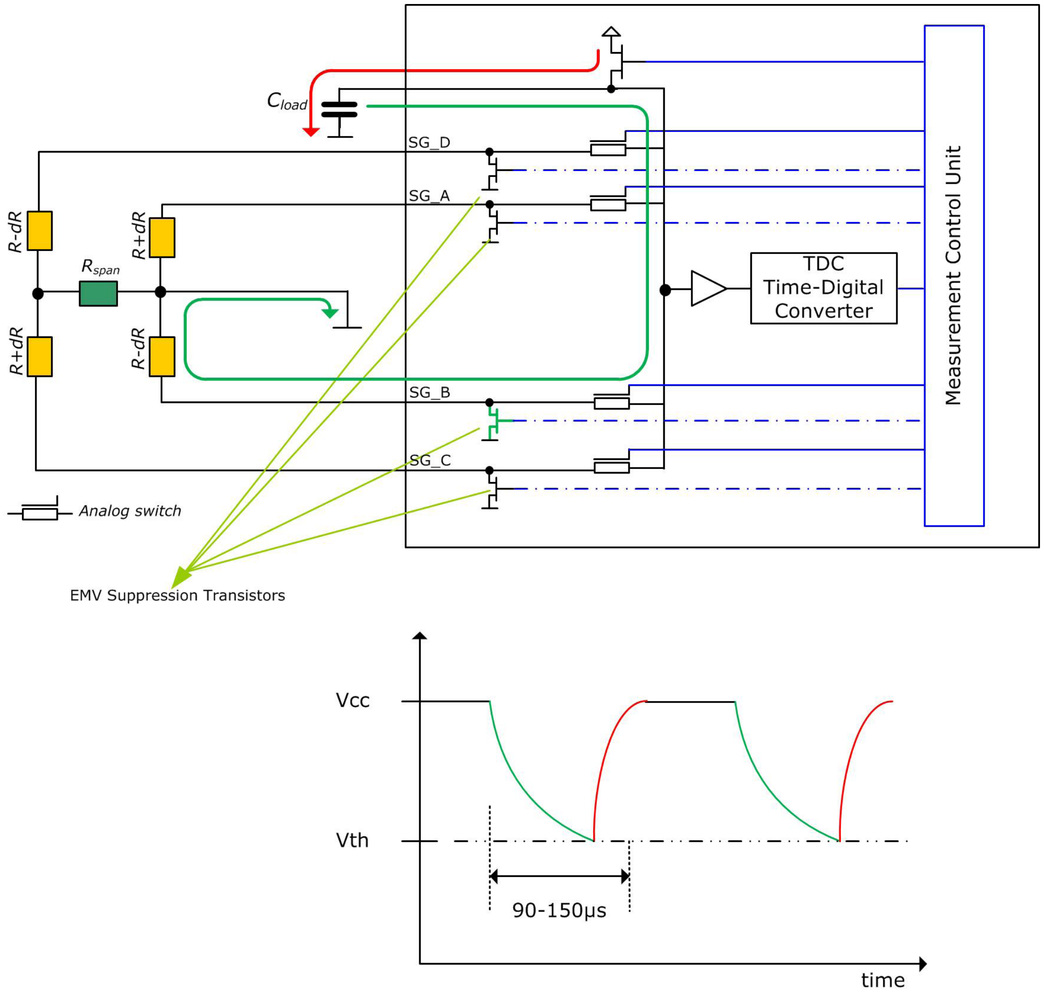
$$HB = HB/(1 + UB*[-128 \ldots 127] / 221).$$

The standard setting for Mult_UB is 0xF7.

3.7 Suppression of EMI

Electromagnetic interference (EMI) is a main cause of concern while considering factors that affect measurement quality. By virtue of the method adopted to control the ports in PS09, the PS09 mode of measurement shows better EMI behavior.
There are dedicated N-Channel transistors in the PS09 connected between each port and ground, only for EMI suppression. In every measurement cycle, the N channel transistors of the unused ports are enabled and the transistor of the currently measured port is disabled. Hence, the unused ports are connected to Ground through a low resistance path of only some few Ohm (through the EMI suppression N Channel transistors). This disconnects the unused port from the measurement path leading to a higher suppression of EMI. This helps in mainly suppressing the 50Hz noise behavior typical to long unshielded lines as in Quattro scales.
Figure 3.30 shows where these transistors are connected at each port. This is shown for a Full bridge connection in PS09 mode.
In this figure, the discharging path through Port SG_B is shown, i.e. Port B is the current port being measured. During this measurement cycle, the EMV suppression transistor on Port B is switched OFF and is therefore open. The EMV-transistors connected to Ports A, C and D (unused ports) are switched ON. These ports are hence connected to Ground through the low ON-resistance of the N-Channel transistor. Thus the unused ports are disconnected from the measurement path.

Figure 3.30 PS09 with EMV transistors



Enabling the EMI suppression transistors:

The protection transistors can be activated by configuring the en_emi_noise_reduction bit (Bit 7) in Configreg_12 to 1. This configuration is ONLY valid in the PS09 mode, do not set the bit when operating the PS081 compatible mode.

Table of Contents

Page

4 Peripheral Components & Special Settings

4.1 Oscillators

The PSØ9 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. This oscillator is always on and running continuously. Further, the PS09 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. This oscillator is configured as follows:

Configuration: Configreg_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

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]). Basically, the configuration bits "sel_start_osz" can be set to 1 for continous, single conversion and stretched modes. In single conversion- and stretched mode sel_start_osz = 2 is preferred. The startup time for the 4MHz oscillator is about 50 µs to 100 µs and slightly depends on the supply voltage.

NEW
with PS09

Optionally in PS09, there is a possibility to use a built in high quality RC oscillator of 4MHz instead of the ceramic resonator. This RC oscillator needs about 1.2 mA@ 3.0 V for operation. This oscillator is selected by setting the sel_rc1 bit in Configreg_13 to 1. The power supply of 3V needed for this RC oscillator must be made available by setting the connect_vcc_rosc - bit in Configreg_00.

NEW
with PS09

There is a third option to use a built in RC oscillator of lower quality than the one mentioned above, but with a significantly lower operating current. However, when this clock is used, the measurement is noisy. As it is only suitable for low current scanning appllications, its use is generally not recommended. This oscillator is selected by setting the sel_rc2 bit in Configreg_13 to 1.

Layout Considerations for 4 MHz ceramic oscillator:

The oscillator should be placed close to the PSØ9. The area around the oscillator should be flooded by a ground plane. The SPI or IIC wires should not cross the oscillator lines.

LCD Clock source:

PS09 can generate a clock output signal, which is suitable to drive a external LCD-controller (e. g. HT1620) or a microcontroller. It is intended to replace a 32Khz quartz oscillator on the PCB. See section 4.8 "Driving an External LCD Controller" for more details.

4.2 Communication Modes (SPI, IIC, Stand-Alone)

NEW with PS09

Unlike PS081 with only 1 communication interface (SPI), PS09 has another serial interface which is IIC. Therefore, instead of SPI_ENA there is the MODE pin (pin 21) to select the interfaces. The modes are as follows:

Table 4.1: PS09 Communication Modes

Communication Mode	Description
MODE = 1	Front end mode with IIC interface active
MODE = 0	Front end mode with SPI interface active
MODE = HiZ (n/c)	PS09 Stand-alone mode

For further details on the interfaces please see the dedicated section (SPI and IIC interface). In case the PS09 is operated in stand-alone mode, the SPI /IIC communication pins can be used as Multi_IO for general Input/output like buttons or software defined interface functions (ie. master SPI /IIC). Changing the status at the MODE pin needs approx. 10ms to become active.

4.3 Multiple Input/Output pins (Mult_IO)

PS09 has eight I/O pins:

0 - DO_IO0	Serial data-out in SPI mode / Bidirectional Data line in IIC mode/ Multipurpose IO
1 - DI_IO1	Serial data-in in SPI mode / Multipurpose IO
2 - CLK_IO2	Serial clock / Multipurpose IO
3 - MULT_IO3_C1	Multipurpose IO / Capacitive sensor 1 / Interrupt
4 - MULT_IO4_C2	Multipurpose IO / Capacitive sensor 2 / Interrupt
5 - MULT_IO5_C3	Multipurpose IO / Capacitive sensor 3
6 - MULT_IO6_C4	Multipurpose IO / Capacitive sensor 4
7 - MULT_IO7_CREF	Multipurpose IO / Reference Capacitive sensor

- 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 (MODE pin = unconnected).
- Using the pins MULT_IO3_C1, MULT_IO4_C2 and MULT_IO6_C4 as inputs, up to 24 multi input keys externally to the PS09 can be realized
- Using the pins MULT_IO3_C1, MULT_IO4_C2, MULT_IO5_C3, MULT_IO6_C4 and MULT_IO7_CREF as inputs, up to 4 capacitive sensor keys can be connected (see next section)

- MULT_IO3_C1 and MULT_IO4_C2 pins can be used as interrupt pins, either for generating interrupt from the PS09 on completion of every measurement or for generating an external interrupt to the processor of the PS09. Additionally, MULT_IO3_C1 or MULT_IO4_C2 can be used to generate a 32 kHz clock to drive an external LCD-driver or microcontroller.
- MULT_IO3_C1 or MULT_IO4_C2 can be configured to act as the TxD (transmit) line of the PS09 UART.
- Mult_IO2, -3, -4, -5. One of these pins can be configured as the RxD (receive) line of the PS09 UART

Note: The terms MULT_IO and GPIO are used in this documentation interchangeably!

4.3.1 Configuration

DO_IO0	Configreg_11, bit [1:0]	io_en_0_sdo
DI_IO1	Configreg_11, bit [3:2]	io_en_1_sdi
CLK_IO2	Configreg_11, bit [5:4]	io_en_2_sck
MULT_IO3_C1	Configreg_11, bit [7:6]	io_en_3_mio
MULT_IO4_C2	Configreg_11, bit [9:8]	io_en_4_mio
MULT_IO5_C3	Configreg_11, bit [11:10]	io_en_5_mio
MULT_IO6_C4	Configreg_11, bit [13:12]	io_en_6_mio
MULT_IO7_CREF	Configreg_11, bit [15:14]	io_en_7_mio

4.3.2 I/O Port definition

In standalone mode (no use of SPI or IIC communication interface) all eight I/O pins are multiple purpose I/Os. They can be configured as input or outputs via enable bit pair in configuration register 11

The appropriate bit pair of each port can be configured as follows:

00 = output

01 = input with pull-up

10 = input with pull-down

11 = input

NEW
with PS09

For compatibility reasons the RAM addresses used to access the status- and result registers are equivalent to that of PS081, but with an offset address of 224. In PS081 the results are located at address 16 to 31, in PS09 they are in address 240 to 255 (224+16 to 224+31)

4.3.2.1 I/Os as digital Inputs

If the I/O pins are configured as digital inputs, the digital input buffers have to be explicitly enabled by setting the respective “io_en_digital“ bit combination in Configreg_11 to “01“. The status for each of the eight I/O pins can be read from I/O status register in RAM adress 224+29, bit 23-16. Rising edge and falling edge occurrence on the 8 Multipurpose I/O pins is also indicated in the I/O status register and provided in bit 15 to 0.

Table 4.1: Bit content of the I/O status register on RAM address 253

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Status_IO7	Status_IO6	Status_IO5	Status_IO4	Status_IO3	Status_IO2	Status_IO1	Status_IO0	Rise_edge_IO7	Rise_edge_IO6	Rise_edge_IO5	Rise_edge_IO4	Rise_edge_IO3	Rise_edge_IO2	Rise_edge_IO1	Rise_edge_IO0	Fall_edge_IO7	Fall_edge_IO6	Fall_edge_IO5	Fall_edge_IO4	Fall_edge_IO3	Fall_edge_IO2	Fall_edge_IO1	Fall_edge_IO0

Note: The status of the above register is updated after every measurement completion and during the active phase of the sleep mode.

4.3.2.2 I/Os as digital Output

When the I/Os are configured as digital output pins (in Configreg_11 by clearing the respective bits to 00), the output states are set in the Configreg_00. Whatever is written to the bit is available on the respective I/O pin.

Table 4.2: Configuration of the output state of the Mult_IOs in Configreg 00

...	15	14	13	12	11	10	9	8	...
...	Output state of MULT_IO7_CREF	Output state of MULT_IO6_C4	Output state of MULT_IO5_C3	Output state of MULT_IO4_C2	Output state of MULT_IO3_C1	Output state of CLK_IO2	Output state of DI_IO1	Output state of DO_IO0	...

4.3.3 Multi-input keys

On each of the pins MULT_IO3_C1, MULT_IO4_C2 and MULT_IO6_C4, up to 8 keys can be connected, thus enabling a maximum of 24 Multi-input keys to be possibly connected by only using 3 I/O pins. The Multi Input keys are connected using Resistors of appropriate values either to Vcc or Gnd. More details will follow in section 4.3.3.1.

Each of the Multi Input ports can be individually enabled / disabled using the mi_enable bits of Configreg_13. The bit 11 corresponds to the enable/disable of Port MULT_IO3_C1, bit 12 corresponds to Port MULT_IO4_C2 and bit 13 corresponds to Port MULT_IO6_C4.

Every key is connected to VCC or GND using a resistor. To connect a key to a Multi Input port, the respective I/O pin must be first configured as an input port, as shown in the port definition above. The default state of the input port is High-Z. Additionally in order to avoid cross currents due to intermediate voltage levels that occur across the switch, the digital input buffers to the chip must be explicitly disabled by setting the respective io_en_digital bit in Configreg_11 to 0. E.g. to connect the keys to MULT_IO3_C1 pin, then, io_en_digital[3] bit in Configreg_11 , Bit 19, must be 0. The values of the resistors to be used are prescribed but are tolerant in a wide range.

The frequency at which the keys are sampled at the MULTI_IN ports is set by controlling bits mi_updaterate and mi_sel_clk5k in Configreg_13. The basic clock frequency for sampling the multi-input keys connected to the ports is set to 5 kHz or 10 kHz, by using the mi_sel_clk5k bit. This basic frequency is divided further in the chip depending on the settings of the mi_updaterate bits and thus the final sampling frequency for the keys is generated. The following table shows how the scanning frequency is selected by configuration.

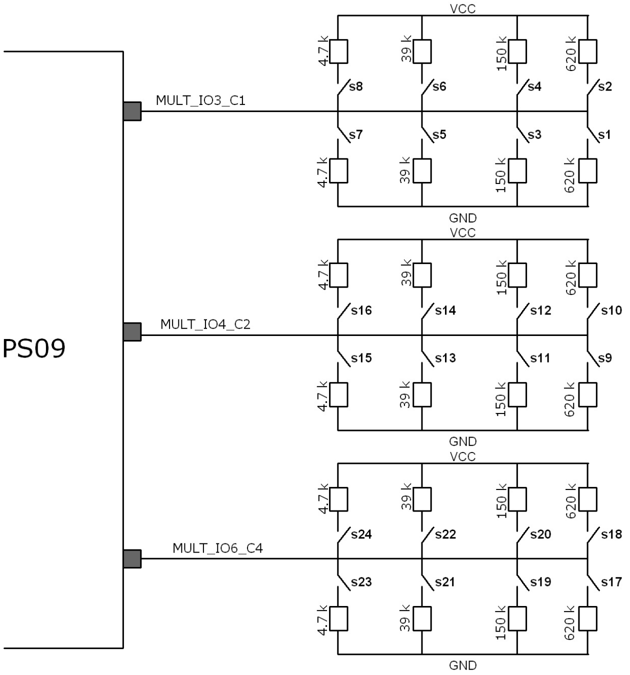
Table 4.3 Configuration of the Multi input keys scanning frequency

mi_sel_clk5khz	mi_updaterate[2:0] in Configreg_13	Scanning frequency for the Multi-input keys
0 -> Basic frequency = 10kHz	0	12.5 Hz
	1	25 Hz
	2	50 Hz
	3	100 Hz
1 -> Basic frequency = 5kHz	0	6.25 Hz
	1	12.5 Hz
	2	25 Hz
	3	50 Hz.

4.3.3.1 Connecting the multi input keys

For a set of 8 keys, 8 resistors are required in total – 2 x 4.7 kOhm, 2 x 39 kOhm, 2 x 150 kOhm and 2 x 620 kOhm. The following figure shows how the keys can be connected by connecting the 8 resistors to either VCC or GND. The keys are labeled as S1-S8 for the MULT_IO3_C1 port, S9-S16 for the MULT_IO4_C2 port and S17-S24 for the MULT_IO6_C4 port.

Figure 4.1 Connecting the multi input keys



When a key is pressed, the external resistor at the key is connected in parallel with an internal circuit. The internal circuit can evaluate by comparison what the external resistor value is, and if the resistor is connected to VCC or GND. Thus the PS09 can detect if the key is pressed or not.

4.3.3.2 Reading the key status

The status of the up to 24 possible multi input keys can be read from Multi Input Status register onRAM address 224+28. Bits 0 to 23 of this register represent the status of the keys S1-S24 respectively. The status of the keys is updated after every measurement completion.

Table 4.4 Current status of the up 24 multi input keys RAM address 224 + 28)

Bit 23	....	Bit 1	Bit 0
Key_Status_S24	....	Key_Status_S2	Key_Status_S1

The occurrence of a falling edge on the 24 Multi input keys is indicated in the 24 bits of RAM address 224+26.

Table 4.5 Falling edge Status register for the up to 24 muli keys, located on RAM address 224+26

Bit 23	....	Bit 1	Bit 0
Fall_edge _S24	....	Fall_edge _S2	Fall_edge_S1

The occurrence of a rising edge on the 24 possible Multi input keys is indicated by the respective bit of RAM address 224+27.

Table 4.6 Rising edge Status register for the up to 24 muli keys, located on RAM address 224+27

Bit 23	....	Bit 1	Bit 0
Rise_edge_S24	....	Rise_edge_S2	Rise_edge_S1

4.4 Capacitive Switches (Inputs)

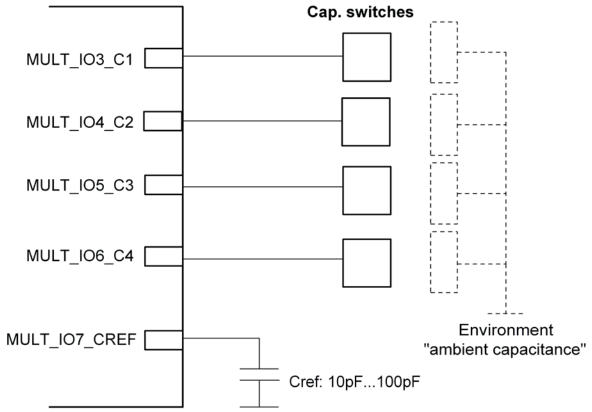
The PS09 offers the possibility to connect up to 4 capacitive switches. For this purpose up to 4 capacitors are connected to the multiple purpose pins 3 to 6 (MULT_IO3 to MULT_IO6) and a reference capacitor at MULT_IO7. The advantage is the very low current consumption of approx. 1µA at a high scanning frequency of the switches (39Hz or 78Hz).

NEW with PS09

The capacitance switch can be easily realized by forming one electrode of a capacitor on the PCB, where the other electrode is indirectly given by the environment. Then, any change in the ambient capacitance can be sensed, e.g. a finger is tapping on or approximating to the electrode on the PCB. Figure 4.2 illustrates the principle,

Background: For the sensing of the capacitors the internal measurement core of the PS09 is used. Similar to the strain gage measurement, where the resistors are discharged over one common capacitor (Cload), with the capacitive switches two capacitors are discharged over one common (internal) resistor and the ratio is calculated. If one of the capacitors changes, the ratio changes too and by defining a threshold a capacitive switch can be realized.

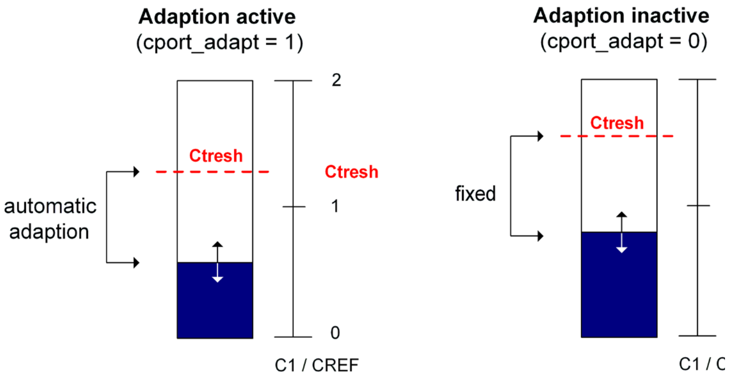
Figure 4.2: Principle of capacitive swwitching



4.4.1 Threshold Adaption

The threshold is a key parameter to configure the capacitive switches, as with this threshold it is decided when the capacitive switch is pressed or not. Basically there are two ways to set this threshold, absolute or relative. In the absolute method there is a fixed threshold defined and the ratio measurement compared to it (cport_adapt = 0). The other method is an adaptive one, where slight changes in the ratio are automatically tracked and adapted, so that the threshold is not applied absolutely but relative to this adapted equilibrium ratio (cport_adapt = 1).

Figure 4.3 Threshold adaption of the capacitive switches



4.4.2 Configuration

To configure the capacitive switches there is the configuration register 15 (Configreg_15). Depending on the previously described difference of an adaptive / non-adaptive threshold, the configuration register changes. Thus you will find two sets of settings in the configuration register description. The most important parameters are:

Table 4.7 Configuration parameters for capacitive switches

Parameter	**Description**
Cport_adapt	Defines whether the ratio shall adapted automatically or not
Cport_update	Update frequency of capacitive switches (39 or 78Hz)
Cport_r	Selects the size of the internal discharging resistor (25k, 50k, 100k or 200k)
Cport_en	Enables the capacitance switches bitwise (1 to 4)
Cport_thres	Sets the threshold when a “1” shall be signaled
Cport_adapt_speed	Selects the speed of the adaption (if cport_adapt = 1)

To use the capacitive switches the Mult_IO3_C1, Mult_IO4_C2, Mult_IO5_C3, Mult_IO6_C4 must be configured as input (Configreg_11). It is furthermore recommended to disable the input buffers of the digital I/Os when capacitive switches are used, thus to set io_en_digital[7:0] to 0 (Configreg_11).

Note: The use of capacitive switches and the UART simultaneously on the same pins is of course not supported. They can be used only mutually exclusively.

The reference capacitor is connected at Mult_IO7_Cref and should be approximately in the range of the other capacitors. A good value to start is 10pF.

Status

Once a threshold is defined, the PS09 compares the actual ratio of the capacitor in question to the reference capacitor and indicates whether the threshold was crossed or not. This is done by giving a status in the status register at RAM address 224+22. The relevant bits are illustrated as follows:

Table 4.8 Reading the Current status of the capacitive switches in RAM address 224+22

23	22	21	20	...	7	6	5	4	3	2	1	0
Status					Rising Edge on				Falling edge on			
Mult_IO6_C4	Mult_IO5_C3	Mult_IO4_C2	Mult_IO3_C1		Mult_IO6_C4	Mult_IO5_C3	Mult_IO4_C2	Mult_IO3_C1	Mult_IO6_C4	Mult_IO5_C3	Mult_IO4_C2	Mult_IO3_C1

4.5 SPI-Interface

Note: All described operations with SPI Interface are also available with IIC

4.5.1 Interfacing

The SPI interface is used to write the configuration and the program to the OTP (or respectively EEPROM for development purposes, see Ch. 6, Section 6.2.3). Furthermore there is a User EEPROM space to store calibration data which can also be addressed by the SPI interface.

Alternatively, the PS09 can be purely operated as converter chip by means of an external microcontroller.

The following illustrations show the several operational modes which are possible with PS09, thereby, the SPI interface has different functionality in each mode:

STAND ALONE

The configuration data and the program are stored in the OTP. The SPI interface is only needed once to program the OTP but has no function further on.

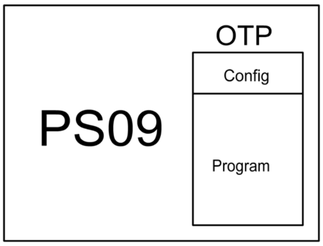
Mode pin has to be left unconnected in stand alone mode. When the SPI interface is not used, the pins of the SPI interface can be used as I/O ports.

Configreg_01: otp_pwr_cfg = 1 (loads configuration from OTP after power-on reset)

Configreg_01: otp_pwr_prg = 1 (after power-on reset user code available in OTP starting at address
48 is executed)

Configreg_01: otp_usr_prg = 1 (after end of measurement user code available in OTP starting at
address 48 is executed)

Figure 4.4: Stand alone mode



FRONT END (CONVERTER) MODE

The SPI interface is used for communication between a master (microcontroller) and the slave (PS09). The OTP need not be written in this case, then the configuration of PS09 is directly written by the microcontroller to the configuration registers in the RAM starting at address 48. Also the results are read by the microcontroller from the status and result registers starting at RAM address 240.

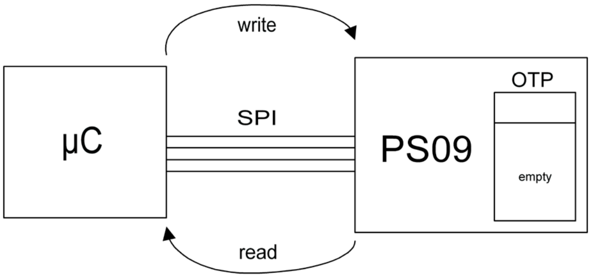
Mode pin is set to GND to activate the SPI interface for communication.

Configreg_01: otp_pwr_cfg = 0

Configreg_01: otp_pwr_prg = 0

Configreg_01: otp_usr_prg = 0

Figure 4.5: Frond end (Converter) Mode

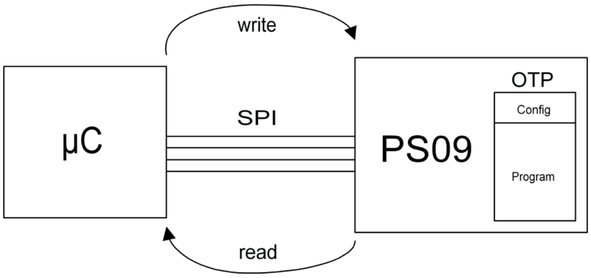


Refer to Section 4.5.3.5 on how to program the SPI master for an application using PS09 in this mode.

MIXED MODE

This mode combines the previous two modes by having a SPI communication between an external microcontroller and PS09, but also running a program in PS09. This is the case where some pre-processing is to be performed in PS09 before reading out the values by the external microcontroller. The Mode pin is set to GND to activate the SPI interface for communication.

Figure 4.6: Mixed Mode



Note: When this mode is used with otp_usr_prg = 1, the result registers from address 240 to 255 must be copied in the user code to RAM address 16 to 31. Additionally addresses 240 to 255 must be copied to RAM address 0 to 5.

Configreg_01: otp_usr_prg = 1

Building applications with multiple slaves (PS09) is also feasible, the selection of the individual chip is then done by SPI_CSN. Generally, before sending an opcode please send a positive pulse on the SPI_CSN line (to reset the interface).

4.5.2 SPI Timing

The following timing parameters describe the pure front end mode. This is the timing between an external microcontroller (µC) and the PS09. PS09 thereby supports only 1 mode out of 4 possible ones:

SPI Mode:

Clock Phase Bit = 1

Clock Polarity Bit = 0

Data transfer with the falling edge of the clock, clock starts from low.

Figure 4.7 SPI timing

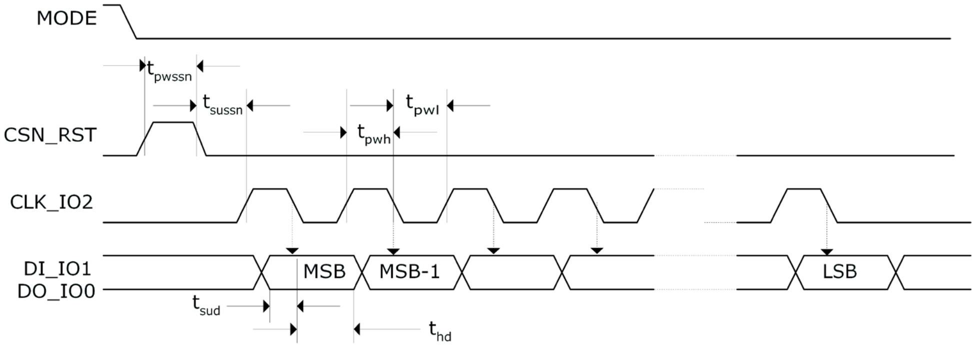


Table 4.7 SPI timing parameters

Time:	Description:	tmin [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.

NEW with PS09

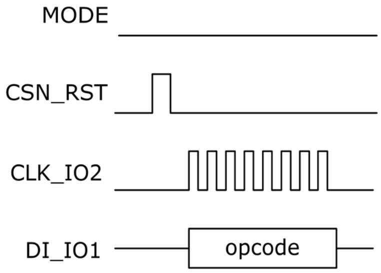
The SPI interface of the PS09 is activated by setting the mode pin (pin 21) to 0. This selection replaces the former SPI_ENA signal which was used with PS081. Please note, that after a change in the mode pin (e.g. from 1 to 0) it takes about 10ms for the SPI interface to become active.

4.5.3 SPI - Instructions

4.5.3.1 Single byte opcodes & RAM access

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	

Figure 4.8 Single Byte Opcode transmission

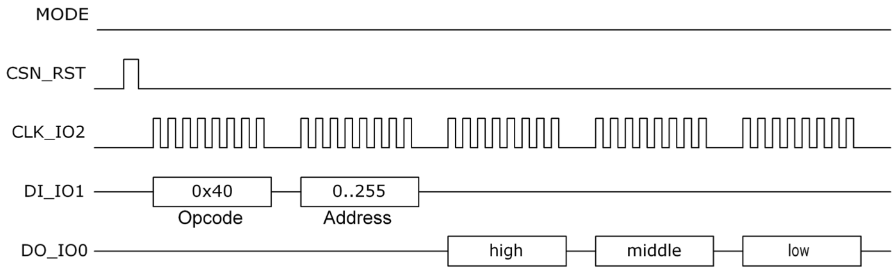


RAM Access

RAM Write	= ’b00000000	= ’h00
RAM Read	= ’b01000000	= ’h40

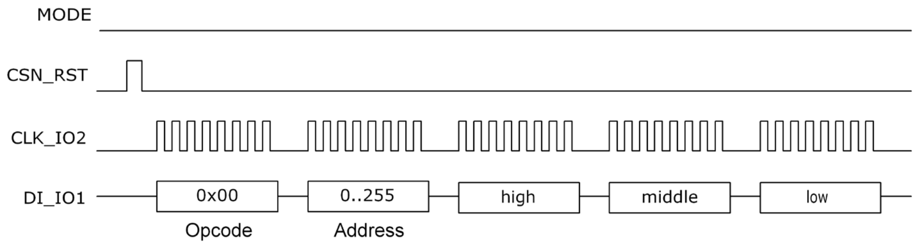
RAM Read Access

Figure 4.9 RAM read access



RAM Write Access

Figure 4.10 RAM write access



4.5.3.2 User EEPROM Access

EEprom_read	= 'b10100000	= 'hA0	(read single word)
EEprom_write	= 'b10100001	= 'hA1	(write single word)
EEprom_erase	= 'b10100010	= 'hA2	(erase single word)
EEprom_berase	= 'b10100100	= 'hA4	(erase whole block)

Single byte opcodes

EEprom_bgap_off	= 'b10000110	= 'h86
EEprom_bgap_on	= 'b10000111	= 'h87
EEprom_enable_off	= 'b10010000	= 'h90
EEprom_enable_on	= 'b10010001	= 'h91

Note: It is necessary to switch on the bandgap and to enable the access before writing to or reading from the User EEPROM (send EEprom_bgap_on and EEprom_enable_on). The User EEPROM is accessed byte-wise, where the block address is always 0 and the address ranges from 0 to 127.

Figure 4.11: User EEPROM Write Access

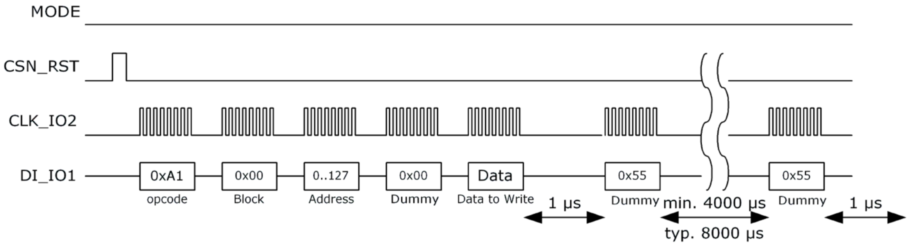
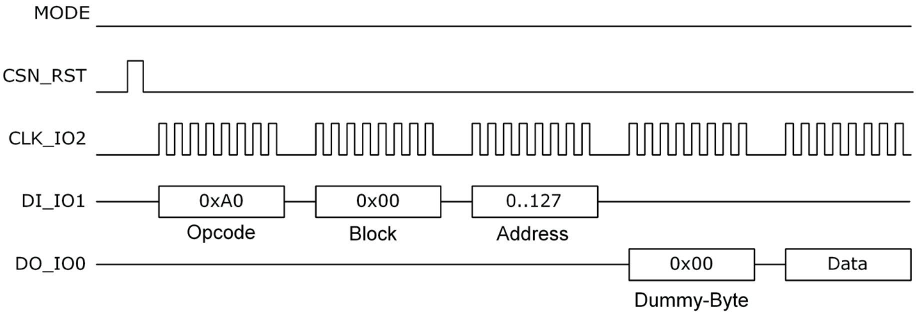


Figure 4.12: User EEPROM Read Access



4.5.3.3 OTP Access:

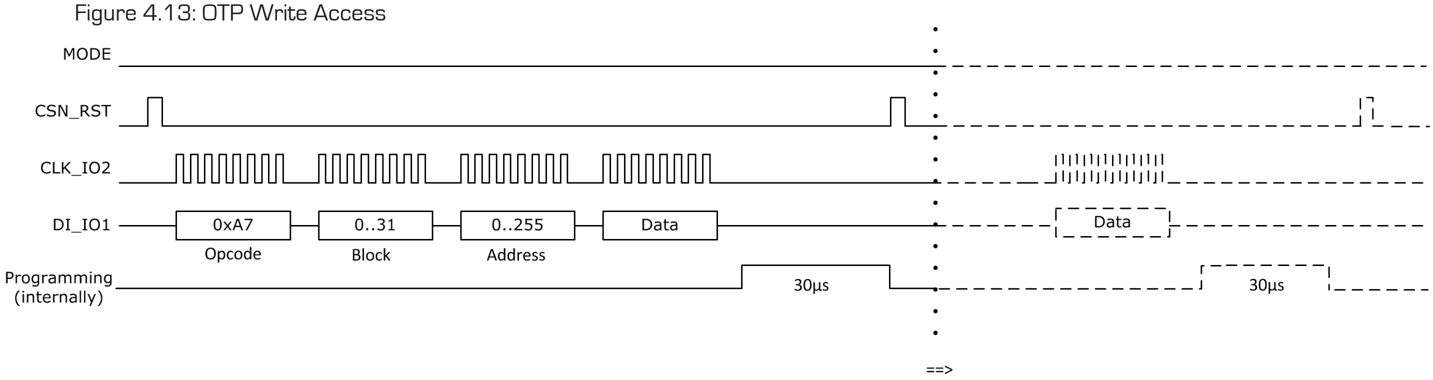
Otp_read	= 'b10100110 = 'hA6
Otp_write	= 'b10100111 = 'hA7

Single byte opcodes

Otp_enable_off	= 'b10101000 = 'hA8
Otp_enable_on	= 'b10101001 = 'hA9
Otp_prog_ena_off	= 'b10101100 = 'hAC
Otp_prog_ena_on	= 'b10101101 = 'hAD

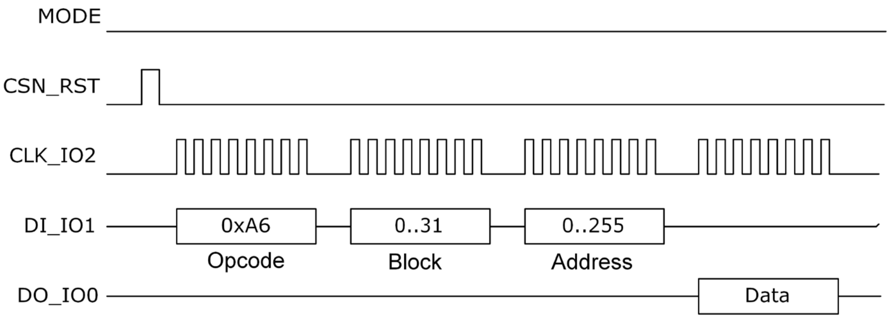
Note: To program the OTP it is necessary to switch on the OTP enable and the program enable (Otp_enable_on and Otp_prog_ena_on). To access the OTP or the external EEPROM alternatively please see also Chapter 6, section 6.2 Memory Organization.

Figure 4.13: OTP Write Access



The write access can be for a single byte only or for a number of bytes written sequentially one after each other (incremental write). For the latter, the internal programming time of approx. 30µs needs to be waited and then the next write sequence is initiated by toggling CSN.

Figure 4.14 OTP Read Access



4.5.3.4 External EEPROM Access

Ext_EEPROM_read = 'b01010000 = 'h50 (read single byte from external EEPROM)

Ext_EEPROM_write = 'b01010010 = 'h52 (write single byte to external EEPROM)

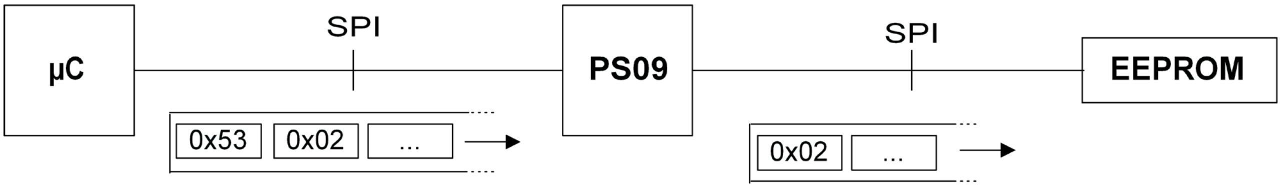
Ext_EEPROM_ena_and_opcode = 'b01010011 = 'h53 (sets pin 31, EE_CSN to 0 → enables EEPROM access and prepares EEPROM for the next opcode to receive)

Single byte opcodes

Ext_EEPROM_disable = 'b0101010x = 'h54 (sets pin 31, EE_CSN to 1 → disables EEPROM access)

An external EEPROM can be connected to the PS09 for developing the program which will then later be stored in the OTP. There are read/write opcodes available to access this external EEPROM by passing it through PS09. The opcodes for the EEPROM itself can vary according to different manufacturers of the external EEPROM. The following graphic illustrates the dependency:

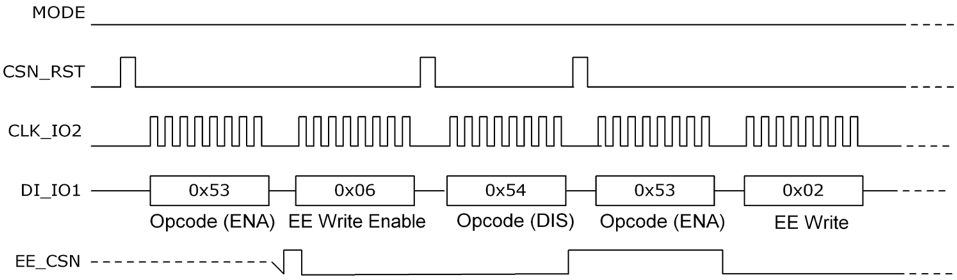
Figure 4.15: Connecting an external EEPROM



In this (simplified) example the opcode “0x53” for PS09 signals that an access to the external EEPROM shall be performed. Then, the write opcode for the EEPROM (here “0x02”, dependent on the EEPROM used) is sent to the external EEPROM along with the data.

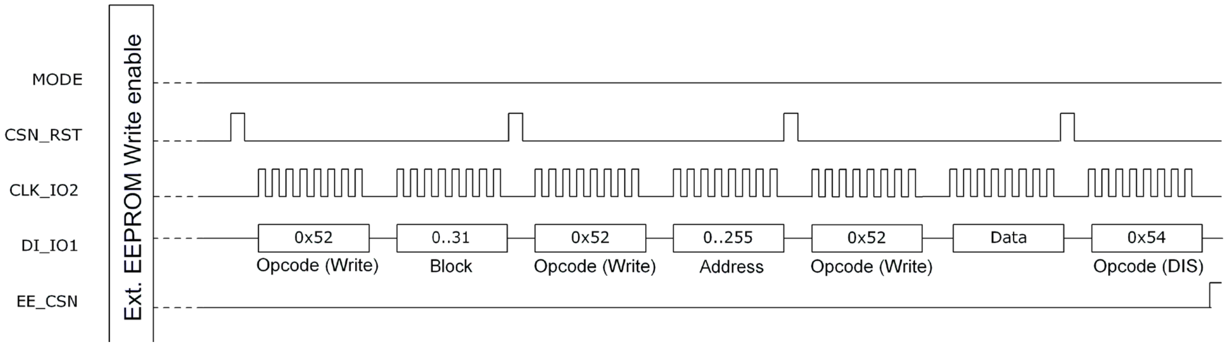
Please note: The illustrations in this data sheet relate to an EEPROM from Microchip M25AA640A. Although the basic access is illustrated, the developer normally doesn’t need to implement the access manually, as this is already implemented in the PicoProg programmer. In other words, for working with an external EEPROM, the PicoProg should be used for programming it.

Figure 4.16 Enable External EEPROM



The ENA opcode (Ext_EEPROM_ena_and_opcode) is used to prepare PS09 a control byte to the external EEPROM. The bytes “EE Write Enable, 0x06” or “EE Write, 0x02” are the instructions for a specific EEPROM type, here M25AA640A from Microchip was used.

Figure 4.17: External EEPROM Write Access



4.5.3.5 Program flow for Front-End Mode

Program start: Power Reset (0xF0)

Watchdog off (0x9E)

Configure PS09 in RAM: RAM Write (0x00) + address + 3 bytes (config data)

Write RAM address 48..63

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

Write RAM address 80, 84-90 for configuring UART (optionally)

Optional: Control read of RAM config

RAM Read (0x40) of RAM address 48..63

Start measurement: Init Reset (0xC0)

Start New Cycle (0xCC)

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

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

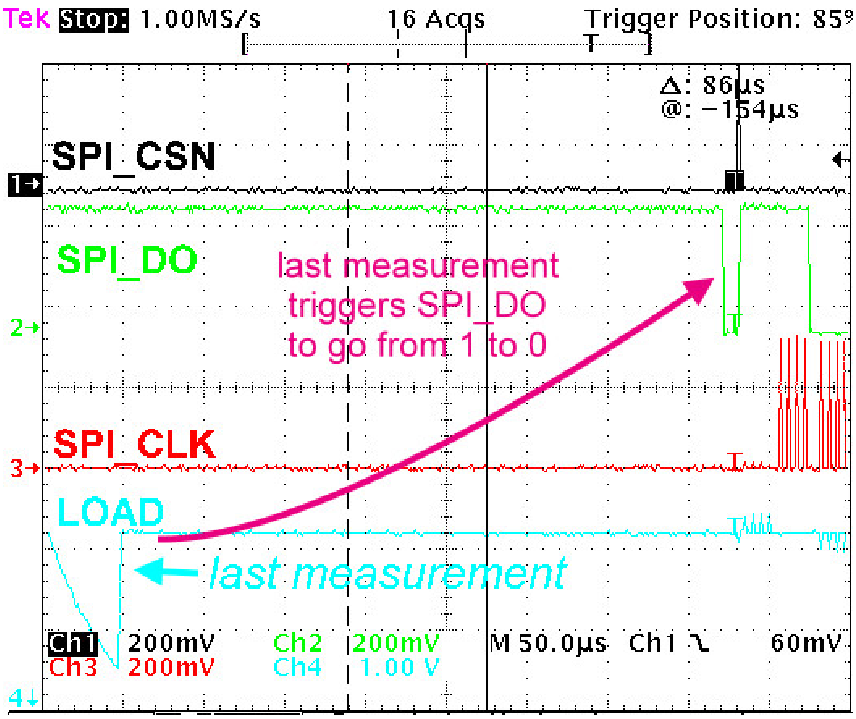
(this way SPI_DO is enabled for SPI communication):

Read HBO result at RAM address 0 (send SPI read opcode)

Annotations:

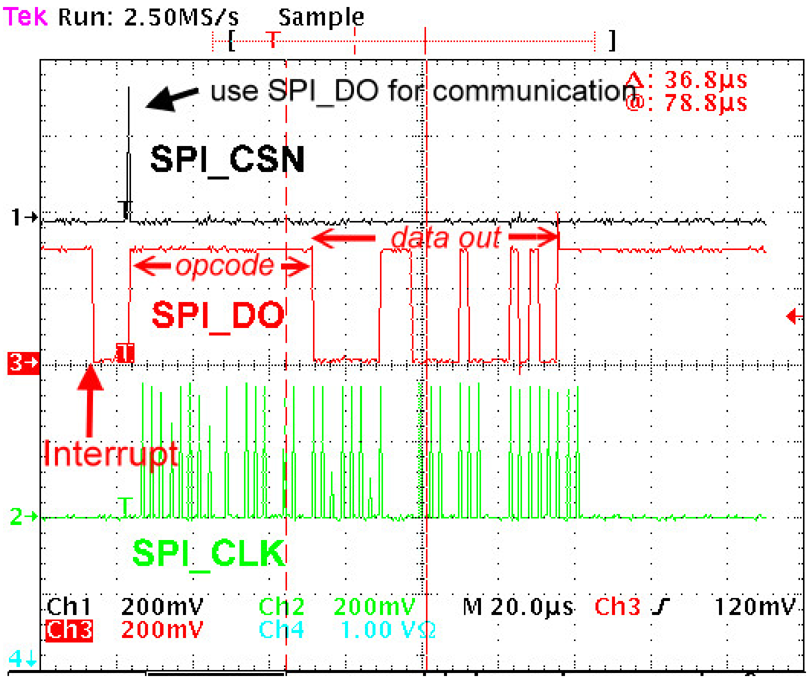
- In pure front-end mode make sure the external EEPROM was erased or the OTP unprogrammed respectively.
- In mixed mode a program is executed in the internal microprocessor in conjunction with the external microprocessor. In this case the OTP / external EEPROM contains a program. Please see chapter 4.5.1 SPI Interfacing for details of the mixed mode.
- 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, the interrupt can be configured to GPIO3 or GPIO4.
- 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)
- It is recommended to read the HBO result from RAM address 0. Reading from RAM address 244 (also HBO result) can result in an address 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 OTP / ext. 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 244 to address 0 manually!

Figure 4.18: Oscillograph of LOAD and SPI signals



In Figure 4.18 you can see that SPI_DO gives an interrupt after the last discharging cycle of the measurement was done.

Fig. 4.19: 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 PS09 (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 I2C Interface

NEW with PS09

The I2C interface is a serial 2 wire interface and newly introduced in the **PICO**STRAIN family with PS09. For IIC there are the various notations used in the literature:

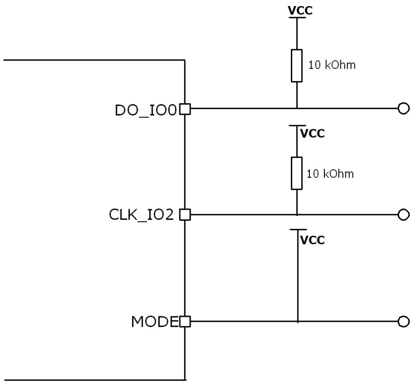
- IIC: Inter IC Bus
- I²C: same as above, Brand name from Philips®/NXP®
- TWI: Notation used by Atmel® for I2C

In PS09 either the SPI mode or the I2C can be used which is selected by the MODE pin (see former description). The I2C Interface can be used to write the program, configuration and calibration data into the OTP and also for pure front end mode between an external microcontroller and PS09. The PS09 can be addressed as an I2C slave device with a 7-bit device address, ’b1100000.

The I2C communication interface in the PS09 is built completely on the SPI Protocol interface that was presented in PS081. The only difference in the I2C interface is that the I2C master initially sends a byte with the 7-bit device address (‘b1100000) and the read/write direction (encoded in 1 bit sent after the address). All subsequent opcodes, addresses and data correspond to the SPI protocol. In other words, at the data layer level, the commands and opcodes to send correspond exactly to the SPI interface. But at the physical layer level, the transmitted bytes are sent in accordance with the I2C protocol.

For the PS09 to operate in I2C mode, the Mode pin of the PS09 must be connected directly to power supply, VCC. The pins are then used for the I2C interface and no longer as I/O ports. As per the I2C specification the CLK_IO2 and DO_IO0 lines must be pulled up. For the evaluation hardware and application, a pull up resistor of 10kOhm was observed to give acceptable performance and results. The resistor value must be suited to the application respectively.

Figure 4.20: I2C mode connection



4.6.1 I2C Timing

The timing described here is the I2C timing for operating the PS09 as a pure converter that communicates with an external microcontroller through the I2C interface. The clock and data lines are by default in high state. Data transfer is with the rising edge of clock. As mentioned earlier, the first byte for initiating the transaction (read or write) is the Device Address byte extended by 1 bit (LSB), e.g. for read the bit is 0 and therefore 'b1100000 becomes 'b11000000 ('hC0). For write the LS Bit is 1 and therefore device address byte is 'b11000001 ('hC1).

Figure 4.21: I2C timing

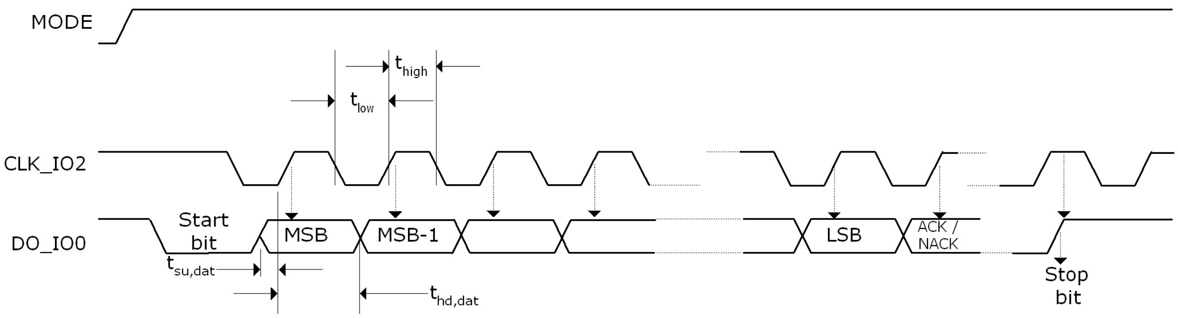


Table 4.8: I2C timing parameters

Symbol	Description	Value ttyp
t_{high}	Pulse width SCK high	4 us
t_{low}	Pulse width SCK low	4.7 us
$t_{su,dat}$	Setup time data	30 ns
$t_{hd,dat}$	Hold time data	30 ns

4.6.2 I2C Instructions

The instructions used in the I2C interface are the same as the SPI Instructions. Only the IIC header byte, i.e. the device address and the direction bit must be sent in advance.

4.6.2.1 RAM Access & Single Byte Opcodes with IIC

RAM access

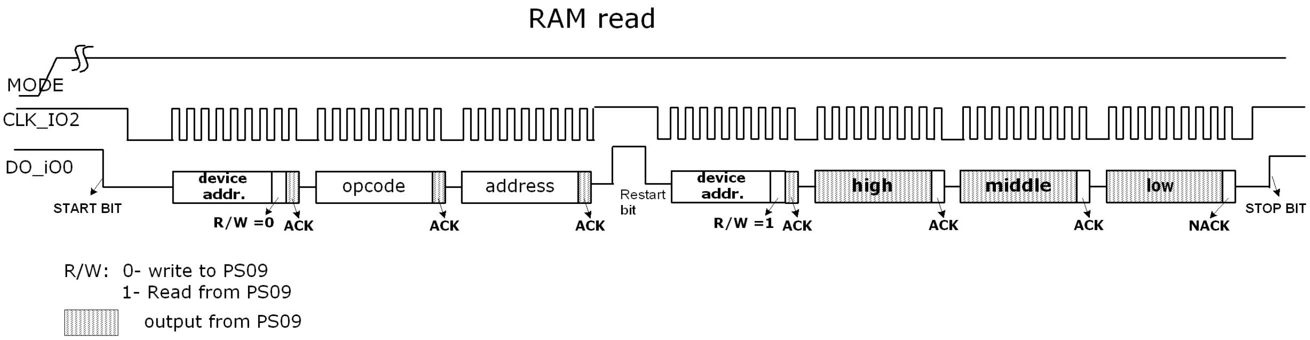
RAM Write	= 'b00000000	= 'h00
RAM Read	= 'b01000000	= 'h40

Single byte opcodes

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	

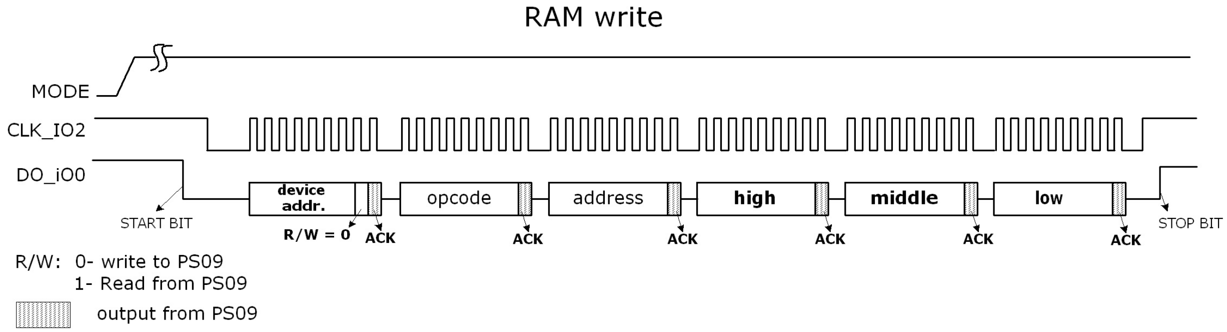
RAM Read Access

Figure 4.22: I2C RAM Read



RAM Write Access:

Figure 4.23: I2C RAM Write



4.6.2.2 User EEPROM Access with IIC

EEprom_read	= 'b10100000	= 'hA0	(read single word)
EEprom_write	= 'b10100001	= 'hA1	(write single word)
EEprom_erase	= 'b10100010	= 'hA2	(erase single word)
EEprom_berase	= 'b10100100	= 'hA4	(erase whole block)

Single byte opcodes

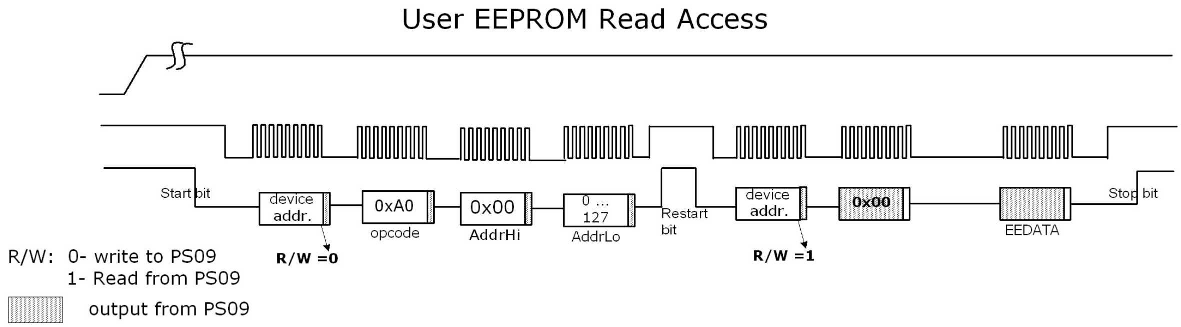
EEprom_bgap_off	= 'b10000110	= 'h86
EEprom_bgap_on	= 'b10000111	= 'h87
EEprom_enable_off	= 'b10010000	= 'h90
EEprom_enable_on	= 'b10010001	= 'h91

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

User-EEPROM Read Access

The user EEPROM is PS09 is 128 bytes in size. It is possible to read from the User EEPROM by means of opcode 0xA0. This opcode reads back 2 bytes of data with the address specified; the first byte which is 0x00 must be ignored. The second byte is the valid 8 bit data.

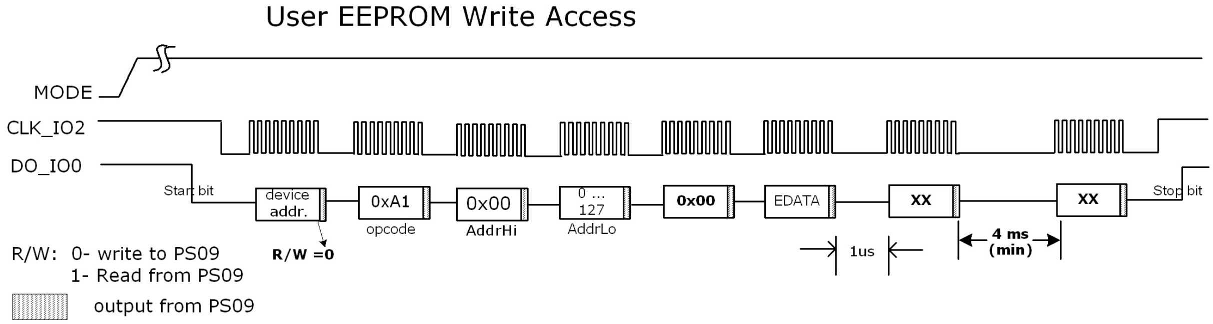
Figure 4.24: User-EEPROM Read access



User-EEPROM Write Access

The User EEPROM can be read through the IIC interface, similar to reading a RAM cell. The SPI instruction to write to the User EEPROM is 0xA1.

Figure 4.25: I2C User-EEPROM Write access



User-EEPROM Erase

The User EEPROM can be erased byte-wise or completely through the I2C interface; the sequence is very similar to EEPROM write show above. The opcode to erase a single byte from the User EEPROM is 0xA2. Except for the opcode, the sequence is same as shown above for write access. The data bytes are ignored and the specified address content is erased to 0x00.

If the complete User EEPROM ought to be erased, the opcode used is 0xA4 and the same sequence as shown above for Write access is adopted. The address and data bytes are ignored and the complete EEPROM is erased.

4.6.2.3 OTP / external EEPROM Access with IIC

Otp_read	= 'b10100110 = 'hA6
Otp_write	= 'b10100111 = 'hA7

Figure 4.26 OTP Write access via I2C

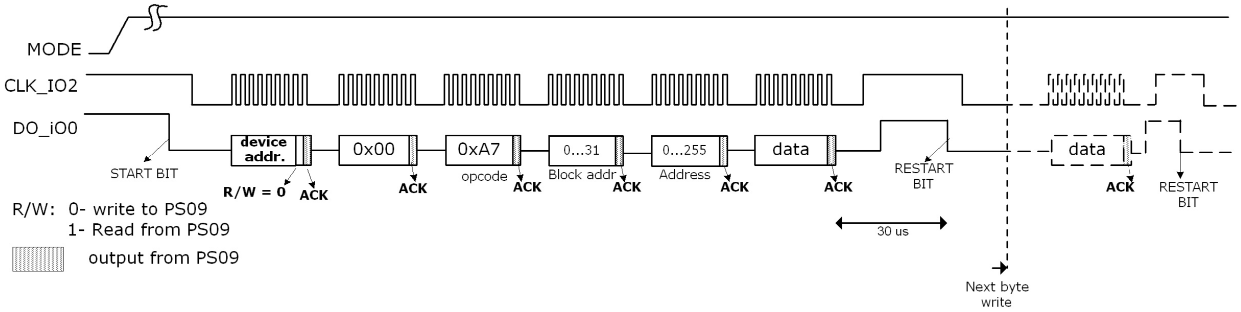
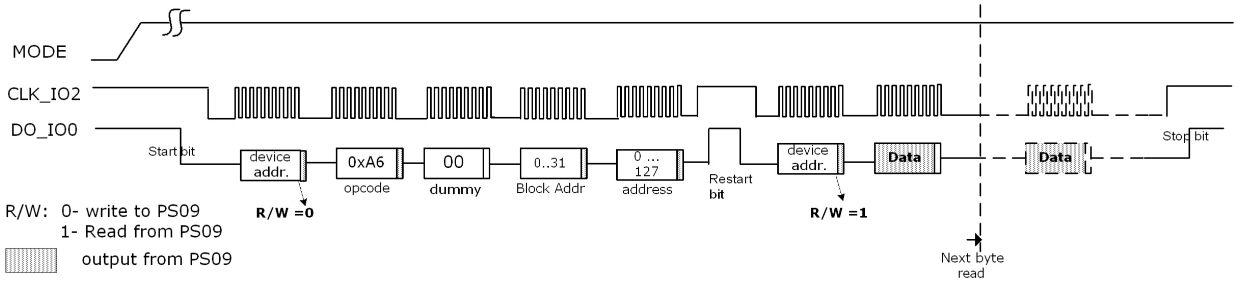


Figure 4.27 OTP Read access via I2C



Single byte opcodes

Otp_enable_off	= 'b10101000 = 'hA8
Otp_enable_on	= 'b10101001 = 'hA9
Otp_prog_ena_off	= 'b10101100 = 'hAC
Otp_prog_ena_on	= 'b10101101 = 'hAD

Note: To program the OTP it is necessary to switch on the OTP enable and the program enable (Otp_enable_on and Otp_prog_ena_on).

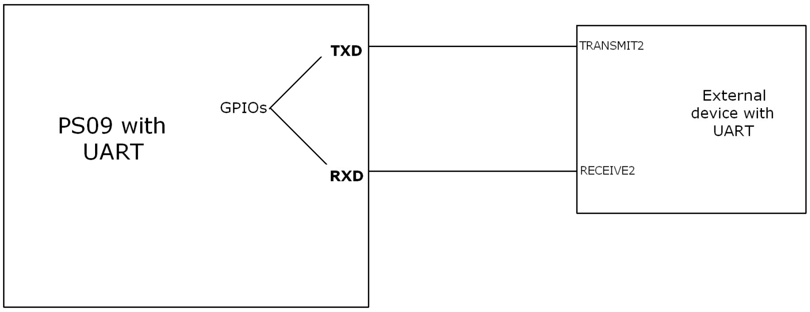
The external EEPROM can be accessed in a similar manner through the IIC Interface. The opcodes used for this access are same as with SPI interface (Please refer Chapter 4.5.3.4 for the opcodes)

4.7 UART

NEW with PS09

UART is an abbreviation for **U**niversal **A**synchronous **R**eceiver **T**ransmitter, is a serial communication interface. The transmitter part of the UART can take in bytes of data, and send the data through a Transmit line serially. The receiver part of the UART receives serial data on its Receive line input and assembles bytes of data.

Figure 4.26: Block diagram UART

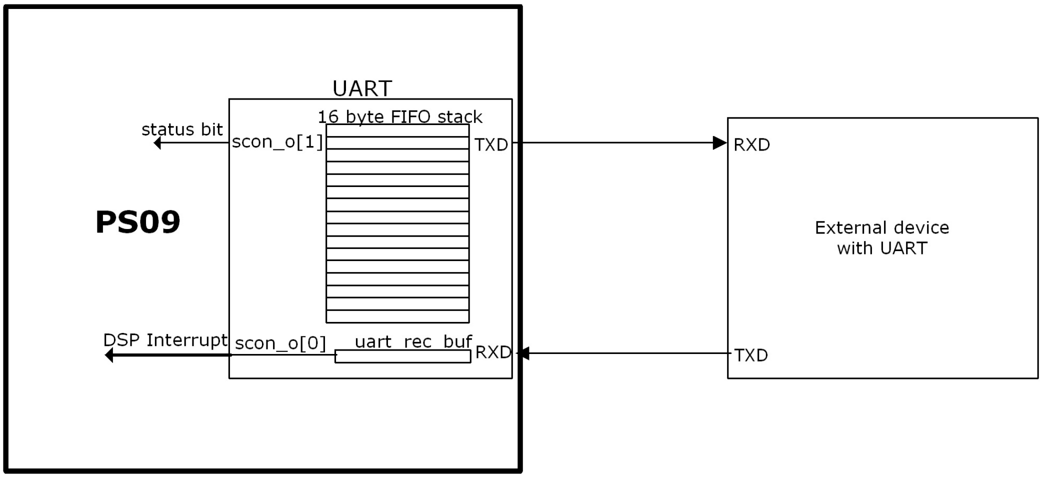


The PS09 has a built in UART module that can be used to perform transmission and reception to and from an external device with a UART interface, using the GPIO / MULT_IO pins. The TXD and RXD lines shown in the figure are the transmit and receive lines of the PS09 respectively. These can be realized using the GPIO pins (Mult_IO3 or 4 for transmission and Mult_IO2 to Mult_IO5 for reception).

4.7.1 Features of the UART

The UART module in the PS09 runs on the 4 MHz oscillator clock. Optionally, a 3.6864 MHz oscillator is also supported. It supports most standard baud rates from the slowest 300 baud to the fastest being 115200 baud. The transmit and receive modules both operate on the same baud rate selected.

Figure 4.27 UART architecture



The UART can transmit up to 16 bytes per transaction. The number of bytes to be transmitted has to be initialized. The UART has a built-in transmit FIFO stack that is to be initialized with the bytes to be transmitted. The transmit operation is started by setting and clearing a bit called uart_trans (Configreg_91, bit 10). The UART sends the assigned number of bytes through the selected GPIO pin. On completion of a transmit transaction, the transmit interrupt bit scon<1> (Configreg_80) is set. This bit can be polled by the user program. Further details on how to configure the transmit module can be found in the section Configuring UART for transmission.

The UART receives one byte at a time. When a new byte is received by the UART from an external device, the receive interrupt scon<0> (Configreg_80) bit is set. This receive-interrupt signal interrupts the DSP, and hence user code can be written to service it. This interrupt to the DSP can be enabled or disabled. When a new byte comes on the RXD line, the previously received byte is overwritten. Hence a received byte must be read by the processor immediately after receiving it. Further details on how to configure the Receive-module and its interrupt can be found in section Configuring the UART for reception.

4.7.2 UART Modes

The UART can basically operate in 2 modes, Mode0 and Mode1.

4.7.2.1 Mode 0

Mode0 is a 9 bit mode where a packet of data for transmitting and receiving is of the format

Start Bit	8 Data bits	Stop bit
0	D7 D6 D5 D4 D3 D2 D1 D0	1

4.7.2.2 Mode 1

Mode1: is a 10 bit mode where the data packet has an additional parity bit. The parity of the data to be transmitted can be chosen to be even or odd parity. During reception, the parity of the received data is evaluated and updated in the status_uart_rx_data_par - bit.

Start Bit	8 Data bits	Parity bit	Stop bit
0	D7 D6 D5 D4 D3 D2 D1 D0	P	1

In both of the above modes, the start bit is always a 0. In mode0, 8 bits of data are sent after the start bit, in mode1, the 8 bits of data and the 9th parity bit are sent. At the end of transmission a stop bit is always sent.

A receive transaction is triggered on the arrival of a start bit on the RXD line. In mode 0, 8 bits after the start bit are assembled as the data byte. The data byte is valid only if the Stop bit arrives after the 8th data bit. This checking for the validity of the stop bit can be switched off by clearing the uart_auto_det_stop configuration bit to 0. In mode 1, 8 bits of data after the start bit are stored as the received data byte. Again a valid stop bit is checked for; this can be optionally switched off as mentioned above. In both the modes, the status of the 9th received bit is readable from scon<2> (Configreg_80). For details about the parity bit in mode 1, see the following section Receive Interrupt and Status Bits.

4.7.3 Configuration of the UART

4.7.3.1 Basic configuration

The following configurations need to be performed to use the UART, irrespective of whether it is going to be used for transmission or reception.

1. The 4MHz clock has to be enabled to the UART module in order to work with the UART. This is done by setting *uart_clk_en* to 1 (Configreg_91).
2. Depending on whether the 4MHz clock is used or a special 3.6864 MHz oscillator, the *uart_4Mhz_divider* bit (Configreg_91) has to be set or cleared respectively.
3. The mode of operation of the UART is chosen by setting/clearing the *uart_mode bit* (Configreg_91) appropriately.
4. The operating baud rate for the UART needs to be selected. The following table shows the settings to be made for different baud rates.

Table 4.9: Supported baud rates

Baud rate	**UART_baud_rate settings with 3.6864MHz or 4 MHz clock**
300	0
600	1
1200	2
2400	3
4800	4
9600	5
19200	6
38400	7
57600	8
76800	9
115200	10

4.7.3.2 Configuring UART for transmission

After the general configurations for the UART stated above, the following are to be performed to configure the UART for transmission.

1. The GPIO3 pin or the GPIO4 pin can be selected to function as the transmit line of the UART. To select the GPIO3 pin as the TXD line, the *multio3_sel* bits (Configreg_12) have to be configured to 2, to select the GPIO4 pin , the *multio4_sel* bits (Configreg_13) have be configured to 2.
2. After selecting the GPIO3 or GPIO4, the respective pin must be configured as outputs, by clearing either the *io_en_3_mio* bits or the *io_en_4* bits (both Configreg_11) appropriately.

3. If the selected mode is Mode1, the parity to be used during transmission can be chosen to be either even or odd parity with the *uart_par* bit (Configreg_91).
   0: Even parity – Even number of 1s in the transmitted byte
   1: Odd parity - Odd number of 1s in the transmitted byte
4. The number of bytes to be transmitted in one transaction is to be initialized in the bits *uart_tx_cnt* (Configreg_91).
5. The bytes to be transmitted are filled in the FIFO order in the bytes *uart_sbuf_i0-15* (Configreg_84, 86-90).
6. Finally the *uart_en* bit (Configreg_3) is set to 1. There is a delay of typically 50us from setting the *uart_en* bit to 1, until the uart module recovers from the reset state. This delay has to be accounted for in software.
7. The transmission is activated by setting the *uart_trans* bit (Configreg_91) to 1, and then clearing it to 0. This generates the start pulse for transmission.
8. Once the transmission is complete the UART indicates the completion by setting the transmit interrupt *scon <1>* bit (Configreg_80). The user program can poll this status bit to wait for the end of transmission.
9. For the next transmission with the same baud rate, the *uart_tx_cnt* (Configreg_91) has to be re-initialized, the new *uart_sbuf_i0-15* bytes have to be updated and the transmission has to be started as in step 7.

4.7.3.3 Configuring UART for reception

After the general configurations for the UART stated above, the following are to be performed to configure the UART for reception.

1. One of the 4 pins GPIO2, GPIO3, GPIO4 or GPIO5 can be configured to act as the RXD input pin to the UART. The selection is done by setting the *uart_rdx_sel[1:0]* bits (Configreg_91) to 0 = GPIO2, 1 = GPIO3, 2 = GPIO4 or 3 = GPIO5 respectively.
2. Depending on the GPIO selected as the RXD line, the respective *io_en* (Configreg_11) bit have to set to 3 and the respective *io_en_digital* (Configreg_11) have to be set to 1.
3. The UART receive module can be configured to automatically detect a stop bit at the end of the received byte. This is done by setting the bit *uart_auto_det_stop* (Configreg_91) to 1. If the stop bit need not be checked for after receiving a data byte, this bit has to be cleared to 0.
4. The receive module generates an interrupt to the DSP after receiving a byte. According to the demands of some systems, the interrupt could be generated to the DSP only based on the status of the 9th bit received. This is achieved by setting the *uart_mpcomm* (Configreg_91) to 1.
   a. If mode 1 is selected and if *uart_mpcomm* =1, the receive interrupt is set only when the 9th received data bit i.e. the parity bit is 1.
   b. If mode 0 is selected and if *uart_mpcomm* = 1, the receive interrupt is set only when the received stop bit is a valid 1.
   c. If the *uart_mpcomm* = 0, the receive interrupt to the DSP is generated unconditionally on the recep-

tion of a new byte of data.

5. The receive interrupt to the DSP can be globally enabled or disabled by configuring the *irq_uart_en* bit (Configreg_91). Like in transmission, the user program can also poll the receive interrupt scon<0> (Configreg_80) to check if a new byte has been received.
6. Finally the UART must be enabled for reception by setting the *uart_rec_en* bit (Configreg_91) to 1

4.7.3.4 Receive Interrupt and Status bits

A received byte is indicated in the status bits and an interrupt is set accordingly. In the following there are some recommendations how to react then in the DSP software:

1. During reception, after each byte of data has been received, the receive interrupt, scon<0> (Configreg_80) is set. This bit is cleared by writing a 1 to *uart_rec_int_ack* bit (Configreg_91). Further new incoming bytes can be received only if this bit is cleared.
2. Hence the user program (ISR) must react quick enough to read the first byte of data as soon as the receive interrupt is generated to the DSP. However the UART can be signaled not to receive any further bytes from the external UART device, by setting the *uart_rec_int_dis* (Configreg_91) to 1.
3. The received byte can be read from *uart_rec_buf* bits (Configreg_80).
4. The status of the 9th received bit, which is the stop bit in mode0 or the parity bit in mode1, is readable from parity status bit scon<2> (Configreg_80).
5. The parity of the 8 data bits received by the UART is evaluated by the UART itself and shown in *status_uart_rx_data_par* bit (Configreg_80).
6. 0: indicates an even parity, i.e. even number of 1s in the received data byte
7. 1: indicates an odd parity, i.e. odd number of 1s in the received data byte
8. The status of the start bit and stop bit corresponding to the last received byte in the UART is shown in bits *status_uart_start* and *status_uart_stop* (Configreg_80).

4.8 Driving an External LCD Controller

With the PS09, using the Multipurpose I/O pins, one can establish an SPI communication manually and operate an external LCD driver like the Holtek HT1620 to display values on an external LCD. The PS09 can be programmed to act as a simplified SPI master where the external LCD driver then can be addressed as an SPI slave.

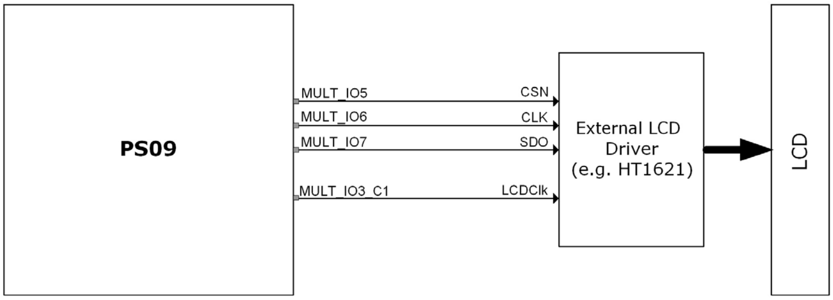
To use the PS09 with an external LCD driver, a minimum of 3 Multipurpose I/O pins are needed for the SPI communication part. Further, the LCD driver needs a clock source to operate that is generally made available from an external oscillator. The PS09 offers the possibility to generate a clock for the external LCD driver which is provided by another Multipurpose I/O pin (either Mult_IO3 or Mult_IO4). Thus, a maximum of 4 I/O pins are needed for the implementation of the simple SPI master.

Basically all Mult_IOs can be used to implement the simple SPI master. However, since Mult_IO0 to Mult_IO2 are used for the (SPI) interfacing from an external microcontroller to the PS09 it is recommended to use from Mult_IO3 upwards for the SPI master implementation. The configuration could be like the following:

Table 4.11. Pin assignment for connecting an external LCD driver

Mult_IO3 or Mult_IO4	LCD_CLK
Mult_IO5	SPI_CSN
Mult_IO6	SPI_CLK
Mult_IO7	SPI-DO

Figure 4.28: Connection Diagram for an external LCD Driver



Configuring PS09 as simple SPI master

1. Three Multipurpose I/Os are needed to communicate with the LCD Driver, they are Chip Select (CSN), the Clock (CLK) and the Data-out (SDO). Although the I/Os can be selected arbitrarily we recommend to configure them on Mult_IO5 upwards because then the ordinary SPI interface of PS09 can be used in parallel, e.g. to access PS09 by an external microcontroller.
2. All the 3 IOs are configured to act as outputs by clearing the io_en_5 to io_en_7 bits in Configreg_11.
3. The state on the output lines 1 or 0 is controlled by the program by setting and clearing the respective io_a bits in Configreg_00.

Note: When capacitive switches are also to be used with an external LCD driver, then connect the LCD driver on I/O0-2 to avoid conflict with the capacitive keys. However, when the LCd driver is connected to I/O0-2, it clashes with the SPI interface pins to operate the PS09 in frontend mode, hence program debugging is difficult.
Connecting the external LCD driver on I/O5-7 is the most comfortable option for debugging the user program.

Providing the Clock to the external LCD Driver

1. The PS09 generates the clock of 32 kHz for the external LCD, which is internally derived from the 10 kHz clock.
2. Either the IO3 or IO4 pin can be selected as the LCD clock output from the chip, by setting the multio3_sel bit (Configreg_12) to 3 or multio4_sel bits (Configreg_13) to 3. The respective selected pin must be naturally configured as output (io_en_3 or io_en_4 in Configreg_11).
3. The frequency of the clock can be trimmed in a limited range by adjusting the osz10kHz_fsoc bits in Configreg_00.
4. The polarity of this clock, i.e. generation of high pulses or low pulses can be controlled by using the lcd_clk_pol bit in Configreg_12.
5. The width of the pulses on the LCD clock is also programmable using the lcd_clk_sel bits in Configreg_12. Pulse widths of 100ns, 200ns and 800ns can be programmed.
6. There is a possibility to generate an open drain clock on the IO3 or IO4 pin by setting the lcd_clk_open_drain

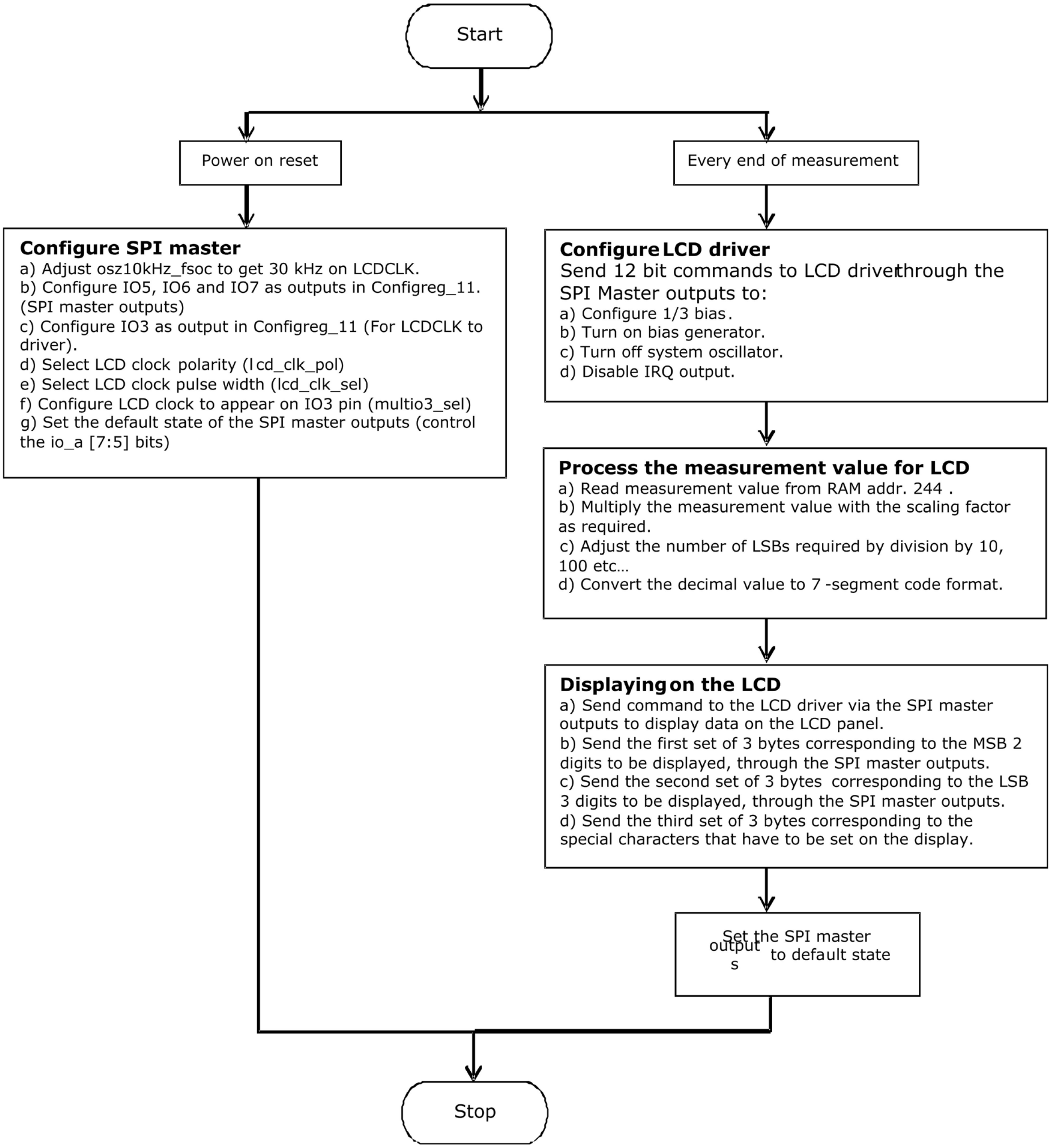
(Configreg_12) to 1, i.e. a clock that switches between High-z and 0, or High-z and 1. The High-z on the IO3 or IO4 has to be pulled up or pulled down externally to VCC or 0 therefore.This option is based on experiments to supply the HT1620 directly from the stabilized 1.8V core voltages for the implementation of a suitable level shifting.

7. The LCD Clock output from the chip should be connected to the OSCO pin of the HT1620 for the driver to function.

Programming the external LCD driver interface

By programming the PS09, the measurement value measured by the chip can be displayed on an external LC Display by controlling the external LCD driver (e.g. Holtek HT1620) by a user program. The following flowchart shows the algorithm to be used in the user code to control the HT1620 LCD driver. Please note that the LCD Clock output from the chip should be connected to the OSCO pin of the HT1620 for the driver to function.

Figure 4.29: Flowchart example, to display using the Holtek HT1620 driver on a LCD



Configure SPI master
a) Adjust osz10kHz_fsoc to get 30 kHz on LCDCLK.
b) Configure IO5, IO6 and IO7 as outputs in Configreg_11. (SPI master outputs)
c) Configure IO3 as output in Configreg_11 (For LCDCLK to driver).
d) Select LCD clock polarity (lcd_clk_pol)
e) Select LCD clock pulse width (lcd_clk_sel)
f) Configure LCD clock to appear on IO3 pin (multio3_sel)
g) Set the default state of the SPI master outputs (control the io_a [7:5] bits)

Configure LCD driver
Send 12 bit commands to LCD driver through the SPI Master outputs to:
a) Configure 1/3 bias.
b) Turn on bias generator.
c) Turn off system oscillator.
d) Disable IRQ output.

Process the measurement value for LCD
a) Read measurement value from RAM addr. 244 .
b) Multiply the measurement value with the scaling factor as required.
c) Adjust the number of LSBs required by division by 10, 100 etc…
d) Convert the decimal value to 7 -segment code format.

Displaying on the LCD
a) Send command to the LCD driver via the SPI master outputs to display data on the LCD panel.
b) Send the first set of 3 bytes corresponding to the MSB 2 digits to be displayed, through the SPI master outputs.
c) Send the second set of 3 bytes corresponding to the LSB 3 digits to be displayed, through the SPI master outputs.
d) Send the third set of 3 bytes corresponding to the special characters that have to be set on the display.

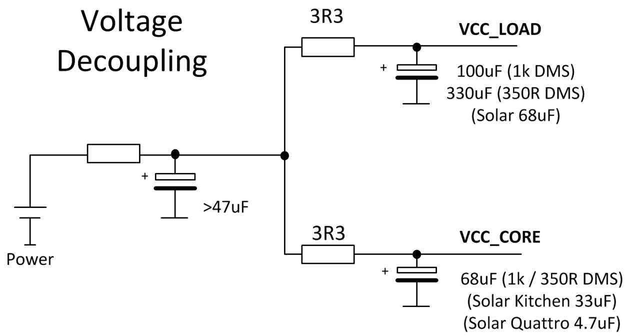
Set the SPI master outputs to default state

4.9 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 and VCC_CORE. It is necessary to feed them with voltages decoupled by low-pass filters. Further, those pins need sufficient blocking capacitance mounted close to the chip.

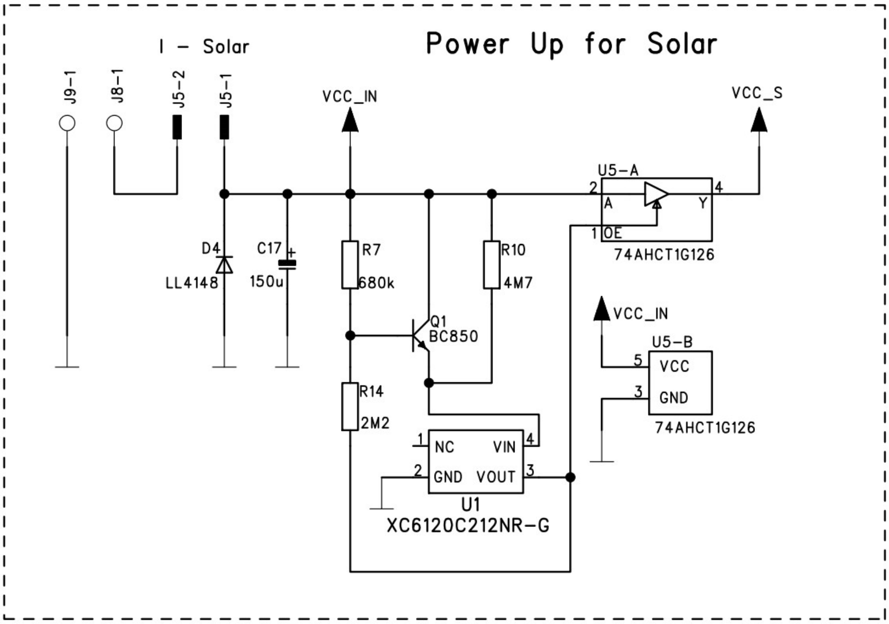
Figure 4.21 Power supply



In case of solar applications without any additional battery it is necessary to implement a power-up circuit. It provides a good start-up behavior 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 an optimized power-up circuit for solar body scales like it was used with PS081. To start up the scale it needs only about 3 µA @ 3.6 V. For other solar applications and with the use of PS09 it might be necessary to change some values of the components.

Figure 4.22 Power-up Circuit



Functional description of the function Power-up circuit

Coming from total darkness, all capacitors are discharged and the output of U5 is high-Z. The input voltage of U5 is zero. PS09 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,U1) 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. U5 is supplied with 3.5 V and regulates to 2.5 V for the PS09. PS09 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 U5 cannot regulate 2.5 V for the PS09. 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 at least one measurement once it comes to the regulation of the voltage.

4.9.1 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)

Table 4.12 LDO Recommendations

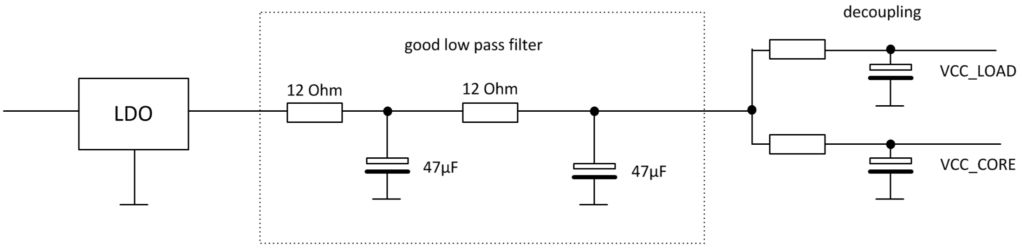
LDO	Features	Low-pass filter	Applications
Torex XC2206	medium noise, low current	use good low-pass filter	Solar
Micrel MC5205	medium noise, cheap solution	use medium low-pass filter	Low-cost solutions
Linear LT1761-BYP	very low noise, costly solution	use standard low-pass filter	High-end applications
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.

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

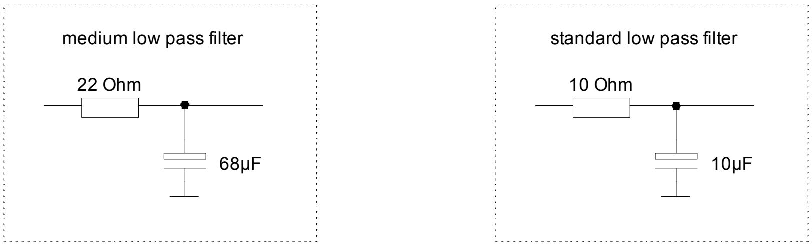
The additional low-pass filtering can help to reduce the noise getting through to the chip. Different types of low-pass filters can be used before decoupling the voltages:

Figure 4.23: Decoupling



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 PS09 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 4.24 Low-pass-filters



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.

4.9.2 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 249, 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]:

Table 4.13: Low batt configuration

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

Note: The default recommended setting (10) for bandap_tr bits in Configreg_14 must be strictly followed for the voltage detection to be accurate.

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.

Table of Contents Page

5 Configuration Registers

PSØ9 has 16 configuration registers of 24 Bit width, to be addressed in the RAM from address 48 to 63. The configuration registers control the whole chip including the strain measurement, the capacitive switches and the basic settings for the UART interface. The configuration settings are mirrored from the lower bytes 0 to 47 of the OTP or EEPROM (only for development) to the RAM.

The UART of PSØ9 is administered by the RAM cells 80, 84, 86 to 91. These cells are NOT mirrored from the OTP/EEPROM, but are only configurable directly in the RAM.

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	OTP / EEPROM bytes		
Configreg_00	48	2	1	0
Configreg_01	49	5	4	3
Configreg_02	50	8	7	6
Configreg_03	51	11	10	9
Configreg_04	52	14	13	12
Configreg_05	53	17	16	15
Configreg_06	54	20	19	18
Configreg_07	55	23	22	21
Configreg_08	56	26	25	24
Configreg_09	57	29	28	27
Configreg_10	58	32	31	30
Configreg_11	59	35	34	33
Configreg_12	60	38	37	36
Configreg_13	61	41	40	39
Configreg_14	62	44	43	42
Configreg_15	63	47	46	45

Internal microprocessor:

Configuration of registers is done in lower bytes (0 to 47) of the OTP / EEPROM and then mirrored to the RAM.

External microprocessor:

The OTP is not used / not programmed. Configure directly the RAM addresses and set the parameters otp_pwr_cfg, otp_pwr_prg and otp_usr_cfg to 0 (Configreg_01, Bits[2:0]).

5.2 Alphanumeric listing of configuration parameters

Table 5.2 Alphanumeric listing of configuration Parameters (including the UART configuration bits, RAM cells 80, 84, 86 to 91)

Bit name	Config Register	Bit position	Recommended Value (decimal)
adj_hr	01	22:19	5
alupernopen	14	03	0
auto10k	02	3	0
avrate	02	23:14	-
bandgap_trim	14	22:19	10
bridge	03	1:0	-
calcor	10	23:16	0
caltime	14	2:1	1
con_comp	12	19:18	1
connect_vcc_rosc	00	00	0
cport_adapt[2]	15	23	1
cport_en[2]	15	19:16	0
cport_r[2]	15	21:20	0
cport_thresh1[2]	15	7:0	-
cport_thresh2[2]	15	15:8	-
cport_update[2]	15	22	0
cpu_speed	00	2:1	1
crf_sen2	13	21	0
crf_sen3	13	22	0
crf_tau	13	18	0
crf_tp1	13	20	0
cytime	02	13:4	-
dis_noise4	03	14	0
dis_pp_cycle_mod	12	21	1
dis_wheat_pp	01	3	0
en_avcal	01	9	0
en_emi_meas_gain_comp	01	23	1
en_emi_noise_reduction	12	07	1
en_gain	01	7	1
en_sdel_noise	03	10	0
en_TkPar	01	11	0
en_wheatstone	03	21	-
ext_eeprom_clk_speed	12	17:16	1
force_quattro_mode	03	16	default: 0

Bit name	Config Register	Bit position	Recommended Value (decimal)
gain_comp	10	7:0	164
integrated_rspan	01	8	-
io_a	00	15:08	-
io_en_0_sdo	11	01:00	-
io_en_1_sdi	11	03:02	-
io_en_2_sck	11	05:04	-
io_en_3_mio	11	07:06	-
io_en_4_mio	11	09:08	-
io_en_5_mio	11	11:10	-
io_en_6_mio	11	13:12	-
io_en_7_mio	11	15:14	-
io_en_digital	11	23:16	default: 1 (digital use) if mult_io matrix or capacitance switches = 0
irq_dsp_edge	12	15	0
irq_dsp_en	12	14	-
irq_dsp_pin_sel	12	23	0
irq_uart_en	91	23	-
lcd_clk_open_drain	12	6	0
lcd_clk_pol	12	22	0
lcd_clk_sel	12	11:10	3
mfake	03	3:2	2
mi_enable[1)]	13	13:11	-
mi_sel_clk5k	13	16	0
mi_updaterate	13	15:14	1
mod_math	00	3	0
mod_rspan	01	6	-
mr2_en	01	18	1
mult_en_ub	01	10	1
mult_Hb1	04	23:00	1
mult_Hb2	05	23:00	1
mult_Hb3	06	23:00	1
mult_Hb4	07	23:00	1
mult_TkG			see Tk-Gain
mult_TkO			see Tk-Offset
mult_TkPar	07	23:00	-
mult_Ub	10	15:8	0xFB

Bit name	Config Register	Bit position	Recommended Value (decimal)
multio3_sel	12	13:12	-
multio4_sel	13	03:00	-
neg_sense	03	15	0
osz10khz_fsoc	00	7:4	13
otp_pwr_cfg	01	2	-
otp_pwr_prg	01	1	-
otp_usr_prg	01	0	-
ps_dis[3]	03	11	0
ps_noise_en[3]	01	15	0
ps_shift_clk_noise[3]	01	16	0
ps_tdc1_adjust[3]	03	9:4	23
scon<1>[4]	80	9	-
scon<2>[4]	80	10	-
scon<0>[4]	80	8	-
sel_comp_r	01	14:13	2
sel_comp_int	12	20	-
sel_compth2	01	12	0
sel_rc_osc2	13	17	-
sel_refresh_vlt	12	9:8	2
sel_rc_osc1	13	19	-
sel_rtemp_300R	14	00	-
sel_start_osz	03	19:17	3
sel_startdel	03	23:22	2
selqha	13	9:4	45
sense_ discharge	13	10	1
single_conv_ extern	01	05:04	-
single_ conversion	02	2	-
status_uart_rx_data_par[6]	80	11	status only (not to write)
status_uart_start	80	22	status only (not to write)
status_uart_stop	80	23	status only (not to write)
store_tdc_ times	13	23	
stretch	03	13:12	-
tdc_conv_cnt	00	23:16	-
tdc_sleepmode	01	17	-
Tk-Offset	08	23:0	0x100000
Tk-Gain	09	23:0	0

Bit name	Config Register	Bit position	Recommended Value (decimal)
uart_4Mhz_divider	91	18	1
uart_auto_det_stop	91	15	-
uart_baud_rate	91	7:4	5
uart_clk_en	91	19	0
uart_en	03	20	-
uart_mode	91	13	-
uart_mpcomm[4)]	91	16	-
uart_par[6)]	91	11	-
uart_rdx_sel	91	9:8	0
uart_rec_buf<7:0>	80	7:0	-
uart_rec_en	91	12	-
uart_rec_int_ack	91	17	-
uart_rec_int_dis	91	14	0
uart_sbuf_i0	86	7:0	-
uart_sbuf_i1	86	15:8	-
uart_sbuf_i10	89	15:8	-
uart_sbuf_i11	89	23:16	-
uart_sbuf_i12	90	7:0	-
uart_sbuf_i13	90	15:8	-
uart_sbuf_i14	90	23:16	-
uart_sbuf_i15	84	23:16	-
uart_sbuf_i2	86	23:16	-
uart_sbuf_i3	87	7:0	-
uart_sbuf_i4	87	15:8	-
uart_sbuf_i5	87	23:16	-
uart_sbuf_i6	88	7:0	-
uart_sbuf_i7	88	15:8	-
uart_sbuf_i8	88	23:16	-
uart_sbuf_i9	89	7:0	-
uart_trans	91	10	-
uart_tx_cnt	91	03:00,20	-
upd_vlt	14	13:12	0
usr_epr_always_on	12	5	0
usr_epr_prg_time	12	3:2	0

1) mi = multi-input
2) cport = capacitive ports
3) ps = phase shifter
4) mpcomm = multi-processor communication
5) scon = serial port control register
6) par = parity

5.3 List of configuration registers

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 OTP / 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_fsoc							
																0	1	1	0	1	0	1	1

mod_math
cpu_speed
connect_vcc_rosc

Parameter	Recommended Value	Description	Settings
tdc_conv _cnt	-	Single Conversion Timer based on 10 kHz/64 = 156.25 Hz	
io_a	-	Setting I/O 0 to I/O 7 to zero or one when the pin is configured as output. io_a [8] = I/O 0 io_a [9] = I/O 1 ... io_a [15] = I/O 7	
mod_math	0	Set mathematics to single variable strain gage.	1= on 0 = off
connect_vcc_rosc	0	0 : The in built 4 MHz oscillator in PS09 is supplied with voltage on Pin 40 (VCC_RC). 1: The in built 4 MHz ring oscillator in PS09 is supplied internally by the chip's power supply itself.	0,1

Configreg_01 RAM address 49 OTP / EEPROM bytes 3 - 5

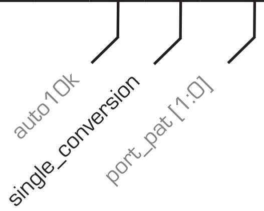
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	adj_hr																						
1	0	1	0	1	1		0	0			0	0	1	0						0			

en_emi_meas_
gain_comp
mr2_en
tdc_sleepmode
ps_shift_clk_noise
ps_noise_en
sel_comp_r [1:0]
sel_compth2
en_tkpar
mult_en_ub
en_avcal
integrated_span
en_gain
mod_rspan
single_conv_extern
dis_wheat_pp
otp_pwr_cfg
otp_pwr_prg
otp_usr_prg

Parameter	Recommended Value	Description	Settings
mr2_en	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
sel_comp_r [1:0]	00	Selects the value of the comparator resistor	00 = 6k 01 = 6k 10 = 2.5k 11 = 1.8k
sel_compth2	0	Selects a 2nd threshold for the comparator, mainly to minimise gain drift over supply voltage at low supply voltages (<2.7 V). Suited for low supply voltage applications.	0 = disabled 1 = enabled
en_tkpar	0	Enables software correction of Rspan characteristics by simulating a parallel resistor	1 = on 0 = off
mult_en_ub		Enable multiplications for supply voltage correction	1 = Enabled
integrated_rspan		Use internal Rspan for temperature compensation	1 = active
en_gain		Enable multiplications in gain correction	1 = Enabled
mod_rspan		Enable internal multiplication of gain compensation resistor Rspan	1 = Enabled
single_conv_extern [1:0]	0	A single “single conversion” can be started by an external pin	0 = Off 1 = GPIO3 2 = GPIO4 3 = GPIO5
otp_pwr_cfg		Configuration in the OTP/EEPROM is used after a power-on reset	as frontend := 0 stand-alone := 1
otp_pwr_prg		Start user code at OTP/EEPROM address 48 after a power-on reset	as frontend := 0 stand-alone := 1
otp_usr_prg		Start user code at OTP/EEPROM address 48 after a measurement	as frontend := 0 stand-alone := 1

Configreg_02 RAM address 50 OTP / EEPROM bytes 6 - 8

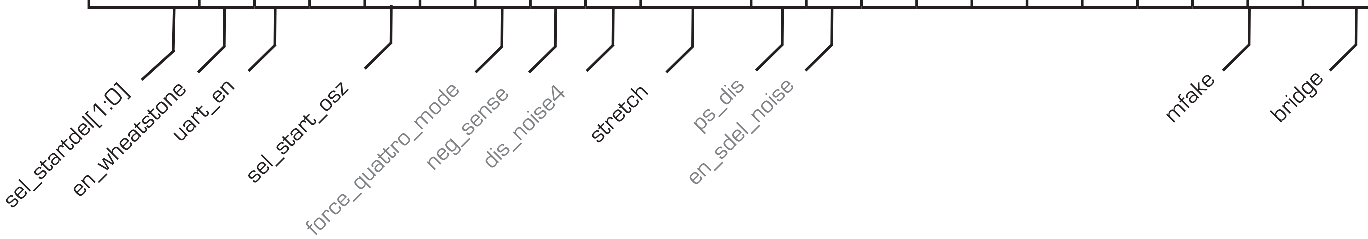
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
avrate										cytime													
																				1		1	0



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)	
single_conversion	-	Select operation mode	0 = Continous mode 1 = Single conversion mode

Configreg_03 RAM address 51 OTP / EEPROM bytes 9 - 11

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														ps_tdc1_adjust									
1	0			0	0	1		0	0			0	0	0	1	0	1	1	1	1	0		



Parameter	Recommended Value	Description	Settings
sel_startdel [1:0]	2	Adds a delay between discharging Cload and TDC Start	0 = 5ns 1 = 40ns 2 = 150ns 3 = 300ns
en_wheatstone [2:0]	-	Enable Wheatestone mode	1 = enabled
uart_en	-	Enables UART	0 = disabled 1 = enabled
sel_start_osz [0:2]	1	Sets delay from start of 4 MHz oscillator to start of measurement	0 = off 1 = continously on 2 = 100 µs 3 = 200 µs 4 = 300 µs 5 = 400 µs 6 & 7 are not supported
stretch	-	Select stretch mode	0 = off 1 = not recommended 2 = 2xR (half bridge), 200 µs delay 3 = 2xR (half bridge) 300 µs delay
mfake	2	Sets the number of fake measurements	0 = 0 Fake measurements 1 = 2 Fake measurements 2 = 4 Fake measurements 3 = 8 Fake measurements

bridge	-	Sets the number of half bridges that are measured	0 = one half bridge 1 = 2 half bridges 2 = not supported 3 = 4 half bridges

Configreg_04 RAM address 52 OTP / EEPROM bytes 12 - 14

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Mult_Hb1</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Mult_Hb1	-	Multiplication factor for HB1 result signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h780000 = 7.5

Configreg_05 RAM address 53 OTP / EEPROM bytes 15 - 17

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Mult_Hb2</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Mult_Hb2	-	Multiplication factor for HB2 result signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h780000 = 7.5

Configreg_06 RAM address 54 OTP / EEPROM bytes 18 - 20

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Mult_Hb3</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Mult_Hb3	-	Multiplication factor for HB3 result signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h780000 = 7.5

Configreg_07 RAM address 55 OTP / EEPROM bytes 21 - 23

The assignment of configreg 07 changes in accordance to en_tkpar (configreg 01, bit 11)

For en_tkpar = 0:

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Mult_Hb4</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Mult_Hb4	-	Multiplication factor for HB4 result signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h780000 = 7.5

For en_tkpar = 1 (only for sensors with Rspan compensation resistor)

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Mult_tkpar</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Mult_tkpar	-	Multiplication factor for virtual Rspan parallel resistor signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h7147AE1 = 1.28

Configreg_08 RAM address 56 OTP / EEPROM bytes 24 - 26

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Tk-Gain (formerly Mult_TKG)</td></tr>
<tr><td></td><td colspan="3">integer</td><td colspan="20">fractional</td></tr>
</table>

sign

Parameter	Recommended Value	Description	Settings
Tk-Gain	-	Multiplication factor for Rspan correction signed fixed-point number	range: [-2^23 to 2^23-1]/ 2^20 example: 'h0C0000 = 0.75

Configreg_09 RAM address 57 OTP / EEPROM bytes 27 - 29

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">Tk-Offset (formerly Mult_TKO)</td></tr>
</table>

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

Configreg_10 RAM address 58 OTP / EEPROM bytes 30 - 32

<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">calcor</td><td colspan="8">Mult_Ub</td><td colspan="8">gain_comp</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td colspan="8"></td><td colspan="8"></td></tr>
</table>

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)
gain_comp	-	Multiplication factor for gain correction. g := g * [0 to 255]/2^7

Configreg_11 RAM address 59 OTP / EEPROM bytes 33 - 35

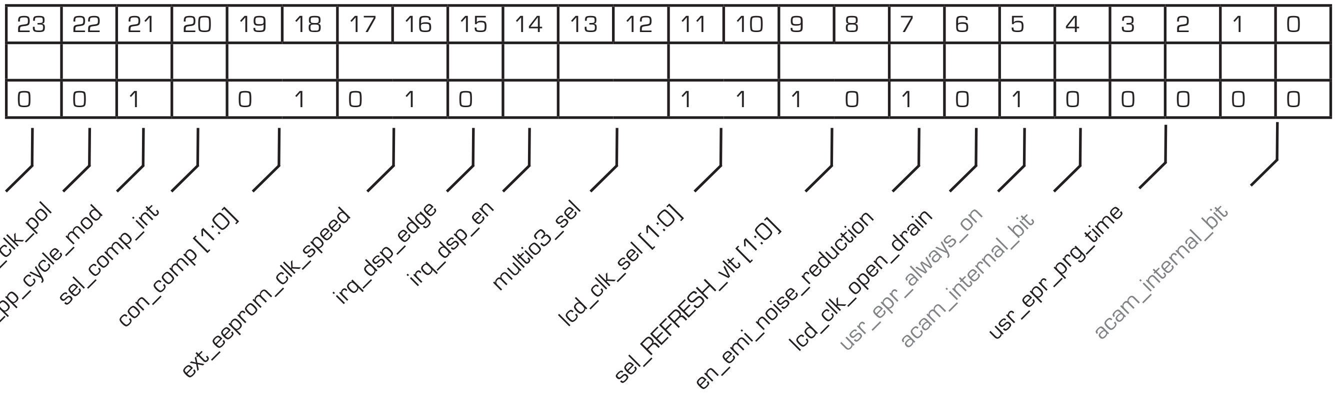
<table>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">io_en_digital</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="8"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td></tr>
</table>

io_en_7_mio io_en_6_mio io_en_5_mio io_en_4_mio io_en_3_mio io_en_2_sck io_en_1_sdi io_en_0_sdo

Parameter	Recommended Value	Description	Settings
io_en_digital	-	Enable for the input buffers for I/O 7..0 respectively	1: Input enabled 0 : Off and hence current free (recommended when using capacitive switches)
io_en_7_mio	-	Port definition	11 = input 10 = in-pull-down 01 = in-pullup 00=out
io_en_6_mio	-	Port definition	11 = input 10 = in-pull-down 01 = in-pullup 00=out

io_en_5_mio	-	Port definition	11 = input 10 = in-pull-down 01 = in-pullup 00=out
io_en_4_mio	-	Port definition	11 = input 10 = in-pull-down 01 = in-pullup 00=out
io_en_3_mio	-	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_1_sdi	-	Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input
io_en_0_sdo	-	Port definition	00 = output 01 = input with pull-up 10 = input with pull down 11 = input

Configreg_12: RAM address 60 OTP / EEPROM bytes 36 - 38



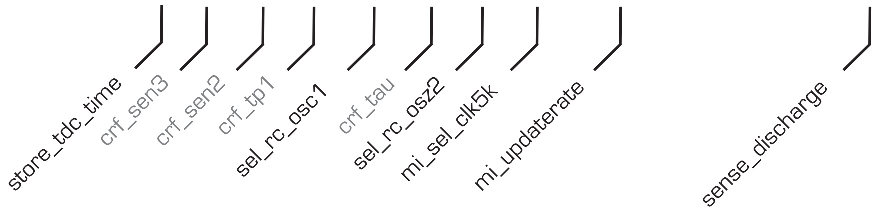
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1		0	1	0	1	0				1	1	1	0	1	0	1	0	0	0	0	0

Parameter	Recommended Value	Description	Settings
irq_dsp_sel	0	Pin assignment for the DSP interrupt	0: GPIO3 1: GPIO4
lcd_clk_pol	-	Lcd clock polarity	0: low pulse (active low) 1: high pulse (active high)
dis_pp_cycle_mod	1	Enables / disables gain measurement cycle modification	0 = enable 1 = disable
sel_comp_int		Selects between internal or external comparator	0: external comparator 1: internal comparator
con_comp [1:0]	0	Controls the comparator switch of mode	00 : off 01 : on during measurement 10 : on during load 11 : always on

ext_eeprom_clk_speed	1	Controls the speed of the eeprom clock. Recommended value: 01	00: fastest 11: slowest
irq_dsp_edge	0	Sets edge sensivity of the DSP interrupt	1: negative edge 0: positive edge
irq_dsp_en	-	Exclusive DSP Interrupt enable, The measuremen resumes after the interrupt is processed	1: Interrupt enabled 0: Interrupt disabled
multio3_sel (Mult_IO3 output)	0	Sets configuration of with GPIO3 Pin, when the GPIO3 is configured as an output pin	0= multio3 1= Interrupt due to measurement end 2= uart_txd 3= lcd_clk
multio3_sel (Mult_IO3 input)	0	Sets configuration of with GPIO3 Pin, when the GPIO3 is configured as an input pin	0= multio3 1= Input for external interrupt
lcd_clk_sel [1:0]		Clock for external LCD Driver Double the 10kHz Clock GPIO3 or GPIO4 must be configured respectively	Programmabke pulse widths: 0: Off 1: 100ns Pulse 2: 200ns Pulse 3: 800ns Pulse
sel_refrewsh_vlt [1:0]		Refresh rate of the 1.8V Voltage	0= 20 Hz 1= 10 Hz 2= 5 Hz 3= after every end_avg (recommended for RC oscillator mode)
en_emi_noise_reduction	1	Enables modified control of strain gage measurement to suppress EMI influence. Only supported in PS09 mode. For PS081 compatible mode, this bit must be 0	0 = disabled (PS081 compatible mode) 1 = enabled (PS09 mode)
lcd_clk_open_drain	0	Generates open drain LCD clock on I/O 3 or I/O 4	0 = Standard LCD clock output 1 = Open drain LCD clock output
usr_epr_prg_time	0	Configures programming time of the user EEPROM	0 = 6.4ms (recommended) 1 = 12.6ms 2 = 6.4ms, only erase 3 = 6.4ms, only write

Configreg_13: RAM address 61 OTP / EEPROM bytes 39 - 41

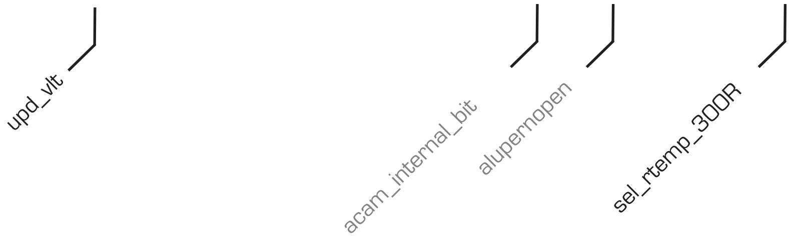
23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										mi_enable				selqha						multio4_sel			
0	0	0	0		0		0	0	1				1	1	0	1	1	0	1				



Parameter	Recommended Value	Description	Settings
store_tdc_time	0	The TDC times are summed up and stored in Ram cells 192 – 223 and are accessible to the user program.	0 = disabled 1 = enabled
sel_rc_osc1	-	Enables integrated RC oscillator (high resoluton)	1 = enable 0 = disable
sel_rc_osz2	-	Enables the on chip RC oscillator (low resolution)	1 = enable 0 = disable
mi_sel_clk5k	0	Select basic clock frequency of the multi input ports	0 = 10 kHz 1 = 5 kHz
mi_updaterate	1	Select final update rate for multi input ports based on basic clock frequency	Basic frequency = 10kHz 0 12.5 Hz 1 25 Hz 2 50 Hz 3 100 Hz Basic frequency = 5kHz 0 6.25 Hz 1 12.5 Hz 2 25 Hz 3 50 Hz
mi_enable[2:0]	-	Enables / Disable each of the the multi input ports individually. mi_enable[0] -> MULT_IO3_C1, mi_enable[1] -> MULT_IO4_C2, mi_enable[2] -> MULT_IO6_C4,	1 = Enable 0 = Disable
sense_discharge	1	Sets fast discharge of comparator‘s low pass capacitor at püin rcomp	1 = enabled
multio4_sel (Mult_IO4 ouput)	0	Configures output options of GPIO4 pin	0 = general purpose output 1 = Interrupt, indicates measurement end 2 = uart_txd 3 = lcdclk 4 = clk10div_3 5 = clk10div_4 6 = epr_acc 7 = clk675khz 8 = phase shifter 9 = start_stop 10 = sense_ac1_comp2 11 = load 12 = clk10khz 13 = clkalu 14 = epr_acc 15 = otp_racc
multio4_sel (Mult_IO4 input)	0	Configure input options of GPIO4 pin	0 = general purpose input 1 = input for external interrupt

Configreg_14: RAM address 62 OTP / 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
	bandgap_trim											acam_internal									caltime		
0	1	0	1	0		0	0	1		0	0	0	0	0	1	0	0	1	1	0	0	1	



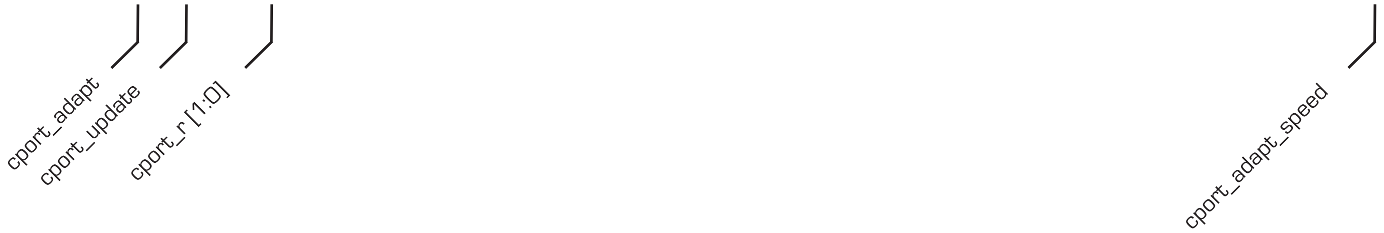
Parameter	Recommended Value	Description	Settings
upd_vlt[1:0]	0	Sets frequency rate of voltage measurement	0 = off 1 = every 0.6 sec 2 = after every refresh 3 = after every 8th refresh
sel_rtemp_300R	-	Select internal reference resistor for integrated temperature measurement	0 : 600 Ohm (recommended for 1 k strain gauge) 1 : 300 Ohm (recomended for 350 Ohm strain gages)

Configreg_15: RAM address 63 OTP / EEPROM bytes 45 - 47

The assignment of configreg 15 changes in accordance to cport_adapt setting

For cport_adapt = 1

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



Parameter	Recommended Value	Description	Settings
cport_adapt	1	Automatic threshold adaption of the capacitive ports	0 = disabled 1 = enabled
cport_update		Sampling frequency for capacitive ports	0 = 39 Hz 1 = 78 Hz
cport_r [1:0]	-	Selects the integrated discharging resistor for capacitive sensing	0 = 25k 1 = 50k 2 = 100k 3 = 200k
cport_en	-	Bitwise enable for each of the 4 capacitive sensor keys	0 = off 1 = on
cport_thresh	100	Initial setting to define the minimal threshold of the capactive switches.	
cport_adapt_speed	1	Speed control for automatic threshold adaption	0 = not supported 1 = fast 2 = medium 3 = slow

For cport_adapt = 0

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				cport_en				cport_thresh2								cport_thresh1							
0	0	0	0	0	0	0	0																

cport_adapt
cport_update
cport_r [1:0]

Parameter	Recommended Value	Description	Settings
cport_adapt	0	Manual threshold adaption of the capacitive ports via cthresh1 & 2	0 = disabled 1 = enabled
cport_update		Sampling frequency for capacitive ports	0 = 39 Hz 1 = 78 Hz
cport_r [1:0]	-	Selects the integrated discharging resistor for capacitive sensing	0 = 25k 1 = 50k 2 = 100k 3 = 200k
cport_en	-	Bitwise enable for each of the 4 capacitive sensor keys	0 = off 1 = on
cport_thresh2	-	Defines threshold for capacitive ports 3 & 4	
cport_thresh1	-	Defines threshold for capacitive ports 1 & 2	

UART - Configreg_80: RAM address 80 Status Register

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

status_uart_stop

status_uart_start

status_uart_rx_data_par

r = read only, status

Parameter	Recommended Value	Description	Settings
status_uart_stop	read only	Indicates reception of he stop bit at the end of a data byte received by the UART	
status_uart_start	read only	Status of the start bit that actually begins the reception of a data byte. The status of this bit helps to identify if the reception started was because of a glitch on the RXD pin or a valid start bit.	
status_uart_rx_data_par	read only	Parity of the 8 data bits received by the UART. Counts number of logic 1s in the received data byte	0: indicates an even parity 1: indicates an odd parity
scon	read only	Indicates the status of UART data transmission	0: indicates Receive interrupt 1: indicates Transmit interrupt 2: indicates Status of the 9th received data bit
uart_rec_buf	read only	Data Byte received by the UART	

UART - Configreg_84: RAM address 84

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i15								n/c															

Parameter	Recommended Value	Description	Settings
uart_sbuf_i15	-	UART Send FIFO: 15th transmit byte	

UART - Configreg_86: RAM address 86

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i2								uart_sbuf_i1								uart_sbuf_i0							

Parameter	Recommended Value	Description	Settings
uart_sbuf_i2	-	UART Send FIFO 3rd transmit byte	
uart_sbuf_i1	-	UART Send FIFO 2nd transmit byte	
uart_sbuf_i0	-	UART Send FIFO 1th transmit byte	

UART - Configreg_87 : RAM address 87

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i5								uart_sbuf_i4								uart_sbuf_i3							

Parameter	Recommended Value	Description	Settings
uart_sbuf_i5	-	UART Send FIFO 6th transmit byte	
uart_sbuf_i4	-	UART Send FIFO 5th transmit byte	
uart_sbuf_i3	-	UART Send FIFO 4th transmit byte	

UART - Configreg_88: RAM address 88

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i8								uart_sbuf_i7								uart_sbuf_i6							

Parameter	Recommended Value	Description	Settings
uart_sbuf_i8	-	UART Send FIFO 9th transmit byte	
uart_sbuf_i7	-	UART Send FIFO 8th transmit byte	
uart_sbuf_i6	-	UART Send FIFO 7th transmit byte	

UART - Configreg_89: RAM address 89

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i11								uart_sbuf_i10								uart_sbuf_i9							

Parameter	Recommended Value	Description	Settings
uart_sbuf_i11	-	UART Send FIFO 12th transmit byte	
uart_sbuf_i10	-	UART Send FIFO 11th transmit byte	
uart_sbuf_i9	-	UART Send FIFO 10th transmit byte	

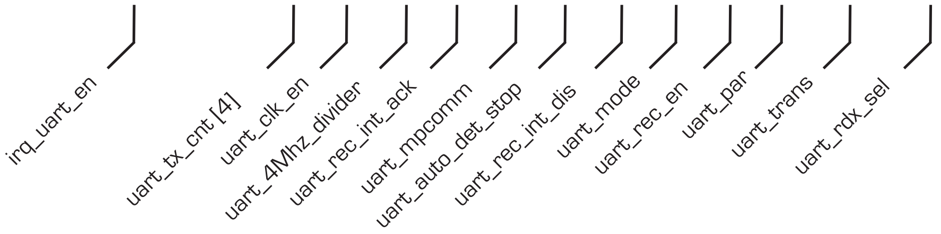
UART - Configreg_90: RAM address 90

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_sbuf_i14								uart_sbuf_i13								uart_sbuf_i12							

Parameter	Recommended Value	Description	Settings
uart_sbuf_i14	-	UART Send FIFO 15th transmit byte	
uart_sbuf_i13	-	UART Send FIFO 14th transmit byte	
uart_sbuf_i12	-	UART Send FIFO 13th transmit byte	

UART - Configreg_91: RAM address 91

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



Parameter	Recommended Value	Description	Settings
irq_uart_en	-	Enables the UART receive interrupt to the DSP. It does not affect the IRQ_DSP_EN in any way.	
uart_tx_cnt [4]	-	Highest bit to count the number of bytes received by the UART. See also bit 0 to 3	
uart_clk_en	-	Switches on the oscillator clock (4MHz or 3.6864 MHz) for UART	
uart_4Mhz_divider	-	UART clock divider	1 : Uses the 4MHz oscillator clock for the baud rate generation. 0: If a 3.6864 MHz oscillator is used. Enable baud rate generation based on this clock frequency.
uart_rec_int_ack	-	This bit must be written to 1 in order to clear the RI, i.e. scon_o<0> [Bit 8 of Reg.80]. This is an acknowledgment to the receive interrupt, so that the receive interrupt will be set again when a further new byte will be received.	
uart_mpcomm	-	Multi-processor communication UART Mode 1	1: Receive Interrupt (RI) is not set when the 9th data bit is 0 0: Receive Interrupt (RI) is always set irrespective of the status of the 9th data bit
		Multi-processor communication UART Mode 0	1: Receive Interrupt (RI) is not set when the Stop bit is 0 0: Receive Interrupt (RI) is always set irrespective of the status of the stop bit
uart_auto_det_stop	-	Configures Receive interrupt conditions	1: Receive interrupt (RI) is set only if a valid stop bit is received at the end of the data byte. 0: Receive interrupt (RI) is set independent whether a stop bit is received or not.
uart_rec_int_dis	0	DSP interrupt generation	0: DSP interrupt with every received byte 1: Disable of interrupt generation to DSP
uart_mode	-	Defines the UART operating modes	0: 8 Bit without parity 1: 8 Bit with automatically generated parity
uart_rec_en	-	UART receive enable	

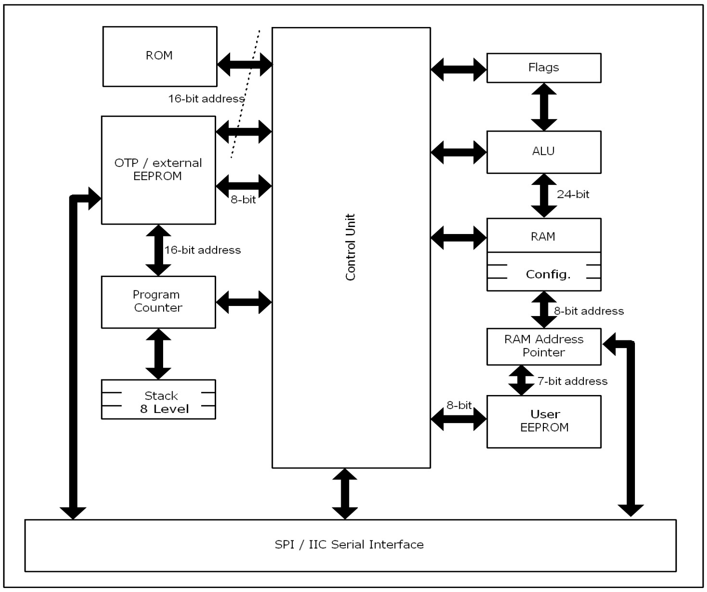
<table>
<tr><td>uart_par</td><td>-</td><td>Parity to be used by the UART when sending the data byte.</td><td>0: indicates an even parity, i.e. Even number of 1s in the transmitted data byte
1: indicates an odd parity, i.e. Odd number of 1s in the transmitted data byte</td></tr>
<tr><td>uart_trans</td><td>-</td><td>Start UART data transmission</td><td></td></tr>
<tr><td>uart_rdx_sel</td><td>-</td><td>Determines the GPIO to be used as receive input RXD</td><td>3: GPIO5
2: GPIO4
1: GPIO3
0: GPIO2</td></tr>
<tr><td>uart_baud_rate</td><td>-</td><td>Sets UART Baudrate</td><td>
<table>
<tr><th>Baudrate</th><th>Configuration with 3.6864MHz or 4 MHz clock</th></tr>
<tr><td>300</td><td>0</td></tr>
<tr><td>600</td><td>1</td></tr>
<tr><td>1200</td><td>2</td></tr>
<tr><td>2400</td><td>3</td></tr>
<tr><td>4800</td><td>4</td></tr>
<tr><td>9600</td><td>5</td></tr>
<tr><td>19200</td><td>6</td></tr>
<tr><td>38400</td><td>7</td></tr>
<tr><td>57600</td><td>8</td></tr>
<tr><td>76800</td><td>9</td></tr>
<tr><td>115200</td><td>10</td></tr>
</table>
</td></tr>
<tr><td>uart_tx_cnt[3:0]</td><td>-</td><td>Defines the number of bytes to be transmitted by the UART.</td><td></td></tr>
</table>

Table of Contents Page

6 Central Processing Unit (CPU)

6.1 Block Diagram

Figure 6.1 Block Diagram



6.2 Memory Organization

Figure 6.2 PS09 Memory Organization

FFFF h ...... F000 h	65535 61440	ROM Program memory	4k
EFFF h .... 2000 h	61439 8192	Reserved	
1FFF h .... 0000 h	8191 0000	User program Memory 8192 bytes of OTP / External EEPROM	8k

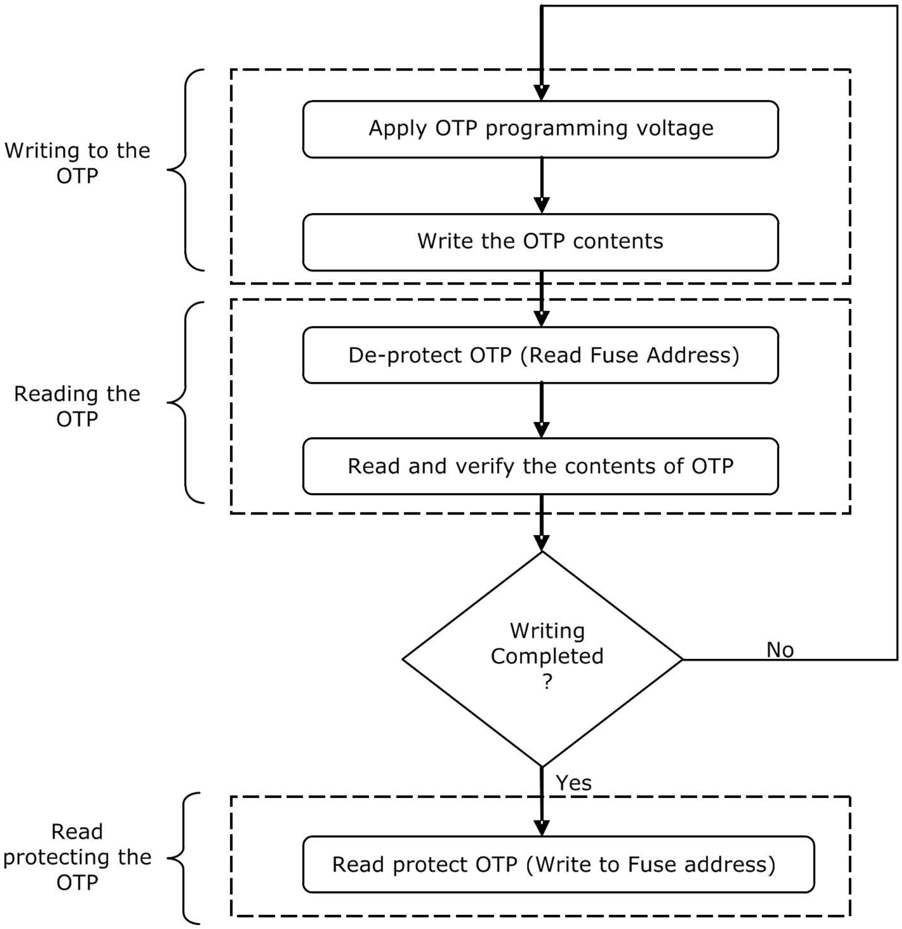
6.2.1 OTP

The user program memory in PS09 available for user programming is 8kbyte in size. This 8kB user program memory is implemented by an on chip One Time Programmable ROM, the OTP. As the name suggests, this memory is writable only once. Hence for development of the user program, the PS09 supports an erasable and re-programmable external EEPROM, maximum 8 kB in size. Once the application program development is complete with the external EEPROM, then the same program can be downloaded into the OTP and it must function in the same manner with the OTP.

The first 48 bytes of the OTP from location 0 – 47 are reserved for the configuration data. In order to enable programming of the OTP, an external programming voltage of 6.5V must be available on pin VPP_OTP of the PS09.

The following flow diagram shows how the OTP is generally handled, details follow in subsequent sections.

Figure 6.3 Using the OTP

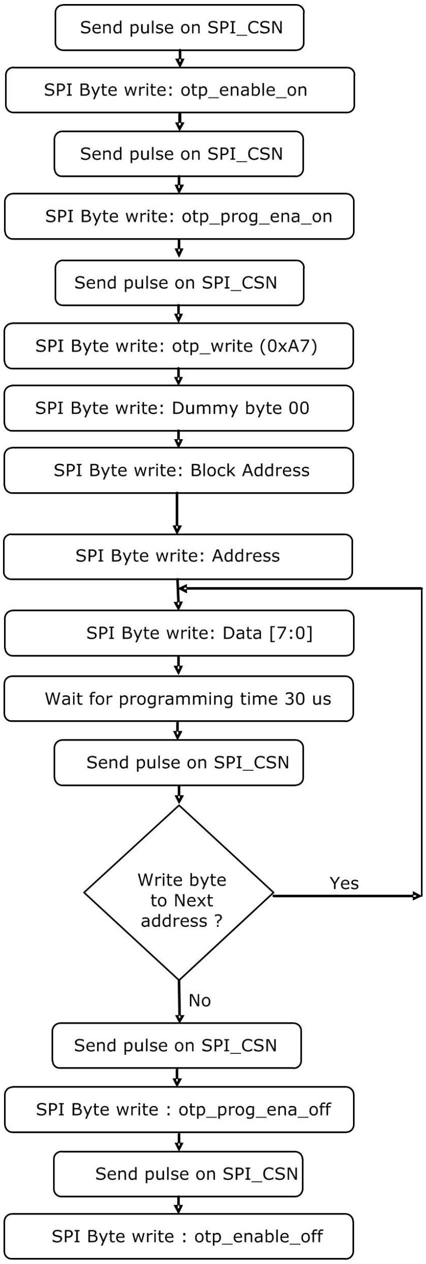

Writing to the OTP
Apply OTP programming voltage
Write the OTP contents
Reading the OTP
De-protect OTP (Read Fuse Address)
Read and verify the contents of OTP
Writing Completed ?
No
Yes
Read protecting the OTP
Read protect OTP (Write to Fuse address)

6.2.1.1 Writing to the OTP

The OTP needs an external voltage of 6.5 V on the VPP_OTP pin of the PS09 in order to enable programming. In addition to enabling the OTP, there are op codes to enable and disable the PROG (Enable Programming) signal of the OTP.

The following is a flowchart that shows the SPI command sequence to write a byte to the OTP.

Figure 6.4 Writing to the OTP


Send pulse on SPI_CSN
SPI Byte write: otp_enable_on
Send pulse on SPI_CSN
SPI Byte write: otp_prog_ena_on
Send pulse on SPI_CSN
SPI Byte write: otp_write (0xA7)
SPI Byte write: Dummy byte 00
SPI Byte write: Block Address
SPI Byte write: Address
SPI Byte write: Data [7:0]
Wait for programming time 30 us
Send pulse on SPI_CSN
Write byte to Next address ?
Yes
No
Send pulse on SPI_CSN
SPI Byte write : otp_prog_ena_off
Send pulse on SPI_CSN
SPI Byte write : otp_enable_off

For a list of all op codes pertaining to accessing the OTP through the SPI / IIC interface, refer to section 4.5.3.3

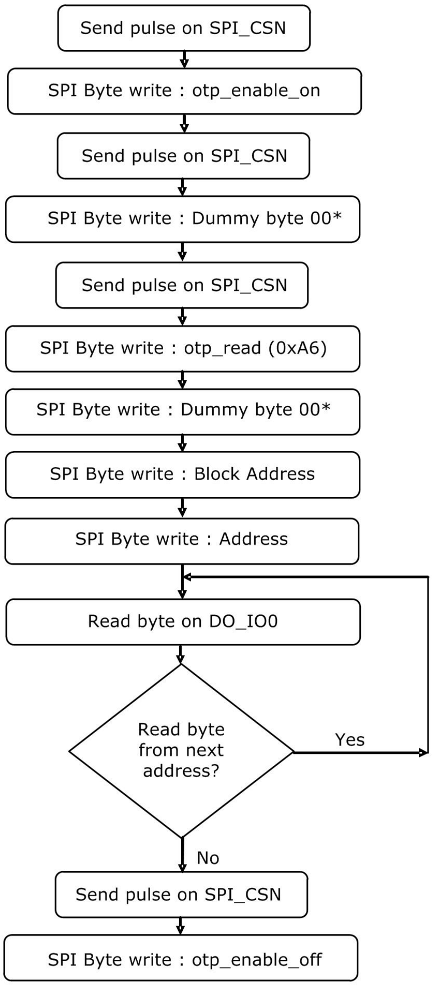
6.2.1.2 Reading the OTP

On power on reset, the OTP is by default read protected. An un-programmed OTP content is all 0s. To enable the OTP, the Address 8143, called the Fuse Address must be read first. When the content of the Fuse address is all 0s indicating an un-programmed OTP, then the OTP is enabled for reading, i.e. the OTP is de-protected.

Hence this de-protection is the first step in working with the OTP.

The following is a flowchart that shows the general sequence of sending SPI commands to read a byte from the OTP. This is the sequence to be used when controlling the PS09 by an external microcontroller, through the SPI / IIC interface.

Figure 6.5 Reading the OTP


Send pulse on SPI_CSN
SPI Byte write : otp_enable_on
Send pulse on SPI_CSN
SPI Byte write : Dummy byte 00*
Send pulse on SPI_CSN
SPI Byte write : otp_read (0xA6)
SPI Byte write : Dummy byte 00*
SPI Byte write : Block Address
SPI Byte write : Address
Read byte on DO_IO0
Read byte from next address?
Yes
No
Send pulse on SPI_CSN
SPI Byte write : otp_enable_off

* the Dummy byte (0x00) is required to be sent as it is needed because of timing purposes

For a list of values of all op codes for accessing the OTP through the SPI / IIC interface, refer to section 4.5.3.3

6.2.1.3 Read protecting the OTP

Once the OTP has been programmed with the user program and when the code development is complete, the code could be read protected with the Fuse address. For read protecting the OTP, the fuse address 8143 must be written with a non-zero value. The read protection process is completed by reading the address 8143 after writing it with the non-zero value.

6.2.2 External EEPROM

An external EEPROM of up to 8kB size is supported as user program memory by the PS09 with the sole purpose of supporting user program development. The final program will be written and realized in the on-chip OTP. It is to be noted that the program will be executed in exactly the same manner, irrespective of whether the user program memory is the OTP or the external EEPROM.

The programming sequence to write a byte into the external EEPROM and to read a byte from the external EEPROM through the SPI / IIC interface can be found under Section 4.5.3.4.

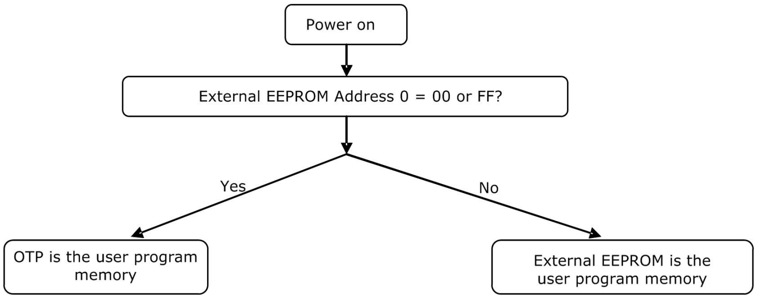
6.2.3 User Program development using the external EEPROM

This section describes how the program can be developed by the user using the external EEPROM as the program memory.

As already stated, basically a user program is executed in the same manner, irrespective of whether the user program memory used is the OTP or the external EEPROM. However the PS09 has to know, which of the two has to be used as the user program memory. For this purpose, as a standard operation on power-up, the PS09 checks for the presence of an external EEPROM by reading address 0 of the external EEPROM. When 00 or FF is read back from address 0 of the EEPROM, then the PS09 takes the internal OTP as the user program memory and executes the code from the OTP. When a value other than 00 and FF is read from the Address 0 of the external EEPROM, then the EEPROM is considered to be the user program memory by the chip and user code in the external EEPROM is executed.

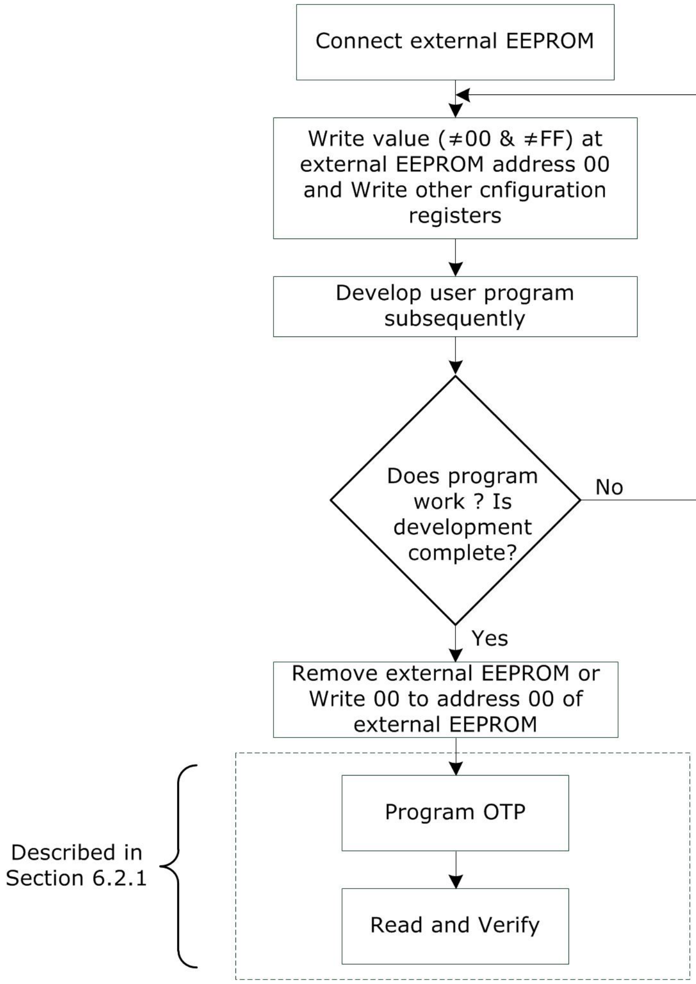
The content of address 00 corresponds to value of the bits 23:16 of Configuration register 0 (tdc_conv_cnt).

Figure 6.6 Program Memory on POR


Power on
External EEPROM Address 0 = 00 or FF?
Yes
No
OTP is the user program memory
External EEPROM is the user program memory

Once the user program development is completed using the external EEPROM then, the final program is ready to be written to the OTP. Then the external EEPROM is either removed physically or it is made inactive to the PS09 by writing the address 00 of the external EEPROM with 00 or FF. The following flowchart gives an overview of how the user program is developed using the EEPROM and transferred to the OTP finally.

Figure 6.7 User Program Development


Connect external EEPROM
Write value (≠00 & ≠FF) at external EEPROM address 00 and Write other cnfiguration registers
Develop user program subsequently
Does program work ? Is development complete?
No
Yes
Remove external EEPROM or Write 00 to address 00 of external EEPROM
Program OTP
Read and Verify
Described in Section 6.2.1

The PS09 Assembler Software which is used for user program development supports downloading the developed program to the external EEPROM or to the on chip OTP. The target for downloading the program can be selected from a drop down list on the Download page of the assembler.

The lower 48 bytes in the user program memory are reserved for an automatic configuration of the PSØ9 during a power-on reset. 3 successive bytes are added to a 24 bit word. So there are 16 words of 24 bit each that are used for configuration register 0 to 15. During a power- on reset they are copied into RAM address 48 to 63.

Generally the code execution from the external EEPROM takes longer than from the internal OTP. This fact needs to be considered when delay routines are realized using incr/decr opcodes in loops as the delay will be longer when executed from the EEPROM in comparison with the OTP. The code execution from the external EEPROM is approx. 10 to 15 times slower than from the internal OTP.

6.2.4 ROM Program memory

In PS09, 4kbytes is reserved for the ROM starting at address F000 h. All computation routines needed for the PICOSTRAIN measuring method reside here. The program can jump back from the ROM to the OTP/external EE-PROM by setting appropriate configuration bits.

The program memory in PS09 available for user programming is 8kbyte in size. This 8kB program memory is implemented by an on chip One Time Programmable ROM, the OTP. An 8kB external EEPROM can be addressed optionally instead of the OTP during program development stages. On power up, as a standard operation the PS09 checks for the presence of an EEPROM by reading address 0. If the external EEPROM has to be recognized and used as program memory instead of the OTP, the address 0 of the EEPROM which corresponds to bits 23..16 of Configuration register 0(tdc_conv_cnt) have to be programmed to any value other than 00 and FF. The lower 48 bytes in the OTP are reserved for an automatic configuration of the PSØ9 during a power-on reset. 3 successive bytes are added to a 24 bit word. So there are 16 words of 24 bit each that are used for configuration register 0 to 15. During a power- on reset they are copied into RAM address 48 to 63.

In PS09, 4kbytes is reserved for the ROM starting at address F000 h. All computation routines needed for the PICOSTRAIN measuring method reside here. The program can jump back from the ROM to the OTP/external EE-PROM when configured.

6.2.5 User EEPROM

The user EEPROM in PS09 is 128 bytes of 8 bits each. This user EEPROM can be used to store calibration data that can be accessed from the user program. The processor can write to and read from these EEPROM, byte-wise using the putepr and getepr op-codes. This EEPROM hangs on the same address bus as the RAM. Hence the RAM address pointer is used to address both the user EEPROM and the RAM. See section 6.2.6 to get more details with code snippets on how the RAM address pointer is used to address both the user EEPROM and the RAM.

6.2.6 RAM Organization

Table 6.1 RAM adress organization

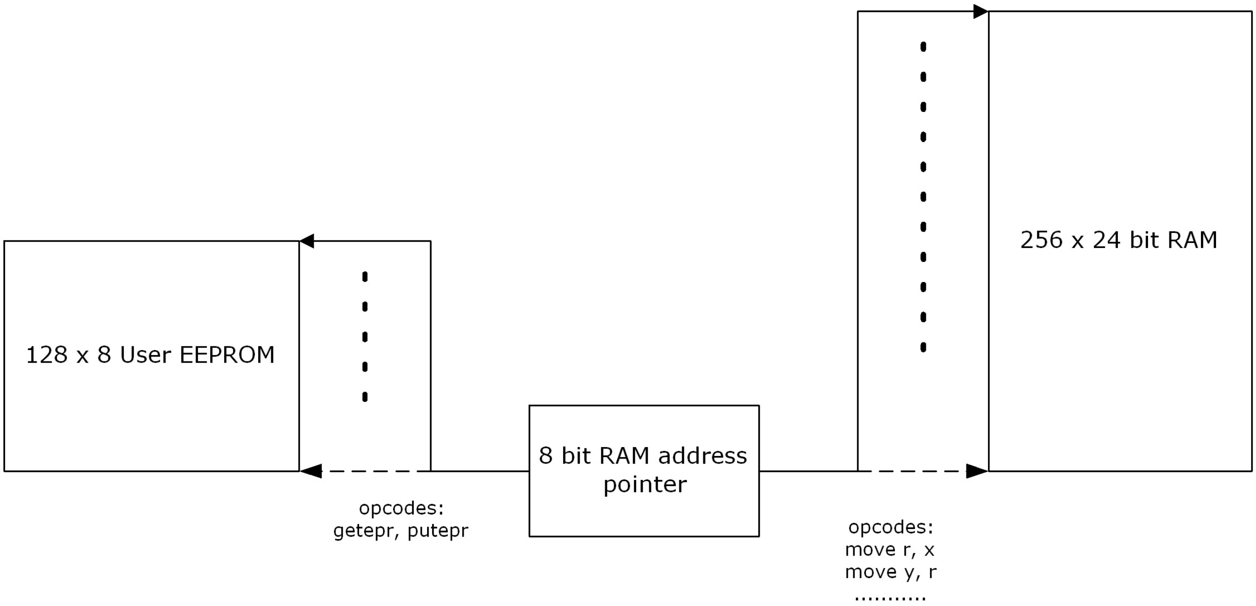
255	Modrspan result
254	Timer
253	I/O status – falling, rising and pressed status of the 8 GPIO s
252	Status of the 24 Multi Input keys, Pressed or Released
251	Status : rising edge on the 24 Multi Input keys
250	Status : falling edge on the 24 Multi Input keys
249	UBATT
248	CAL
247	HB1+
246	Status flags
245	p1/p2
244	HB0 = (A-B)+(C-D)+(...)/(A+B)+(C+D)+(...) *
243	HB4 = (G-H)/(G+H)
242	HB3 = (E-F)/(E+F)
241	HB2 = (C-D)/(C+D)
240	HB1 = (A-B)/(A+B)
239 ... 208	System RAM
207 ... 96	User RAM 207 ... User RAM 96
95 ... 92	Reserved
91 ... 86	UART Config/status reg
85 ... 81	Internal registers
80	UART Config/status reg
79... 64	Reserved for internal use
63 48	Config reg 15 ...... Config reg 0
47 32	User RAM 47 ..... User RAM 32
31... 16	User RAM in stand-alone mode; Status and Result registers in front end mode (same content as 255-240)
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

6.2.7 RAM Address Pointer

The RAM has its own address bus with 256 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 affect the content of the addressed RAM. The RAM address pointer itself is modified by separate opcodes (ramadr, incramadr,...). As explained in the previous section, the RAM address bus is additionally used to address 128 bytes of user EEPROM with particular op codes.

Figure 6.8 RAM Address Pointer



When the RAM address pointer is set to a value and op codes putepr and getepr are used, the RAM address pointer points to the respective byte in the user EEPROM. Hence operations are carried out with the respective user EEPROM byte. All other op codes like move r,x set the RAM address pointer to point to the RAM, hence the operation is done on the RAM content.

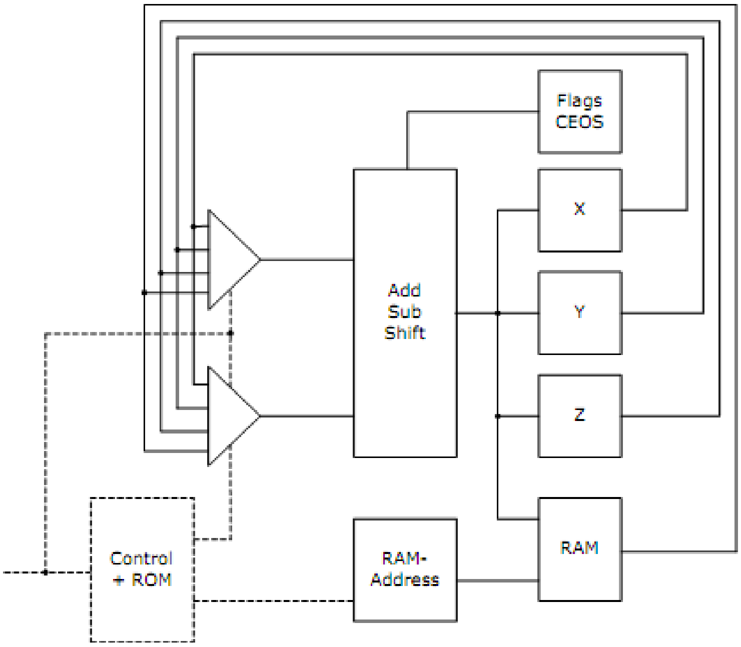
The following sample code illustrates how the RAM address pointer is used to access the user EEPROM and the RAM, based on the op code used.

Sample code:

```
Ramadr  3            // Sets the RAM address pointer to address 3
Move           r, x  // Moves the content of the X accumulator to the RAM address 3
                     // RAM Address Pointer is pointing to the RAM
Ramadr 4             // Sets the RAM address pointer to address 4
Getepr x             // Gets the content of the user EEPROM address 4 into the X
                     // accumulator
                     // RAM Address Pointer is pointing to the user EEPROM
Ramadr 3             // Sets the RAM address pointer to address 3
Putepr x             // Moves the content of the X accumulator to the user EEPROM
                     // address   3
                     // RAM Address Pointer is pointing to the user EEPROM
Clear         r      // Clears the content of RAM address 3
                     // RAM Address Pointer is pointing to the RAM
```

6.3 Arithmetic Logic Unit (ALU)

Figure 6.9 ALU block diagram



6.3.1 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.

6.3.2 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 in a jump instruction.

6.3.2.1 Carry

Shows the carry over in an addition or subtraction. With shift operations (shiftL, rotR etc.) it shows the bit that has been shifted out.

Not-equal zero

This flag is set to zero in case a new result not equal to zero is written into an accumulator (add,sub,move,swap etc.).

6.3.2.2 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.

6.3.2.3 Overflow

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

6.4 Status and Result Registers

The RAM addresses used to access the Status- and Result registers are equivalent to that of PS081, but with an offset address of 224.

NEW with PS09

In PS081 the results are located at address 16 to 31, in PS09 they are in address 240 to 255 (224+**16** to 224+**31**)

Example:

HBO value = RAM address:	**20** in PS081
HBO value = RAM adress:	244 (224 + **20**) in PS09

6.4.1 Result Registers

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

ram=240 :	HB1=(A-B) / (A+B)	HB1 un-compensated
ram=241 :	HB2=(C-D) / (C+D)	HB2 un-compensated
ram=242 :	HB3=(E-F) / (E+F)	HB3 un-compensated
ram=243 :	HB4=(G-H) / (G+H)	HB4 un-compensated
ram=244 :	HB0=(A-B)+(C-D)+(..)/ (A+B)+(C+D)+(..)	HB0 compensated sum
ram=245 :	TMP=P1/P2	Temperature measurement value
ram=246 :	Status flags	See 6.3.2
ram=247 :	HB1+	Time measurement TDC at SG_A1, Pin11
ram=248 :	CAL	Resolution TDC
ram=249 :	UBATT	Measured supply voltage
ram=250 :	Status_Multi_F	Indicates falling edge occurrence on 24 possible Multi Input keys
ram=251 :	Status_Multi_R	Indicates rising edge occurrence on 24 possible Multi Input keys
ram=252 :	Status_Multi_P	Status of the 24 Multi Input keys, Pressed or Released
ram=253 :	Status_IO	Falling, Rising and Current Status of 8 GPIO pins
ram=254 :	Timer	Status of the timer on measurement completion
ram=255 :	Modrspan	Rspan value on measurement completion. For load cells with Rspan, the ratio Rspan/Rsg when bit mod_rspan=1 in Config_reg1.

Descriptions:

A :	Discharge time measurement at
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 through the combination of Integrated Rspan and strain gage resistor at SG_D1 and SG_C2

P2: Discharge time measurement at SG_D1 | SG_C2

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 * 2^20) - 1

Status: See above

HB1+: Result (HB1+)/ns = 250 * HB1+ /214 [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 PS09. This explanation is based on Mult_HBx = 1.

Examples:

1.5 mV/V load cell, **PICO**STRAIN wiring, Mult_HBx = 1:
1.5 mV/V = 1500 ppm → result in PS09 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 PS09 at maximum strain: 133,333 (0x0208D5)

1 mV/V load cell, **PICO**STRAIN wiring, Mult_HBx = 4:
1 mV/V = 1000 ppm → result in PS09 at max. strain: 400,000 (0x061A80)

6.4.2 Status Register

Table 6.3 Status Register (RAM Address 246)

Bit	Description
Status[23]= flg_status_cport4	Status flag of capacitive port 4
Status[22]= flg_status_cport3	Status flag of capacitive port 3
Status[21]= flg_status_cport2	Status flag of capacitive port 2
Status[20]= flg_status_cport1	Status flag of capacitive port 1
Status[19]= flg_rstpwr	1 = Power-on reset caused jump into OTP / ext. EEPROM
Status[18]= flg_rstssn	1 = Pushed button caused jump into OTP / ext. EEPROM
Status[17]= flg_wdtalt	1 = Watchdog interrupt caused jump into OTP / ext. EEPROM
Status[16]= flg_endavg	1 = End of measurement caused jump into OTP / ext. EEPROM
Status[15]= flg_intav0	1 = Jump into OTP / ext. EEPROM in sleep mode
Status[14]= flg_ub_low	1 = Low voltage
Status[13]= flg_errtdc	1 = TDC error
Status[12]= reserved	1 = reserved
Status[11]= flg_err_cport	1 = Error at capacitive ports
Status[10]= flg_errprt	1 = Error at strain gauge ports
Status[09]= flg_timout	1 = Timeout TDC
Status[08]= flg_ext_interrupt	1 = DSP start by external interrupt
Status[07]= flg_cport4_r	1 = Rising edge at capacitive port 4, 0 = no edge
Status[06]= flg_cport3_r	1 = Rising edge at capacitive port 3, 0 = no edge
Status[05]= flg_cport2_r	1 = Rising edge at capacitive port 2, 0 = no edge
Status[04]= flg_cport1_r	1 = Rising edge at capacitive port 1, 0 = no edge
Status[03]= flg_cport4_f	1 = Falling edge at capacitive port 4, 0 = no edge
Status[02]= flg_cport3_f	1 = Falling edge at capacitive port 3, 0 = no edge
Status[01]= flg_cport2_f	1 = Falling edge at capacitive port 2, 0 = no edge
Status[00]= flg_cport1_f	1 = Falling edge at capacitive port 1, 0 = no edge

* Pin numbers in brackets = dice

The status of the inputs can be queried from the status registers at RAM address 250 to 252. Please see chapter 4.4.4 on page 4-17 for more details

6.5 Instruction Set

The complete instruction set of the PS09 consists of 69 core instructions that have unique op-codes decoded by the CPU.

6.5.1 Branch instructions

There are 3 principles of jumping within the code:

Jump. Absolute addressing within the whole address space of 8kB.

Branch. Relative to the actual address, jump within the address range of –128 to +127.

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.

6.5.2 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 points to the RAM address that is handled in the same way as an accumulator. Each operation on the accumulator affects the four flags. The status of the flags refers 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	EEPROM access OTP/external EPROM
and	bitclr	equal
eor	bitinv	getepr
nor	bitset	putepr
invert		addepr
nand		
nor		
or		

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

abs	**Absolute value of register**
Syntax:	abs p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	p1 := | p1 |
Flags affected:	C O S Z
Bytes:	2
Cycles:	2
Description:	Absolute value of register
Category:	Simple 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:	1 (p2 = ACCU) 4 (p2 = number)
Cycles:	1 (p2 = ACCU) 4 (p2 = number)
Description:	Addition of two registers or addition of a constant to a register
Category:	Simple arithmetic

addepr	
Syntax:	addepr x
Parameters:	ACCU[x]
Calculus:	x = x+Value (EEprom(rampointer))
Flags:	Z S C O
Bytes:	2
Cycles:	100..200
Description:	Adds the value from the content of the EEPROM register, currently addressed by the ram address pointer, to the X-Accumulator.
Category:	EEPROM access

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. With the 10Khz clock beeper application at the IO-Port may programmed with the microcontroller. Do not switch directly between CLK4MHz and CLK 10kHz.
Category:	Miscellaneous

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

clrwdt	**Clear watchdog**
Syntax:	clrwdt
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	
Description:	Clear watchdog. This opcode is used to clear the watchdog at the end of a 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:	Comparison of 2 registers by subtraction. Comparison 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

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 the OTP or the external EEPROM
Syntax:	equal p1
Parameters:	p1 = 24-Bit number
Calculus:	-
Flags affected:	-
Bytes:	3
Cycles:	
Description:	Write 3 bytes (p1) to configuration register of OTP/external EEPROM. The equal opcode is used to write 3 bytes of configuration data directly to a register. Therefore the opcode is simply used 16 times in the beginning of the assembler listing, fed with the configuration data given through p1. The configuration of the OTP/ external 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. Use this opcode to provide your own configuration instead of the standard configuration.
Category:	OTP/ External 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 temporarily a place in the program counter stack (explanation see below).
Category:	EEPROM Access

getflag	**Set S and Z flags**
Syntax:	getflag p1
Parameters:	p1 = ACCU [x,y,z,r]

Calculus:	signum := set if p1 < 0
	notequalzero := set if p1 <> 0

Flags affected:	S Z
Bytes:	1
Cycles:	1
Description:	Set the signum and notequalzero flag according to the addressed register, content of the register is not affected
Category:	Simple arithmetic

goto	**jump without condition**
Syntax:	goto p1
Parameters:	p1 = JUMPLABEL
Calculus:	PC:= p1
Flags affected:	-
Bytes:	2 (relative jump)
3 (absolute jump)	
Cycles:	3 (relative jump) 4 (absolute jump)
Description:	Jump without condition. Program counter is set to target address. The target address is given by using a jump label. Jump range: 0< address < 8kB 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

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

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

initTDC	**Initialize TDC**
Syntax:	initTDC
Parameters:	-
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	3
Description:	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). Jump range: 0< address < 8kB
Category:	Unconditional Jump

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

move	**Move**
Syntax:	move p1,p2

Parameters:	p1 = ACCU [x,y,z,r]
	p2 = ACCU [x,y,z,r] or 24-bit number

Calculus:	p1 := p2
Flags affected:	S Z
Bytes:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
Description:	Move content of p2 to p1 (p1=ACCU, p2=ACCU) Move constant to p1 (p1=ACCU, p2=NUMBER)
Category:	RAM access

mult24	**Signed 24-Bit multiplication**
Syntax:	mult24 p1,p2

Parameters:	p1 = ACCU [x,y,z,r]
	p2 = ACCU [x,y,z,r]

Calculus:	p1 := (p1 * p2) >> 24
Flags affected:	S & Z of p1
Bytes:	2
Cycles:	30
Description:	Signed multiplication of 2 registers like mult48, but only the 24 higher bits of the multiplication of 2 registers, result is stored in p1
Category:	Complex arithmetic

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 PSØ9 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

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 lower 8 bits of register to internal EEPROM**
Syntax:	putepr p1
Parameters:	p1 = ACCU [x,y,z,r]
Calculus:	EEPROM register (addressed by RAM address pointer) := p1 [7:0]
Flags affected:	-
Bytes:	4
Cycles:	~12.5 ms
Description:	Put register into EEPROM. The lower 8 bits of the addressed register p1 is moved to the EEPROM (the EEPROM register address is set by the RAM address pointer). EEPROM bytes 0 to 127 are accessible via ‚putepr‘, bysetting the RAM address pointer to addresses 0 to 127 respectively. This opcode needs temporarily a place in the program counter stack (explanation see below). It is recommended not to use putepr in combination with the skip opcodes due to relatively longer execution times (~30ms).
Category:	EEPROM access

ramadr	**Set RAM address pointer**
Syntax:	ramadr p1
Parameters:	p1 = 8-Bit number
Calculus:	-
Flags affected:	-
Bytes:	2
Cycles:	2
Description:	Set pointer to RAM address (range: 0...255)
Category:	RAM access

rotL	**Rotate left**
Syntax:	rotL p1(,p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none

Calculus:	p1 := p1<< 1+ carry; carry:=MSB(x) (in case rotL p1, without p2)
	p1 := repeat (p2) rotL p1(in case rotL p1,p2)

Flags affected:	C O S Z (of the last step)
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
Description:	Rotate p1 left –> shift p1 register to the left, fill LSB with carry, MSB is placed in carry register Rotate p1 left p2 times with carry –> shift p1 register p2 times to the left, in each step fill LSB with the carry and place the MSB in the carry
Category:	Shift and rotate

rotR	**Rotate right**
Syntax:	rotR p1(,p2)
Parameters:	p1 = ACCU [x,y,z,r] p2 = 4-Bit number or none

Calculus:	p1 := p1>> 1+ carry;
	carry: =MSB(x) (in case rotR p1, without p2)
	p1 := repeat (p2) rotR p1 (in case rotR p1,p2)

Flags affected:	C O S Z (of the last step)
Bytes:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
Cycles:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
Description:	Rotate p1 right –> shift p1 register to the right, fill MSB with carry, LSB is placed in carry register Rotate p1 right p2 times with carry –> shift p1 register p2 times to the right, in each step fill MSB with the carry and place the LSB in the carry
Category:	Shift and rotate

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

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]
	p3 = 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

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

6.6 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 F000 h. There a first check is done whether the ALU was activated after a reset or not.

In case of a reset, the flag otp_pwr_cfg is checked to decide whether the auto-configuration data from the OTP/external EEPROM have to be copied into the RAM or not.

Subsequently, the flag otp_pwr_prg is checked to decide whether OTP/ external EEPROM user code (starting at address 48) ought to be executed. In stand-alone operation this is reasonable and otp_pwr_cfg bit should be 1. In front end operation this is unlikely and with otp_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 otp_usr_prg is checked to decide whether a jump into the user code in OTP/external EEPROM (address 48) must be performed or not. Again, in stand-alone operation otp_usr_prg =1 is reasonable, in front-end operation otp_usr_prg = 0 will be more likely.

In the user code in the OTP / external EEPROM 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.

6.6.1 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 otp_pwr_prg = 1 the user code at EEPROM address 48 is started.

6.6.2 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 224+22; bit 17 indicates a timeout of the watchdog timer.

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

6.6.3 External Reset on Pin 6

In stand-alone mode (if Mode pin is unconnected) it is possible to apply an external power-on at pin 6 (SPI_CSN_RST). This can be used for a reset button. The status of the button can be requested from status bit flg_rstssn in register 224+22, bit 18.

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

6.6.4 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/Os. A start-up of the microprocessor from sleep mode is indicated by status bit flg_intav0 in register 224+22, bit 22.

Configuration:	tdc_sleepmode	Register 1, Bit 17
	tdc_conv_cnt [7:0]	Register 0, Bits 23 to 16

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.

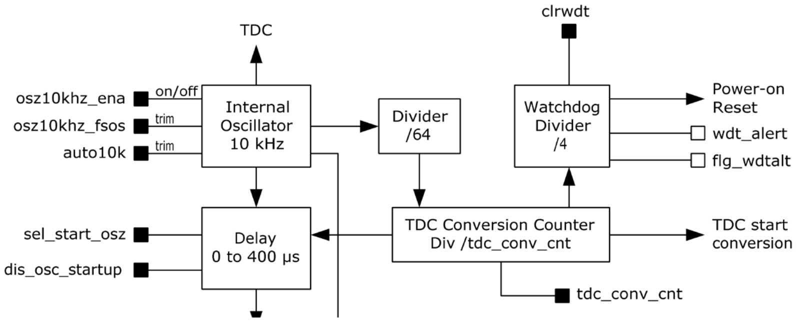
After running in sleep mode the TDC has to be reinitialized for measurements.

6.7 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 6.10 Clock Generation



6.8 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 user code does not run to end.

It is possible to switch off the watchdog when controlling the PSØ9 by the SPI interface (Mode pin is connected to 0) sending SPI opcode watch_dog_off. Further the watchdog is reset by each signal edge at the SPI_CSN_RST pin.

6.9 Timer

PSØ81has a real time counter that counts automatically after a power-on reset in periods of 12.8 ms. The value of this timer can be read out at address 254, it is updated at the end of each measurement. The counter rolls over at 2^24 bit, which corresponds to a period of 46 hours

7 Miscellaneous

7.1 Bug Report

Date	Reported Bug	Solution (if any)
27 May 2011	Increased noise in case the specific combination of [Wheatsone mode + Continuous mode + IIC read communication]	- Use Wheastone mode in combination with SPI - Use [Wheatstone mode + IIC mode] in combination with single conversion mode

7.2 Literature Guide

Datasheets

Title	Document-No	Date
PSØ9-EVA-KIT Evaluation System for PSØ9	DB_PS09_EVA V0.1	May 2011
ALCS-350 V2 Load Cell Simulator	DB_ALCS_V2 V0.1	August 2009

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 **PICO**STRAIN conveniently	WP004 V1.0	March 2010

Application Notes

Title	Document-No	Date
Meterological Investigations of PSØ81 Determining Zero Drift and Gain Drift	AN018 V1.0	July 2008
Strain Gauge Wiring with **PICO**STRAIN	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

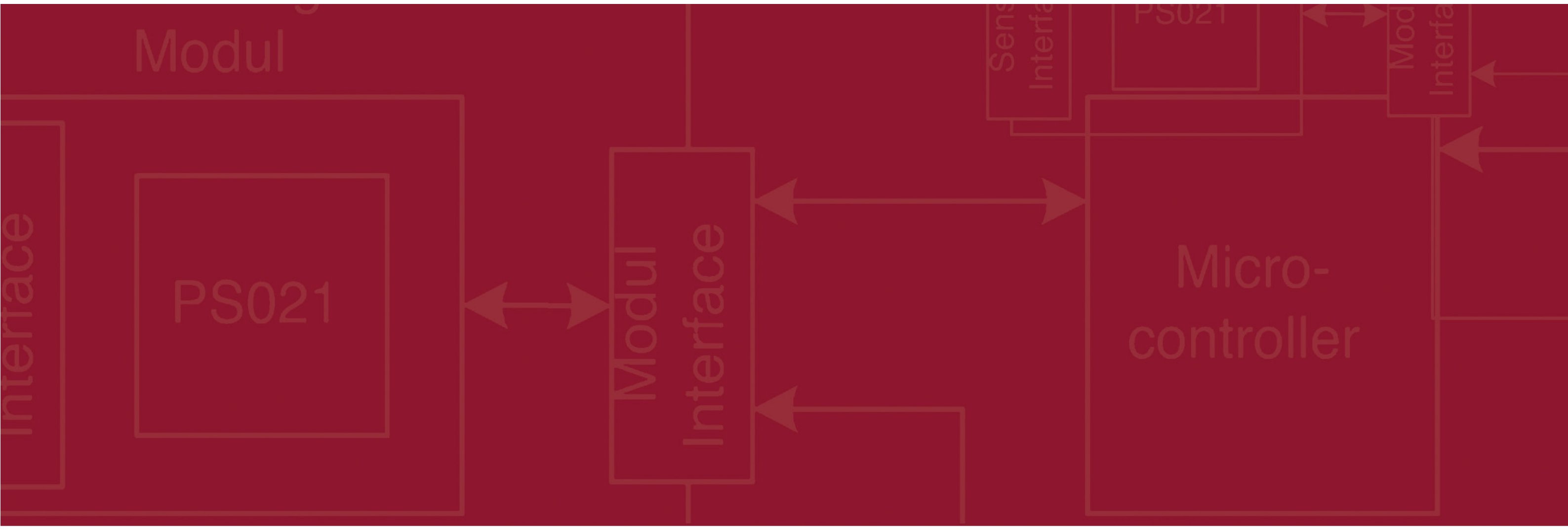
All available documents can be downloaded from the acam website at:

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

7.3 Document History

01.03.2011	First release of preliminary version
21.03.2011	Release V0.2 of preliminary version
27.05.2011	Release Version 0.3. Major change is cancellation of the QFN32 package. ONly QFN40 will be available.

PICOSTRAIN



acam-messelectronic gmbh
Am Hasenbiel 27
76297 Stutensee-Blankenloch
Germany / Allemagne
ph. +49 7244 7419 - 0
fax +49 7244 7419 - 29
e-mail: support@acam.de
www.acam.de