

# PCap04

## Capacitance-to-Digital Converter

### General Description

PCap04 is a capacitance-to-digital converter (CDC) with integrated digital signal processor (DSP) for on-chip data post-processing. Its front end is based on **ams** PICOCAP principle.

This conversion principle offers outstanding flexibility with respect to power consumption, resolution and speed. Further, PCap04 covers a wide capacitance input range from a few femtofarads up to several hundreds of nanofarads. It is easy to configure the PCap04 for different capacitance measurement tasks, i.e. single as well as differential sensors in both, grounded or floating connection. The on-chip DSP allows to implement sensor algorithms like linearization and temperature compensation, with data output in a digital (SPI or IIC) or analog (PDM/PWM) way.

*Ordering Information and Content Guide appear at end of datasheet.*

### Key Benefits & Features

The benefits and features of PCap04, Capacitance-to-Digital Converter are listed below:

**Figure 1:**  
Added Value of Using PCap04

| Benefits   | Features   |
|--|--|
| <ul style="list-style-type: none"> <li>High flexibility: Easy adaption to various applications - ultra low power, high resolution or high speed - just by configuration</li> </ul> | <ul style="list-style-type: none"> <li>Up to 6 capacitors grounded, 3 capacitors floating</li> <li>Capacitance range 1pF to 100nF</li> <li>Internal reference 1pF to 31pF</li> <li>Integrated guard driver</li> </ul>                    |
| <ul style="list-style-type: none"> <li>High resolution or high speed</li> </ul>  | <ul style="list-style-type: none"> <li>Up to 8aF at 2.5Hz and 10pF base capacitance</li> <li>Up to 50kHz sample rate</li> <li>Up to 20-bit resolution</li> </ul>   |
| <ul style="list-style-type: none"> <li>On-chip DSP for sensor algorithms, signal post-processing including linearization and temperature compensation</li> </ul>                   | <ul style="list-style-type: none"> <li>32-bit DSP</li> <li>3k ROM code, 1k NVRAM</li> <li>96 x 32 bit RAM</li> <li>SPI / IIC interface</li> <li>PDM / PWM outputs, GPIO</li> </ul>   |
| <ul style="list-style-type: none"> <li>On-chip and external temperature measurement capability</li> </ul>  | <ul style="list-style-type: none"> <li>Supply voltage 2.1/3.0V to 3.6V</li> <li>Operating current down to 3µA</li> <li>PCap04-Bxxx -40°C to 85°C</li> <li>PCap04-Axxx -40°C to 125°C</li> <li>QFN24 or die (1.588mm x 1.46mm)</li> </ul> |

## Applications MEMS Sensors

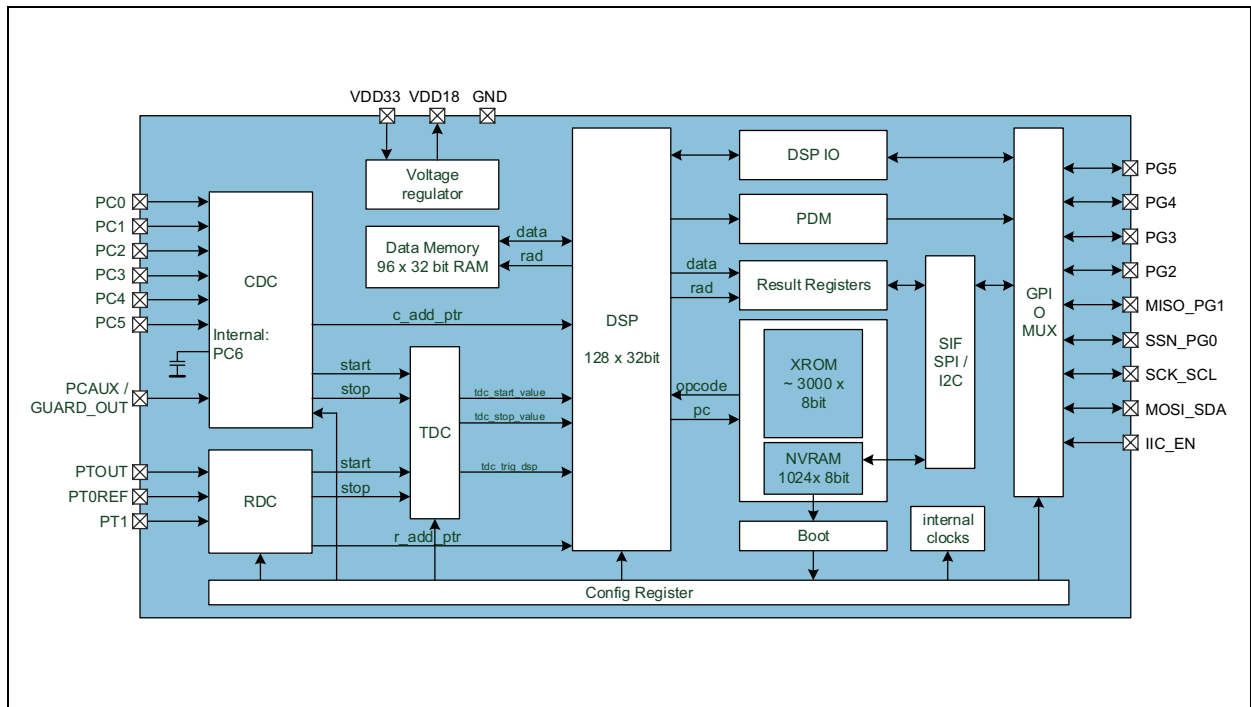
The PCap04 applications include:

- Position sensors
- Pressure sensors
- Force sensors
- Proximity sensor
- Acceleration sensors
- Inclination sensors
- Humidity sensors
- Dewpoint sensors
- Tilt sensors
- Angle sensors
- Wireless applications
- Level sensors

## Block Diagram

The functional blocks of this device are shown below:

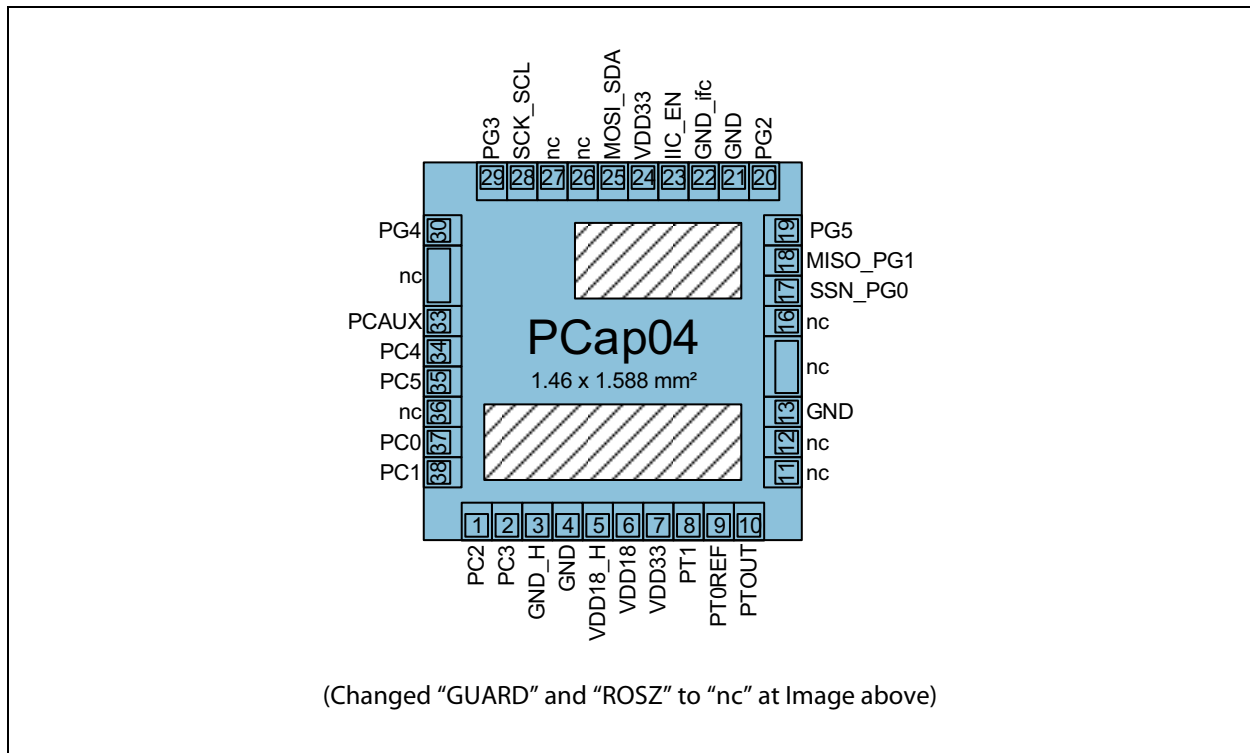
**Figure 2:**  
Functional Blocks of PCap04



### Pin Assignments

PCap04 will be available in QFN24 package or as pure die.

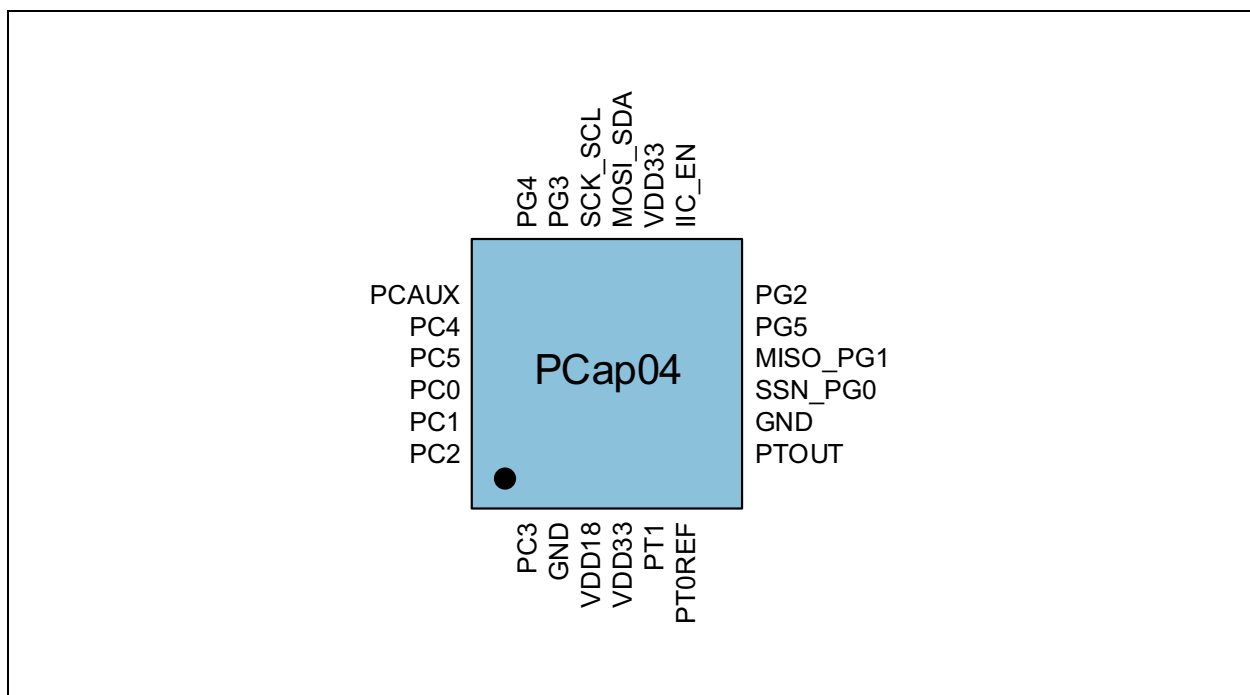
**Figure 3:**  
Pin Diagram of PCap04 Die



**Note(s):**

1. Die dimensions: 1.588 mm x 1.46 mm (w/ seal) with pad pitch 120 μm, pad opening is 85 μm x 85 μm. Thickness 290μm.

**Figure 4:**  
Pin Diagram of PCap04 QFN24



## Pin Description

Figure 5:  
Pin Description of PCap04

| Pin Number |            | Pin Name             | Description   | If Not Used |
|------------|------------|----------------------|---|-------------|
| Die        | 24-Pin QFN |                      |   |             |
| 1          | 24         | PC2                  | CDC port  | open        |
| 2          | 1          | PC3                  | CDC port  | open        |
| 3          |            | GND_H                |   | GND         |
| 4          | 2          | GND                  |   | GND         |
| 5          |            | VDD18_H              |   | VDD18       |
| 6          | 3          | VDD18 <sup>(1)</sup> | Core supply voltage   | VDD18       |
| 7          | 4          | VDD33 <sup>(2)</sup> | I/O supply voltage  | VDD33       |
| 8          | 5          | PT1                  | RDC port (temperature sensor)   | open        |
| 9          | 6          | PT0REF               | RDC port (temperature sensor or external reference)                       | open        |
| 10         | 7          | PTOUT <sup>(3)</sup> | Discharge capacitor for RDC   | open        |
| 11         |            | n.c.                 | No pad  |             |
| 12         |            | n.c.                 | No pad  |             |
| 13         | 8          | GND                  |   |             |
| 14/15      |            | n.c.                 | No pad  |             |
| 16         |            | n.c.                 | Always open   |             |
| 17         | 9          | SSN_PG0              | Serial select line. Otherwise, general purpose I/O port                   | open        |
| 18         | 10         | MISO_PG1             | Master in/Slave out when SPI is used. Otherwise, general purpose I/O port | open        |
| 19         | 11         | PG5                  | General purpose I/O port  | open        |
| 20         | 12         | PG2                  | General purpose I/O port  | open        |
| 21         |            | GND                  |   | GND         |
| 22         |            | GND_ifc              |   | GND         |
| 23         | 13         | IIC_EN               | Serial interface select, 0 = SPI enable<br>1 = IIC enable                 |             |
| 24         | 14         | VDD33                |   | VDD33       |

| Pin Number |            | Pin Name | Description   | If Not Used |
|------------|------------|----------|---|-------------|
| Die        | 24-Pin QFN |          |   |             |
| 25         | 15         | MOSI_SDA | Master out/Slave in when SPI is used. Otherwise, serial data for IIC  |             |
| 26         |            | n.c.     | Always open   |             |
| 27         |            | n.c.     | Always open   |             |
| 28         | 16         | SCK_SCL  | Serial clock for SPI/IIC  |             |
| 29         | 17         | PG3      | General purpose I/O port  | open        |
| 30         | 17         | PG4      | General purpose I/O port  | open        |
| 31/32      |            | n.c.     | No pad  |             |
| 33         | 19         | PCAUX    | Auxiliary port<br>- For external compensation capacitance<br>or<br>- External discharge resistor<br>- Guarding output | open        |
| 34         | 20         | PC4      | CDC port  | open        |
| 35         | 21         | PC5      | CDC port  | open        |
| 36         |            | n.c.     | Always open   | open        |
| 37         | 22         | PC0      | CDC port  | open        |
| 38         | 23         | PC1      | CDC port  | open        |

**Note(s):**

1. Connect buffer capacitor  $\geq 4.7\mu\text{F}$
2. Connect buffer capacitor  $\geq 10\mu\text{F}$
3. Connect 10nF C0G
4. Center pad is internally connected to GND. No wires other than GND are allowed underneath.
5. It is recommended to not use the center pad. Too much solder paste could reduce solder quality.
6. Suitable socket: e.g. Plastronics 32QN50S15050D.

## Pad Coordinates

**Figure 6:**  
Pad Coordinates of PCap04

| Pad Number | Description | X-POS (µm) | Y-POS (µm) |
|------------|-------------|------------|------------|
| 1          | PC2         | 260.0      | 59.5       |
| 2          | PC3         | 380.0      | 59.5       |
| 3          | GND_H       | 488.0      | 59.5       |
| 4          | GND         | 608.0      | 59.5       |
| 5          | VDD18_H     | 728.0      | 59.5       |
| 6          | VDD18       | 848.0      | 59.5       |
| 7          | VDD33       | 968.0      | 59.5       |
| 8          | PT1         | 1088.0     | 59.5       |
| 9          | PT0REF      | 1208.0     | 59.5       |
| 10         | PTOUT       | 1328.0     | 59.5       |
| 11         | n.c.        | No pad     | No pad     |
| 12         | n.c.        | No pad     | No pad     |
| 13         | GND         | 1528.5     | 501.0      |
| 14/15      | n.c.        | No pad     | No pad     |
| 16         | n.c.        | No pad     | No pad     |
| 17         | SSN_PG0     | 1528.5     | 965.0      |
| 18         | MICO_PG1    | 1528.5     | 1085.0     |
| 19         | PG5         | 1528.5     | 1205.0     |
| 20         | PG2         | 1333.0     | 1400.5     |
| 21         | GND         | 1213.0     | 1400.5     |
| 22         | GND_ifc     | 1093.0     | 1400.5     |
| 23         | IIC_EN      | 973.0      | 1400.5     |
| 24         | VDD33       | 853.0      | 1400.5     |
| 25         | MOSI_SDA    | 733.0      | 1400.5     |
| 26         | n.c.        | No pad     | No pad     |
| 27         | n.c.        | No pad     | No pad     |
| 28         | SCK_SCL     | 374.7      | 1400.5     |

| Pad Number | Description | X-POS ( $\mu\text{m}$ ) | Y-POS ( $\mu\text{m}$ ) |
|------------|-------------|-------------------------|-------------------------|
| 29         | PG3         | 254.7                   | 1400.5                  |
| 30         | PG4         | 59.5                    | 1205.0                  |
| 31/32      | n.c.        | No pad                  | No pad                  |
| 33         | PCAUX       | 59.5                    | 859.9                   |
| 34         | PC4         | 59.5                    | 739.8                   |
| 35         | PC5         | 59.5                    | 619.8                   |
| 36         | n.c.        | No pad                  | No pad                  |
| 37         | PC0         | 59.5                    | 379.8                   |
| 38         | PC1         | 59.5                    | 259.8                   |

**Note(s):**

1. Pad coordinates are center/center (x/y-direction), relative to die origin die dimensions: 1.588mm x 1.46mm (with seal, 15 $\mu\text{m}$  each side) with pad pitch 120  $\mu\text{m}$ , pad opening is 85 $\mu\text{m}$  x 85 $\mu\text{m}$

## Absolute Maximum Ratings

Stresses beyond those listed under [Absolute Maximum Ratings](#) may cause permanent damage to the device. These are stress ratings only. Functional operation of the device at these or any other conditions beyond those indicated under [Electrical Characteristics](#) is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**Figure 7:**  
Absolute Maximum Ratings

| Symbol  | Parameter  | Min        | Max       | Units    | Comments               |
|---|--|------------|-----------|----------|------------------------|
| <b>Electrical Parameters</b>  |  |            |           |          |                        |
| $V_{DD} / V_{GND}$  | Supply Voltage to Ground                                     | -0.3       | 4.0       | V        |                        |
| $V_{IN}$  | Input Pin Voltage to Ground                                  | -0.3       | 4.0       | V        |                        |
| $I_{SCR}$   | Input Current (latch-up immunity) @125°C                     | ±100       |           | mA       | JEDEC JESD78D Nov 2011 |
| <b>Continuous Power Dissipation (<math>T_A = 70^\circ\text{C}</math>)</b> |  |            |           |          |                        |
| $P_T$   | Continuous Power Dissipation                                 |            | 1.44      | mW       |                        |
| <b>Electrostatic Discharge</b>  |  |            |           |          |                        |
| $ESD_{HBM}$   | Electrostatic Discharge HBM                                  | ± 1000     |           | V        | JS-001-2014            |
| <b>Temperature Ranges and Storage Conditions</b>                          |  |            |           |          |                        |
| $T_A$   | Operating Ambient Temperature<br>PCap04-Bxxx<br>PCap04-Axxx  | -40<br>-40 | 85<br>125 | °C<br>°C |                        |
| $R_{THJA}$  | Junction to Ambient Thermal Resistance                       |            | 28        | °C/W     |                        |
| $T_J$   | Operating Junction Temperature<br>PCap04-Bxxx<br>PCap04-Axxx |            | 85<br>125 | °C<br>°C |                        |
| $T_{STRG}$  | Storage Temperature Range                                    | -55        | 150       | °C       |                        |



| Symbol                   | Parameter   | Min | Max | Units  | Comments   |
|--------------------------|---|-----|-----|--------|--|
| $T_{\text{BODY}}$        | Maximum Package Body Temperature during Reflow                    |     | 260 | °C     | IPC/JEDEC J-STD-020<br>The reflow peak soldering temperature (body temperature) is specified according to IPC/JEDEC J-STD-020<br>"Moisture/Reflow Sensitivity Classification for Non-hermetic Solid State Surface Mount Devices." The lead finish for Pb-free leaded packages is "Matte Tin" (100% Sn) |
| $RH_{\text{NC}}$         | Relative Humidity (non-condensing)                                | 5   | 85  | %      |  |
| MSL                      | Moisture Sensitivity Level  | 3   |     |        | Maximum floor life time of 168h  |
| $t_{\text{STRG-DOF}}$    | Storage Time for DOF/die or wafers on foil                        |     | 3   | months | Refers to indicated date of packing  |
| $T_{\text{STRG-DOF}}$    | Storage Temperature for DOF/die or wafers on foil                 | 17  | 28  | °C     |  |
| $RH_{\text{OPEN-DOF}}$   | Relative Humidity for DOF/die or wafers on foil in open package   |     | 15  | %      | Opened package   |
| $RH_{\text{UNOPEN-DOF}}$ | Relative Humidity for DOF/die or wafers on foil in sealed package | 40  | 60  | %      | Sealed bag   |
| $t_{\text{STRG-WP}}$     | Storage Time for WP/wafers or die in waffle pack                  |     | 6   | months | 17-28°C<br>40-60% relative humidity<br>storage in original Ultrapak boxes  |
| $t_{\text{STRG-WP}}$     | Storage Time for WP/wafers or die in waffle pack                  |     | 2   | years  | 19-25°C<br><15% relative humidity<br>storage in closed cabinet with dry air  |
| $t_{\text{STRG-WP}}$     | Storage Time for WP/wafers or die in waffle pack                  |     | 5   | years  | 19-25°C<br><5% relative humidity<br>storage in closed cabinet with dry air   |
| $t_{\text{STRG-WP}}$     | Storage Time for WP/wafers or die in waffle pack                  |     | 10  | years  | 19-25°C<br><5% relative humidity<br>storage in closed cabinet and closed Ultrapak box with safeguarded Nitrogen atmosphere   |

## Electrical Characteristics

Characteristics indicate conditions for which the device is guaranteed to be functional. For details on the test conditions see the notes at the table footer.

**Figure 8:**  
Electrical Characteristics of PCap04

| Symbol            | Parameter   | Conditions  | Min                  | Typ                         | Max                          | Unit    |
|-------------------|---|---|----------------------|-----------------------------|------------------------------|---------|
| $V_{DD}$          | Supply Voltage <sup>(4)</sup>                       | Relative to GND<br>Power-down: VDD has to be <0.05V before power-up | 2.1                  |                             | 3.6                          | V       |
| $V_{DD}$          | Supply Voltage                                      | NVRAM recall<br>-Bx: -40°C to 85°C<br>-Ax: -40°C to 125°C           | 2.5                  |                             | 3.6                          | V       |
| $V_{DD}$          | Supply Voltage                                      | NVRAM store<br>-25°C to 60°C  | 3.0                  |                             | 3.6                          | V       |
| $V_{IO\_DIGITAL}$ | Digital Ports Input Voltage <sup>(4)</sup>          | Relative to GND   | -0.6                 | 3.3                         | $V_{DD} + 0.6$<br>$\leq 3.6$ | V       |
| $V_{IO\_DIGITAL}$ | Digital Ports Input Switching Levels <sup>(2)</sup> | HIGH to LOW<br>LOW to HIGH  | $0.7 * V_{DD}$       |                             | $0.3 * V_{DD}$               | V       |
| $V_{OH}$          | Digital Ports Output Voltage <sup>(2)</sup>         | HIGH  | $V_{DD} - 0.4$       |                             | $V_{DD} + 0.1$               | V       |
| $V_{OL}$          | Digital Ports Output Voltage <sup>(2)</sup>         | LOW   | -0.1                 |                             | 0.4                          | V       |
| $I_{leakH}$       | Digital Ports Leakage <sup>(2)</sup>                | IIC_EN= $V_{DD}$<br>HIGH  | -0.1                 |                             | 1.0                          | $\mu A$ |
| $I_{leakPU}$      | Digital Ports Leakage <sup>(2)</sup>                | Internal pull-ups   | -2.8                 |                             | -5.2                         | $\mu A$ |
| $I_{leakL}$       | Digital Ports Leakage <sup>(2)</sup>                | IIC_EN= $V_{DD}$<br>LOW   | -1.0                 |                             | 0.1                          | $\mu A$ |
| $I_{ddq}$         | Static Supply Current <sup>(2)</sup>                | -40°C<br>35°C<br>85°C<br>125°C                                      |                      | 1.28<br>1.6<br>4.26<br>22.5 | -<br>5<br>18<br>50           | $\mu A$ |
| $R_p$             | Precharge Resistance <sup>(2)</sup>                 | R_C_PCHG0<br>R_C_PCHG1  | 7<br>126             | 10<br>180                   | 13<br>234                    | kOhm    |
| $R_D$             | Discharge Resistance <sup>(2)</sup>                 | R_DCHG0<br>R_DCHG1<br>R_DCHG2<br>R_DCHG3                            | 7<br>21<br>63<br>126 | 10<br>30<br>90<br>180       | 13<br>39<br>117<br>234       | kOhm    |

| Symbol  | Parameter   | Conditions   | Min  | Typ                              | Max                          | Unit   |
|---|---|--|--|----------------------------------|------------------------------|--------|
| C <sub>ref0</sub><br>C <sub>ref1</sub><br>C <sub>ref2</sub><br>C <sub>ref3</sub><br>C <sub>ref4</sub> | Internal Reference<br>(5)                                     |  |  | 0.4<br>1.4<br>3.4<br>7.4<br>15.4 |                              | pF     |
| C <sub>R</sub>  | Internal RDC<br>Discharge Cap (2)                             | PTOUT n.c.   | 65   | 94                               | 122                          | pF     |
| I <sub>C_G7</sub><br>I <sub>C_GLow</sub>  | Driver 7<br>Driver Low (3)                                    | PC0 = V <sub>DD</sub><br>PC0 = V <sub>DD</sub> * 0.05<br>PCAUX = V <sub>DD</sub> |  | -3.5<br>-9                       |                              | mA     |
| V <sub>GUARD0</sub><br>V <sub>GUARD1</sub><br>V <sub>GUARD2</sub><br>V <sub>GUARD3</sub>              | Gain Guarding<br>Opamp (3)                                    | PC0 = 1.0V   |  | 1.00<br>1.01<br>1.02<br>1.03     |                              | V      |
| R <sub>R_DIS_SEN</sub>  | RDC Temperature<br>Sensor (2) (6)                             | -40 °C<br>35°C<br>85 °C<br>125 °C  | -<br>1120<br>1200<br>1360  | 1047<br>1330<br>1572<br>1688     | -<br>1680<br>1800<br>2040    | Ohm    |
| R <sub>R_DIS_REF</sub>  | RDC Reference (2)   |  | 1100   | 1476                             | 1690                         | Ohm    |
| C <sub>R</sub>  | RDC Internal<br>Capacitance (2)                               |  | 65   | 94                               | 122                          | pF     |
| f <sub>OLF0</sub><br>f <sub>OLF1</sub><br>f <sub>OLF2</sub><br>f <sub>OLF3</sub>                      | OLF Frequency (2)   | High temperature<br>minimum values, low<br>temperature maximum<br>values         | 3.6<br>25<br>47.6<br>92.8  | 10<br>60<br>100<br>200           | 16.5<br>96.4<br>171.4<br>305 | kHz    |
| f <sub>OHF</sub>  | OHF Frequency (2)   |  | 1.36   | 2                                | 2.6                          | MHz    |
|   | NVRAM Data<br>Retention (4) (7)<br>PCap04-Bxxx<br>PCap04-Axxx | 3.0V to 3.6V<br>85°C<br>125°C  | 20<br>20   |                                  |                              | Years  |
|   | NVRAM Endurance<br>(4) (7)<br>PCap04-Bxxx<br>PCap04-Axxx      | 25°C<br>85°C<br>25 °C<br>125 °C  | 10 <sup>4</sup><br>10 <sup>3</sup><br>10 <sup>5</sup><br>10 <sup>4</sup> |                                  |                              | Cycles |

**Note(s):**

- 100% production tested
- 100% production tested at 85°C (-Bxxx)/125°C (-Axxx) wafer sort and guaranteed by design and characterization at specified temperatures.

- 3. Sample tested only
- 4. Parameter is guaranteed by design and characterization testing.
- 5. Parameter is a typical value only
- 6. 100% production tested at 25°C and guaranteed by design and characterization for industrial temperature range
- 7. **Important:** We guarantee the data for data retention and endurance only under the assumption, that the customer does not change the registers 62 and 63 and NVRAM adr 654 to 959 (Unique ID). In addition, it is mandatory to follow the given procedure for ERASE NVRAM as described in section [NVRAM](#) and [ROM](#) precisely. Otherwise, we do no longer guarantee the data retention time and endurance cycles.

**Figure 9:**  
**Total Current I [μA] as a Function of Conversion Rate (CONV\_TIME) and Resolution (C\_AVRG) in Triggered Mode**

| LP Oscillator Freq. [kHz] | CONV_TIME | Measure Rate [Hz] | I [μA]                         |        |        |        |        |        |
|---------------------------|-----------|-------------------|--------------------------------|--------|--------|--------|--------|--------|
|                           |           |                   | C_AVRG (RMS Resolution [Bits]) |        |        |        |        |        |
|                           |           |                   | 1                              | 4      | 16     | 64     | 256    | 1024   |
|                           |           |                   | (13.6)                         | (14.6) | (15.6) | (16.6) | (17.8) | (18.6) |
| 50                        | 10000     | 2.5               | 2.3                            | 2.5    | 2.7    | 4.6    | 11.8   | 44.0   |
| 50                        | 2500      | 10                | 2.8                            | 3.3    | 4.6    | 12.2   | 43.8   |        |
| 50                        | 1250      | 20                | 3.3                            | 4.2    | 7.9    | 23.9   |        |        |
| 50                        | 625       | 40                | 4.6                            | 6.0    | 14.0   |        |        |        |
| 50                        | 250       | 100               | 8.3                            | 12.0   | 32.2   |        |        |        |
| 50                        | 125       | 200               | 14.3                           | 22.3   |        |        |        |        |
| 50                        | 50        | 500               | 32.6                           | 53.8   |        |        |        |        |
| 50                        | 25        | 1000              | 63.9                           |        |        |        |        |        |
| 50                        | 12        | 2080              | 90                             |        |        |        |        |        |
| 200                       | 24        | 4160              | 156                            |        |        |        |        |        |
| 200                       | 12        | 9320              | 305                            |        |        |        |        |        |

**Note(s):**

- 1. Temperature measurement in addition to capacitive measurement will add between 2μA and 10μA approximately, depending on speed. Total consumption values below 30 μA may be obtained only when driving the on-chip 1.8 volts core supply generator in an energy-saving mode; ultimate microampere savings also demand to slow down the DSP. Typical data.

## CDC Characteristics

**Figure 10:**  
Electrical Characteristics

| Rate<br>[Hz] | Floating<br>Fully Compensated |                           |          | Grounded<br>Internally Compensated |                           |          |
|--------------|-------------------------------|---------------------------|----------|------------------------------------|---------------------------|----------|
|              | RMS<br>Noise<br>[aF]          | Eff. Resolution<br>[Bits] |          | RMS<br>Noise<br>[aF]               | Eff. Resolution<br>[Bits] |          |
|              |                               | 10pF Base                 | 1pF Span |                                    | 10pF Base                 | 1pF Span |
| 2.5          | 8                             | 20.2                      | 16.9     | 9                                  | 20.1                      | 16.8     |
| 5            | 12                            | 19.7                      | 16.4     | 13                                 | 19.6                      | 16.3     |
| 10           | 19                            | 19.0                      | 15.7     | 19                                 | 19.0                      | 15.7     |
| 25           | 28                            | 18.4                      | 15.1     | 26                                 | 18.5                      | 15.2     |
| 100          | 56                            | 17.4                      | 14.1     | 52                                 | 17.5                      | 14.2     |
| 250          | 91                            | 16.7                      | 13.4     | 78                                 | 17.0                      | 13.7     |
| 1000         | 156                           | 16.0                      | 12.7     | 148                                | 16.0                      | 12.7     |
| 2000         | 218                           | 15.5                      | 12.2     | 192                                | 15.6                      | 12.3     |
| 4000         | 328                           | 14.9                      | 11.6     | 272                                | 15.1                      | 11.8     |
| 8000         |                               |                           |          | 385                                | 14.6                      | 11.3     |

**Note(s):**

1. Typical capacitive noise and resolution vs. output data rate, 10pF base + 1pF span, fast settle, MR1, V = 3.0V. Span means the maximum variation of the sensor capacitance in the application. The table gives the root mean-square (RMS) noise in aF as a function of output data rate in Hz, measured at 3.0V supply voltage using the maximum possible sample size for in-chip averaging at the minimum possible cycle time. Bit values are calculated as a binary logarithm of noise over the span ( $BITs = \ln(\text{span}/\text{noise})/\ln(2)$ ). The measurements have been done with the PCap04 evaluation board, with fixed C0G ceramic capacitors, configuration for maximum resolution.

Both, sensor and reference are connected “floating” or “grounded”, as indicated. In floating mode compensation mechanisms for both internal and external stray capacitances are activated, in grounded mode only the internal compensation is active.

**Figure 11:**  
Voltage-Dependent Offset Error (PSRR)

| Base /Gain | Mode                                | 2.4V  | 2.7V  | 3.0V | 3.3V  | 3.6V  |
|------------|-------------------------------------|-------|-------|------|-------|-------|
| 10 pF      | Single-ended, internal compensation | < 1fF | < 1fF | 0fF  | < 1fF | < 1fF |
| 150 pF     |                                     | < 1fF | < 1fF | 0fF  | < 1fF | < 1fF |
| 10 pF      | Floating, full compensation         | < 1fF | < 1fF | 0fF  | < 1fF | < 1fF |
| 150 pF     |                                     | < 1fF | < 1fF | 0fF  | < 1fF | < 1fF |

**Figure 12:**  
Voltage-Dependent Gain Error (PSRR)

| Base /Gain   | Mode                                | 2.4V   | 2.7V   | 3.0V | 3.3V    | 3.6V    |
|--------------|-------------------------------------|--------|--------|------|---------|---------|
| 10pF / 4.7pF | Single-ended, internal compensation | 1.9fF  | 0.9fF  | 0fF  | - 0.7fF | - 1.2fF |
| 150pF / 47pF |                                     | - 22fF | - 10fF | 0fF  | 0.7fF   | 20fF    |
| 10pF / 4.7pF | Floating, full compensation         | 0.6fF  | 6fF    | 0fF  | - 0.7fF | - 1.5fF |
| 150pF / 47pF |                                     | - 11fF | - 11fF | 0fF  | 8fF     | 19fF    |

**Figure 13:**  
Temperature-Dependent Offset and Gain Error

| Error Type   | Capacitance [pF] | Mode                                | Temperature Range | Typ. Drift     |
|--------------|------------------|-------------------------------------|-------------------|----------------|
| Offset drift | 10               | Single-ended, internal compensation | -10°C to 85°C     | 2.5fF          |
|              | 150              |                                     |                   | 9fF            |
|              | 10               | Floating, full compensation         |                   | 1.5fF          |
|              | 150              |                                     |                   | 12fF           |
| Gain drift   | 10 + 4.7         | Single-ended, internal compensation |                   | 42fF = 94ppm/K |
|              | 150 + 47         |                                     |                   | 69fF = 15ppm/K |
|              | 10 + 4.7         | Floating, full compensation         |                   | 8fF = 18ppm/K  |
|              | 150 + 47         |                                     |                   | 96fF = 22ppm/K |

### RDC Characteristics

**Figure 14:**  
Resolution RDC Unit

| Measurement Conditions                 | R2/Rref Typ.        | RMS Noise R2/Rref | Typical RMS Noise Temperature <sup>(1)</sup> |
|--|---------------------|-------------------|--|
| No averaging, 2 fake measurements      | 0.899               | 31.7ppm           | 12.2mK                                       |
| 16-fold averaging, 8 fake measurements | 0.897               | 20.2ppm           | 7.8mK  |
| Measurement conditions                 | 10nF @ PTOOUT, 25°C |                   |  |

**Note(s):**

1. After linearization in post-processing software

Typical linearity error with internal AI-thermometer after linearization and conversion into temperature, assuming a linear relation between temperature and resistivity:

- $-20^{\circ}\text{C} < \text{Temp.} < 0^{\circ}\text{C} \rightarrow 290\text{mK}$
- $0^{\circ}\text{C} < \text{Temp.} < 80^{\circ}\text{C} \rightarrow 110\text{mK}$

## Timing Characteristics

Figure 15:  
PCap04 Timing Characteristics

| Symbol      | Parameter                      | Conditions      | Min | Typ | Max | Unit    |
|-------------|--------------------------------|-----------------|-----|-----|-----|---------|
| $t_{START}$ | Start-Up Time                  |                 | 3.9 |     | 4.0 | ms      |
| $t_C$       | CDC Discharge Time             | Measure range 1 | 0   |     | 20  | $\mu$ s |
| $t_R$       | RDC Discharge Time             | Measure range 1 | 0   |     | 20  | $\mu$ s |
| $f_{SPI}$   | SPI Bus Frequency              | Clock frequency | 0   |     | 20  | MHz     |
| $f_{I2C}$   | I <sup>2</sup> C Bus Frequency | Data rate       | 0   | 100 |     | kHz     |



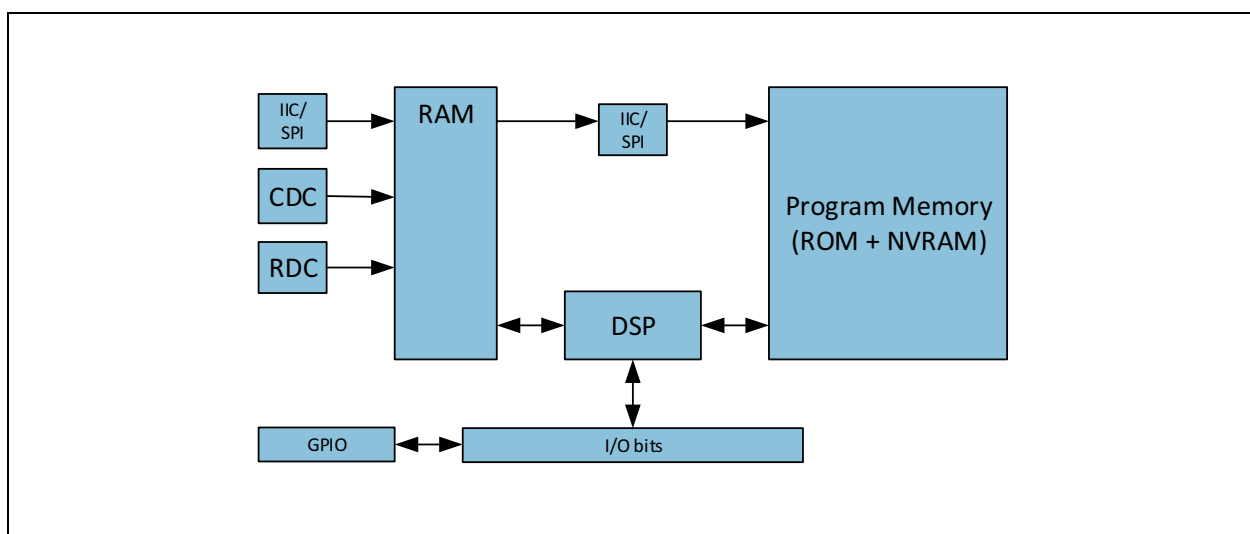
## Detailed Description

PCap04 is an integrated solution for digitizing capacitive and resistive sensors, including a DSP for data processing like linearization and temperature correction. A 6-channel CDC allows to handle grounded and floating sensors in single and differential mode. The capacitance range applicable is from a few pF to hundreds of nanofarads. The RDC unit is mainly intended for measuring temperature, by means of an internal sensor and reference or by means of external resistors like PT1000.

A 32-bit digital signal processor (DSP) in Harvard architecture is integrated to the PCap04. It is responsible for taking the information from the CDC and RDC measuring units, for processing the data and making them available to the user interface. Both, the CDC/RDC raw data as well as the data processed by the DSP are stored in the RAM. The program for the DSP is stored in the NVRAM. The DSP can collect various status information from a set of 64 I/O Bits and write back 16 of those. This way, the DSP can react on and also control the GPIO pins of PCap04. The DSP is internally clocked at approximately 60MHz. The internal clock is stopped through a firmware command, to save power. The DSP starts again upon a GPIO signal or an “end of measurement” condition.

In its simplest form, the DSP transfers the pure time measurement information from the CDC/RDC to the read registers without any further processing. The next higher step is to calculate the capacitance ratios including the information from the compensation measurements, as it is provided in **ams** standard firmware version PCap04\_standard\_v01.hex. Finally, **ams** provides a ready-made linearize firmware that performs a linearization via polynomial of third degree and temperature compensation via polynomial of second degree. Many functional blocks for the linearization firmware are implemented as ROM code. This way, the main firmware can be very compact and can fit into the 1k NVRAM.

**Figure 16:**  
CDC, RDC and DSP Embedding



The content of the read registers will always depend on the firmware in use. With the standard firmware it will be the pure capacitance and resistance ratios. With the linearization firmware it might be the linearized and calibrated result, e.g. a pressure given in Pascal or humidity given in percent.

The DSP is **ams** proprietary to cover low-power tasks as well as very high data rates. It is programmed in assembler. A user-friendly assembler software with a graphical interface, help text pop-ups as well as sample code sustain programming efforts.

## Register Description

### Configuration Registers

The PCap04 offers 48 registers for configuring the hardware (CDC, RDC, clocks, PDM/PWM, DSP). All these 48 registers are of one byte size. Additional four registers are used as special function registers. The 48<sup>th</sup> register contains nothing but one single bit, the RUNBIT, which enables/disables the front-end and the DSP.

All configurations are written simultaneously to registers and to the RAM Part of NVRAM and can be read back.

### Register Overview

Figure 17:  
Register Overview

| Addr                           | Name  | <D7>                | <D6>             | <D5>           | <D4>                | <D3>         | <D2>          | <D1>           | <D0>         |
|--------------------------------|-------|---------------------|------------------|----------------|---------------------|--------------|---------------|----------------|--------------|
| <b>Configuration Registers</b> |       |                     |                  |                |                     |              |               |                |              |
| 0                              | CFG0  | I2C_A               |                  | OLF_FTUNE      |                     |              | OLF_CTUNE     |                |              |
| 1                              | CFG1  | OX_DIS              |                  | OX_DIV4        | OX_AUTOST<br>OP_DIS | OX_STOP      | OX_RUN        |                |              |
| 2                              | CFG2  | RDCHG_INT_SEL1      |                  | RDCHG_INT_SELO |                     | RDCHG_INT_EN |               | RDCHG_EXT_EN   |              |
| 3                              | CFG3  |                     | AUX_PD_DIS       | AUX_CINT       | RDCHG_OPEN          |              | RDCHG_PERM_EN | RDCHG_EXT_PERM | RCHG_SEL     |
| 4                              | CFG4  | C_REF_INT           |                  | C_COMP_EXT     | C_COMP_INT          |              |               | C_DIFFERENTIAL | C_FLOATING   |
| 5                              | CFG5  | CY_PRE_MR1_SHORT    |                  | C_PORT_PAT     |                     | CY_HFCLK_SEL | CY_DIV4_DIS   | CY_PRE_LONG    | C_DC_BALANCE |
| 6                              | CFG6  |                     |                  | C_PORT_EN      |                     |              |               |                |              |
| 7                              | CFG7  | C_AVRG<7:0>         |                  |                |                     |              |               |                |              |
| 8                              | CFG8  |                     |                  |                | C_AVRG<12:8>        |              |               |                |              |
| 9                              | CFG9  | CONV_TIME<7:0>      |                  |                |                     |              |               |                |              |
| 10                             | CFG10 | CONV_TIME<15:8>     |                  |                |                     |              |               |                |              |
| 11                             | CFG11 |                     | CONV_TIME<22:16> |                |                     |              |               |                |              |
| 12                             | CFG12 | DISCHARGE_TIME<7:0> |                  |                |                     |              |               |                |              |

| Addr | Name  | <D7>                  | <D6>          | <D5>           | <D4>           | <D3>             | <D2>          | <D1>                 | <D0>     |  |
|------|-------|-----------------------|---------------|----------------|----------------|------------------|---------------|----------------------|----------|--|
| 13   | CFG13 | C_STARTONPIN          |               |                | C_TRIG_SEL     |                  |               | DISCHARGE_TIME <9:8> |          |  |
| 14   | CFG14 | PRECHARGE_TIME<7:0>   |               |                |                |                  |               |                      |          |  |
| 15   | CFG15 |                       |               | C_FAKE         |                |                  |               | PRECHARGE_TIME <9:8> |          |  |
| 16   | CFG16 | FULLCHARGE_TIME <7:0> |               |                |                |                  |               |                      |          |  |
| 17   | CFG17 |                       | C_REF_SEL     |                |                |                  |               | DISCHARGE_TIME <9:8> |          |  |
| 18   | CFG18 | C_G_OP_RUN            | C_G_OP_EXT    | C_G_EN         |                |                  |               |                      |          |  |
| 19   | CFG19 | C_G_OP_VU             |               | C_G_OP_ATTN    |                | C_G_TIME         |               |                      |          |  |
| 20   | CFG20 | R_CY                  |               |                |                |                  | C_G_OP_TR     |                      |          |  |
| 21   | CFG21 | R_TRIG_PREDIV<7:0>    |               |                |                |                  |               |                      |          |  |
| 22   | CFG22 |                       | R_TRIG_SEL    |                |                | R_AVRG           |               | R_TRIG_PREDIV <9:8>  |          |  |
| 23   | CFG23 | R_PORT_EN             |               | R_PORT_EN_IMES | R_PORT_EN_IREF |                  | R_FAKE        | R_STARTONPIN         |          |  |
| 24   | CFG24 |                       |               | TDC_CHAN_EN    |                | TDC_ALUPER MOPEN | TDC_NOISE_DIS | TDC_MUPU_SPEED       |          |  |
| 25   | CFG25 | TDC_MUPU_NO           |               |                |                |                  |               |                      |          |  |
| 26   | CFG26 | TDC_QHA_SEL           |               |                |                |                  |               | TDC_NOISE_CY_DIS     |          |  |
| 27   | CFG27 | DSP_MOFLO_EN          |               |                |                | DSP_SPEED        |               | PG1x PG3             | PG0x PG2 |  |
| 28   | CFG28 | WD_DIS                |               |                |                |                  |               |                      |          |  |
| 29   | CFG29 | DSP_STARTONPIN        |               |                |                | DSP_FF_IN        |               |                      |          |  |
| 30   | CFG30 | PG5_INTN_EN           | PG4_INTN_EN   |                |                | DSP_START_EN     |               |                      |          |  |
| 31   | CFG31 | PI1_TOGGLE_EN         | PI0_TOGGLE_EN | PI0_RES        |                | PI0_PDM_SEL      | PI0_CLK_SEL   |                      |          |  |
| 32   | CFG32 |                       |               | PI1_RES        |                | PI1_PDM_SEL      | PI1_CLK_SEL   |                      |          |  |
| 33   | CFG33 | PG_DIR_IN             |               |                |                | PG_PU            |               |                      |          |  |
| 34   | CFG34 | INT_TRIG_BG           | DSP_TRIG_BG   | BG_PERM        | AUTO START     |                  |               |                      |          |  |

| Addr | Name  | <D7>                | <D6>               | <D5> | <D4>               | <D3>                | <D2>                | <D1>                | <D0>                |  |
|------|-------|---------------------|--------------------|------|--------------------|---------------------|---------------------|---------------------|---------------------|--|
| 35   | CFG35 | CDC_GAIN_CORR<7:0>  |                    |      |                    |                     |                     |                     |                     |  |
| 36   | CFG36 | -                   |                    |      |                    |                     |                     |                     |                     |  |
| 37   | CFG37 | -                   |                    |      |                    |                     |                     |                     |                     |  |
| 38   | CFG38 | BG_TIME             |                    |      |                    |                     |                     |                     |                     |  |
| 39   | CFG39 | PULSE_SEL1          |                    |      |                    | PULSE_SELO          |                     |                     |                     |  |
| 40   | CFG40 | C_SENSE_SEL         |                    |      |                    |                     |                     |                     |                     |  |
| 41   | CFG41 | R_SENSE_SEL         |                    |      |                    |                     |                     |                     |                     |  |
| 42   | CFG42 |                     | ALARM1<br>_SEL ECT |      | ALARM0<br>_SEL ECT | EN_<br>ASYNC_<br>RD | HS_<br>MODE_<br>SEL | R_<br>MEDIAN_<br>EN | C_<br>MEDIAN_<br>EN |  |
| ...  |       |                     |                    |      |                    |                     |                     |                     |                     |  |
| 47   | CFG47 |                     |                    |      |                    |                     |                     |                     | RUNBIT              |  |
| 48   | CFG48 |                     |                    |      |                    | MEM_LOCK            |                     |                     |                     |  |
| 49   | CFG49 | SERIAL_NUMBER<7:0>  |                    |      |                    |                     |                     |                     |                     |  |
| 50   | CFG50 | SERIAL_NUMBER<15:8> |                    |      |                    |                     |                     |                     |                     |  |
| ...  |       |                     |                    |      |                    |                     |                     |                     |                     |  |
| 54   | CFG54 | MEM_CTRL            |                    |      |                    |                     |                     |                     |                     |  |
| ...  |       |                     |                    |      |                    |                     |                     |                     |                     |  |
| 62   | CFG62 | CHARGE_PUMP<7:0>    |                    |      |                    |                     |                     |                     |                     |  |
| 63   | CFG63 | CHARGE_PUMP<15:8>   |                    |      |                    |                     |                     |                     |                     |  |

| Addr                  | Name | <D7> | <D6> | <D5> | <D4> | <D3> | <D2> | <D1> | <D0> |
|-----------------------|------|------|------|------|------|------|------|------|------|
| <b>Read Registers</b> |      |      |      |      |      |      |      |      |      |
| 0                     | RES0 | 7    |      |      |      |      |      |      | 0    |
| 1                     |      | 15   |      |      |      |      |      |      | 8    |
| 2                     |      | 23   |      |      |      |      |      |      | 16   |
| 3                     |      | 31   |      |      |      |      |      |      | 24   |
| 4                     | RES1 | 7    |      |      |      |      |      |      | 0    |
| 5                     |      | 15   |      |      |      |      |      |      | 8    |
| 6                     |      | 23   |      |      |      |      |      |      | 16   |
| 7                     |      | 31   |      |      |      |      |      |      | 24   |
| 8                     | RES2 | 7    |      |      |      |      |      |      | 0    |
| 9                     |      | 15   |      |      |      |      |      |      | 8    |
| 10                    |      | 23   |      |      |      |      |      |      | 16   |
| 11                    |      | 31   |      |      |      |      |      |      | 24   |
| 12                    | RES3 | 7    |      |      |      |      |      |      | 0    |
| 13                    |      | 15   |      |      |      |      |      |      | 8    |
| 14                    |      | 23   |      |      |      |      |      |      | 16   |
| 15                    |      | 31   |      |      |      |      |      |      | 24   |
| 16                    | RES4 | 7    |      |      |      |      |      |      | 0    |
| 17                    |      | 15   |      |      |      |      |      |      | 8    |
| 18                    |      | 23   |      |      |      |      |      |      | 16   |
| 19                    |      | 31   |      |      |      |      |      |      | 24   |
| 20                    | RES5 | 7    |      |      |      |      |      |      | 0    |
| 21                    |      | 15   |      |      |      |      |      |      | 8    |
| 22                    |      | 23   |      |      |      |      |      |      | 16   |
| 23                    |      | 31   |      |      |      |      |      |      | 24   |
| 24                    | RES6 | 7    |      |      |      |      |      |      | 0    |
| 25                    |      | 15   |      |      |      |      |      |      | 8    |
| 26                    |      | 23   |      |      |      |      |      |      | 16   |
| 27                    |      | 31   |      |      |      |      |      |      | 24   |

| Addr | Name     | <D7>         | <D6>            | <D5>         | <D4>        | <D3>        | <D2>        | <D1>        | <D0>        |
|------|----------|--------------|-----------------|--------------|-------------|-------------|-------------|-------------|-------------|
| 28   | RES7     | 7            |                 |              |             |             |             |             | 0           |
| 29   |          | 15           |                 |              |             |             |             |             | 8           |
| 30   |          | 23           |                 |              |             |             |             |             | 16          |
| 31   |          | 31           |                 |              |             |             |             |             | 24          |
| 32   | STATUS_0 | POR_FLAG_WDG | POR_FLAG_CONFIG | IR_FLAG_COLL | AUTOBOOT    |             | RDC_READY   | CDC_ACTIVE  | RUNBIT      |
| 33   | STATUS_1 |              |                 |              |             | RDC_ERR     | MUP_ERR     | ERR_OVFL    | COMB_ERR    |
| 34   | STATUS_2 |              | C_PORTERR_INT   | C_PORTERR_5  | C_PORTERR_4 | C_PORTERR_3 | C_PORTERR_2 | C_PORTERR_1 | C_PORTERR_0 |

**Note(s):**

1. Register 35 to 42 depend on the firmware. The values given are true for the standard firmware.
2. The content of the results registers depend on the firmware.

## Detailed Configuration Register Description

### Configuration Register 0 (Address 0x0)

Figure 18:  
Register 0

| Bit | Bit Name  | Settings   | Bit Description                            |
|-----|-----------|--|--|
| 7:6 | I2C_A     | 0 to 3   | Complement to the I <sup>2</sup> C-address |
| 5:2 | OLF_FTUNE | 0 : Minimum<br>7 : Typ., recommended<br>15 : Maximum | Fine-tune the low-frequency clock          |
| 1:0 | OLF_CTUNE | 0 : 10kHz<br>1 : 50kHz<br>2 : 100kHz<br>3 : 200kHz   | Coarse-tune the low-frequency clock        |

### Configuration Register 1 (Address 0x1)

Figure 19:  
Register 1

| Bit | Bit Name        | Settings   | Bit Description  |
|-----|-----------------|--|--|
| 7   | OX_DIS          | Default : 0  | Disable the OX clock   |
| 5   | OX_DIV4         | 0 : No division;<br>f <sub>ox</sub> = 2MHz<br>1 : Division by 4;<br>f <sub>ox</sub> = 0.5MHz   | OX clock frequency : Raw freq./ 4  |
| 4   | OX_AUTOSTOP_DIS | Default : 0  | ams internal bits  |
| 3   | OX_STOP         | Default : 0  | ams internal bits  |
| 2:0 | OX_RUN          | 0 : Generator off<br>6 : OX latency = 1 / fOLF<br>3 : OX latency = 2 / fOLF<br>2 : OX latency = 31 / fOLF<br>1 : OX runs permanently | Control the permanency or the latency of the OX generator. Latency means an oscillator settling time before a measurement starts |



**Configuration Register 2 (Address 0x2)****Figure 20:  
Register 2**

| Bit | Bit Name       | Settings  | Bit Description   |
|-----|----------------|---|---|
| 7:6 | RDCHG_INT_SEL1 | 0 : 180kΩ<br>1 : 90kΩ<br>2 : 30kΩ (default)<br>3 : 10kΩ | Same, but for ports PC4 – PC5   |
| 5:4 | RDCHG_INT_SEL0 |   | Choice of one out of 4 on-chip discharge resistors for the CDC ports PC0 – PC3 plus internal port PC6   |
| 3   | RDCHG_INT_EN   | 0 : Off<br>1 : Internal on (default)                    | Enable internal discharge resistors   |
| 1   | RDCHG_EXT_EN   | 0 : Off (default)<br>1 : External on                    | Enable external discharge resistor switching on PCAUX during discharge phase (High-Z during pre- and full-charge phase)<br>Note: AUX_PD_DIS (PCAUX pull down disable) : 1 |

**Configuration Register 3 (Address 0x3)****Figure 21:  
Register 3**

| Bit | Bit Name       | Settings   | Bit Description   |
|-----|----------------|--|---|
| 6   | AUX_PD_DIS     | 0 : Pull-down active<br>1 : Pull-down disabled           | Disable pull-down at PCAUX  |
| 5   | AUX_CINT       | 0 : Normal (default)<br>1 : Aux during c-internal active | Activates auxiliary Port PCAUX during internal c-reference conversion only  |
| 4:3 | RDCHG_OPEN     | 2 : Recommended  | <b>ams</b> internal bits  |
| 2   | RDCHG_PERM_EN  | 0 : Off (default)<br>1 : On                              | Keep the chip-internal discharge resistor permanently connected.  |
| 1   | RDCHG_EXT_PERM | 0 : Off (default)<br>1 : On                              | Activates auxiliary Port PCAUX permanently for a) permanently discharge or b) to add an offset capacitance to every charge/discharge cycle<br><b>Note:</b> AUX_PD_DIS (PCAUX pull down disable) : 1 |
| 0   | RCHG_SEL       | 0 : 180kΩ<br>1 : 10kΩ (default)                          | Choice of one out of 2 on-chip charging resistors for the CDC, permitting to limit the charging current, avoiding transients  |

**Configuration Register 4 (Address 0x4)**

**Figure 22:  
Register 4**

| Bit | Bit Name       | Settings   | Bit Description  |
|-----|----------------|--|--|
| 7   | C_REF_INT      | 0: External reference at PC0/GND or PC0/PC1<br>1: Internal reference | Use on-chip reference capacitor at CDC special ports PC6                                     |
| 5   | C_COMP_EXT     | 0: Idle<br>1: Active; must be avoided when C_FLOATING==0             | Activate the compensation mechanism for off-chip parasitic capacitances                      |
| 4   | C_COMP_INT     | 0: Idle<br>1: Active   | Activate the compensation mechanism for on-chip parasitic capacitances and gain compensation |
| 1   | C_DIFFERENTIAL | 0: Ordinary<br>1: Differential                                       | Select between single or differential sensors  |
| 0   | C_FLOATING     | 0: Grounded<br>1: Floating   | Select between grounded or floating sensors  |

**Configuration Register 5 (Address 0x5)**

**Figure 23:  
Register 5**

| Bit | Bit Name         | Settings                                       | Bit Description  |
|-----|------------------|--|--|
| 7   | CY_PRE_MR1_SHORT | 0: Normal (recommended)<br>1: Reduced          | Reduce delays between internal clock paths   |
| 5   | C_PORT_PAT       | 0: Normal<br>1: Alternating order of ports     | The order of the measured ports will be reversed after each sequence. If C_PORT_PAT is activated then C_AVRG + C_FAKE should be an even number |
| 3   | CY_HFCLK_SEL     | 0: OLF<br>1: OHF                               | Clock source for the CDC   |
| 2   | CY_DIV4_DIS      | 0: Off<br>1: On                                | Quadruple the clock period (only in combination with CY_HFCLK_SEL == 1)  |
| 1   | CY_PRE_LONG      | 0: Off, recommended<br>1: On                   | Adds safety delay between internal clock paths   |
| 0   | C_DC_BALANCE     | 0: Off ("single HiZ)<br>1: DC free ("both HiZ) | Only for differential floating mode (other modi are DC free), changes port control to eliminate DC at Capacity sense                           |

**Note(s):**

1. CY\_HFCLK\_SEL and CY\_DIV4\_DIS are combined in the evaluation software as 'Cycle Clock Select'

### Configuration Register 6 (Address 0x6)

Figure 24:  
Register 6

| Bit | Bit Name  | Settings  | Bit Description   |
|-----|-----------|---|---|
| 5:0 | C_PORT_EN | 0x00 : All off, the CDC will not work<br>0x01 : Only port PC0 is activated etc.<br>0x3F : All ports activated | Enables bitwise the CDC ports from PC0 to PC5, bit #0 for port PC0, #1 for PC1 etc. |

### Configuration Register 7:8 (Address 0x7, 0x8)

Figure 25:  
Register 7:8

| Bit  | Bit Name | Settings  | Bit Description  |
|------|----------|---|--|
| 12:0 | C_AVRG   | 0, 1 : Sample size = 1<br>2 : Sample size = 2<br>3 : Sample size = 3<br>...<br>8191 : Maximum sample size | Sample size for averaging (calculating the mean value) over CDC measurements |

### Configuration Register 11:9 (Address 0x9;0xA;0xB)

Figure 26:  
Register 11:9

| Bit  | Bit Name  | Settings  | Bit Description   |
|------|-----------|---|---|
| 22:0 | CONV_TIME | Concerning CDC, a particular period for triggering the measurements<br>$T_{\text{conv./seq}} = 2 * \text{CONV\_TIME[.]} / f_{\text{OLF}}$ | Conversion trigger period or: sequence period (in stretched mode) |

**Configuration Register 12 (Address 0xC)**

**Figure 27:  
Register 12**

| Bit                        | Bit Name       | Settings   | Bit Description  |
|----------------------------|----------------|--|--|
| 7:0<br>&<br>Reg.13:<br>1:0 | DISCHARGE_TIME | OLF:<br>$T_{discharge} = (DISCHARGE\_TIME + 1) * T_{cycleclock}$<br>OHF:<br>$T_{discharge} = (DISCHARGE\_TIME + 0) * T_{cycleclock}$<br>1023 : Off | Sets CDC discharge time $T_{discharge}$ . Time interval reserved for discharge time measurement. |

**Configuration Register 13 (Address 0xD)**

**Figure 28:  
Register 13**

| Bit                       | Bit Name       | Settings   | Bit Description  |
|---------------------------|----------------|--|--|
| 7:6                       | C_STARTONPIN   | 0 : PG0, 1 : PG1,<br>2 : PG2, 3 : PG3  | Selection of the GPIO port that permits triggering a CDC start |
| 4:2                       | C_TRIG_SEL     | 0 : Continuous<br>1 : Read triggered<br>2 : Timer triggered<br>3 : Timer triggered (stretched)<br>4 : n.d.<br>5 : Pin triggered<br>6 : Opcode triggered<br>(7 : continuous_exp, not recommended) | CDC Trigger Mode   |
| 1:0<br>&<br>Reg12:<br>7:0 | DISCHARGE_TIME | See <a href="#">Register 12</a>  | See <a href="#">Register 12</a>                                |

**Configuration Register 14 (Address 0xE)**

**Figure 29:  
Register 14**

| Bit                        | Bit Name       | Settings  | Bit Description  |
|----------------------------|----------------|---|--|
| 7:0<br>&<br>Reg.15:<br>1:0 | PRECHARGE_TIME | OLF:<br>$T_{precharge} = (PRECHARGE\_TIME + 1) * T_{cycleclock}$<br>OHF & FULLCHARGE_TIME = 1023:<br>$T_{precharge} = (PRECHARGE\_TIME + 2) * T_{cycleclock}$<br>OLF & FULLCHARGE_TIME != 0x3FF:<br>$T_{precharge} = (PRECHARGE\_TIME + 1) * T_{cycleclock}$<br>1023: Off | Sets CDC discharge time $T_{precharge}$ . Time interval reserved for discharge time measurement. |

### Configuration Register 15 (Address 0xF)

Figure 30:  
Register 15

| Bit                       | Bit Name       | Settings                                      | Bit Description   |
|---------------------------|----------------|---|---|
| 5:2                       | C_FAKE         | 0 : None<br>1 : 1 fake<br>...<br>15: 15 fakes | Number of “fake” or “warm-up” measurements for the CDC, performed just before the “real” ones; the “fake” values do not count |
| 1:0<br>&<br>Reg14:<br>7:0 | PRECHARGE_TIME | See <a href="#">Register 14</a>               | See <a href="#">Register 14</a>   |

### Configuration Register 16 (Address 0x10)

Figure 31:  
Register 16

| Bit                        | Bit Name        | Settings   | Bit Description  |
|----------------------------|-----------------|--|--|
| 7:0<br>&<br>Reg.17:<br>1:0 | FULLCHARGE_TIME | OLF:<br>$T_{fullcharge} = (FULLCHARGE\_TIME + 1) * T_{cycleclock}$<br>OHF:<br>$T_{fullcharge} = (FULLCHARGE\_TIME + 2) * T_{cycleclock}$ | Sets CDC discharge time $t_{fullcharge}$ .<br>Time interval reserved for discharge time measurement. |

### Configuration Register 17 (Address 0x11)

Figure 32:  
Register 17

| Bit                        | Bit Name        | Settings   | Bit Description   |
|----------------------------|-----------------|--|---|
| 1:0<br>&<br>Reg.16:<br>7:0 | FULLCHARGE_TIME | See <a href="#">Register 16</a>  | See <a href="#">Register 16</a>   |
| 6:2                        | C_REF_SEL       | 0 : Minimum<br>1 : Approx.1pF<br>...<br>31 : Maximum (approx.<br>31pF) | Setting the on-chip reference capacitor for the CDC<br><b>Note:</b> Step width varies from 0.3pF to 1.5pF |

**Configuration Register 18 (Address 0x12)**

**Figure 33:  
Register 18**

| Bit | Bit Name   | Settings                    | Bit Description  |
|-----|------------|-----------------------------|--|
| 7   | C_G_OP_RUN | Guard: OP Mode              | 0 : Permanent<br>1 : Pulsed (set OP to sleep mode between conversions)   |
| 6   | C_G_OP_EXT | Guard: Activate external OP | 0 : Internal OP<br>1 : External OP, PG3 as C_G_MUX_SEL   |
| 5:0 | C_G_EN     | Guard Enable, for each port | b'xxxxx1 : Activates port PC0<br>b'xxxx1x : Activates port PC1<br>b'xxx1xx : Activates port PC2<br>b'xx1xxx : Activates port PC3<br>b'x1xxxx : Activates port PC4<br>b'1xxxxx : Activates port PC5 |

### Configuration Register 19 (Address 0x13)

Figure 34:  
Register 19

| Bit | Bit Name    | Settings  | Bit Description                                      |
|-----|-------------|---|--|
| 7:6 | C_G_OP_VU   | Guard: OP gain (from Sense Port to Guard)                                       | 0 : x 1.00<br>1 : x 1.01<br>2 : x 1.02<br>3 : x 1.03 |
| 5:4 | C_G_OP_ATTN | Guard: OP attenuation   | 0 : 0.5aF<br>1 : 1.0aF<br>2 : 1.5aF<br>3 : 2.0aF     |
| 3:0 | C_G_TIME    | Guard: Time during Precharge to switch Guard Port from "direct connected" to OP | t : C_G_TIME * 500ns                                 |

### Configuration Register 20 (Address 0x14)

Figure 35:  
Register 20

| Bit | Bit Name  | Settings   | Bit Description               |        |        |
|-----|-----------|--|-------------------------------|--------|--------|
|     |           |  | OLF f                         | R_CY=0 | R_CY=1 |
| 7   | R_CY      | Cycle-time for the RDC Precharge/Charge/ Discharge, depending on OLF frequency | 10kHz                         | 100µs  | 200µs  |
|     |           |  | 50kHz                         | 20µs   | 40µs   |
|     |           |  | 100kHz                        | 10µs   | 20µs   |
|     |           |  | 200kHz                        | 20µs   | 40µs   |
| 2:0 | C_G_OP_TR | Guard OP current trim  | 0 :<br>...<br>7 : Recommended |        |        |

### Configuration Register 21 (Address 0x15)

Figure 36:  
Register 21

| Bit                       | Bit Name      | Settings   | Bit Description  |
|---------------------------|---------------|--|--|
| 7:0<br>&<br>Reg22:<br>1:0 | R_TRIG_PREDIV | 0, 1 : Every signal triggers<br>2 : Every 2 <sup>nd</sup> signal triggers<br>3 : Every 3 <sup>rd</sup> signal triggers<br>...<br>1023 : Maximum factor | Pre-divider, permits to make less temperature measurements than capacitance measurements. This is a factor between measurement rates of CDC over RDC. It is used also as OLF clock divider if OLF is used as trigger source. |

### Configuration Register 22(Address 0x16)

Figure 37:  
Register 22

| Bit                       | Bit Name      | Settings  | Bit Description  |
|---------------------------|---------------|---|--|
| 6:4                       | R_TRIG_SEL    | 0 : Off<br>1 : Timer triggered<br>3 : Pin triggered<br>5 : CDC asynchronous (recommended)<br>6 : CDC synchronous                                      | Trigger source selection for the RDC<br>5 & 6: triggered by the end of CDC conversion  |
| 3:2                       | R_AVRG        | 0 : Not averaged<br>1 : 4-fold averaged<br>2 : 8-fold averaged<br>3 : 16-fold averaged  | Sample size for the mean value calculation (averaging) in the RDC part   |
| 1:0<br>&<br>Reg21:<br>7:0 | R_TRIG_PREDIV | 0, 1 : Every signal trigger<br>2 : Every 2 <sup>nd</sup> signal triggers<br>3 : Every 3 <sup>rd</sup> signal triggers<br>...<br>1023 : Maximum factor | Pre-divider, permits to make less temperature measurements than capacitance measurements. This is a factor between measurement rates of CDC over RDC. It is used also as OLF clock divider if OLF is used as trigger source. |



**Configuration Register 23 (Address 0x17)****Figure 38:  
Register 23**

| Bit | Bit Name       | Settings  | Bit Description   |
|-----|----------------|---|---|
| 7:6 | R_PORT_EN      | 'bx0 : Disabled<br>'bx1 : Activates port PT0REF<br>'b0x : Disabled<br>'b1x : Activates port PT1 | Port activation for the RDC part  |
| 5   | R_PORT_EN_IMES | 0 : Disabled<br>1 : Enabled   | Port activation for internal aluminum temperature sensor  |
| 4   | R_PORT_EN_IREF | 0 : Disabled<br>1 : Enabled   | Port activation for internal reference resistor   |
| 3   | -              | -   | -   |
| 2   | R_FAKE         | 0 : 2 fake cycles per average value<br>1 : 8 fake cycle per average value                       | Number of "fake" or "warm-up" measurements for the RDC, performed just before the "real" ones; the "fake" values do not count |
| 1:0 | R_STARTONPIN   | 0 : PG0<br>1 : PG1,<br>2 : PG2<br>3 : PG3   | Selection of the GPIO port that permits triggering a RDC start  |

**Configuration Register 24 (Address 0x18)****Figure 39:  
Register 24**

| Bit | Bit Name        | Settings      | Bit Description          |
|-----|-----------------|---------------|--------------------------|
| 7:6 | -               | -             | -                        |
| 5:4 | TDC_CHAN_EN     | Mandatory : 3 | <b>ams</b> internal bits |
| 3   | TDC_ALUPERMOPEN | Mandatory : 0 | <b>ams</b> internal bits |
| 2   | TDC_NOISE_DIS   | Mandatory : 0 | <b>ams</b> internal bits |
| 1:0 | TDC_MUPU_SPEED  | Mandatory : 3 | <b>ams</b> internal bits |

### Configuration Register 25 (Address 0x19)

Figure 40:  
Register 25

| Bit | Bit Name    | Settings      | Bit Description   |
|-----|-------------|---------------|-------------------|
| 7:2 | TDC_MUPU_NO | Mandatory : 1 | ams internal bits |
| 1:0 | -           | -             | -                 |

### Configuration Register 26 (Address 0x1A)

Figure 41:  
Register 26

| Bit | Bit Name         | Settings       | Bit Description   |
|-----|------------------|----------------|-------------------|
| 7:2 | TDC_QHA_SEL      | Mandatory : 20 | ams internal bits |
| 1   | TDC_NOISE_CY_DIS | Mandatory : 1  | ams internal bits |
| 0   | -                | -              | -                 |

### Configuration Register 27 (Address 0x1B)

Figure 42:  
Register 27

| Bit | Bit Name     | Settings  | Bit Description   |
|-----|--------------|---|---|
| 7:6 | DSP_MOFLO_EN | 0 : Off<br>3 : On   | Enable the mono-flop (anti-bouncing filter) in the GPIO pulse line. |
| 5:4 | -            | -   | -   |
| 3:2 | DSP_SPEED    | 0 : Fastest<br>1 : Fast<br>2 : Slow, recommended<br>3 : Slowest | DSP speed   |
| 1   | PG1xPG3      | 0 : Pulse output at PG3<br>1 : Pulse output at PG1              | Switch PG1/PG3 wiring to/from DSP                                   |
| 0   | PG0xPG2      | 0 : Pulse output at PG2<br>1 : Pulse output at PG0              | Switch PG0/PG2 wiring to/from DSP                                   |

### Configuration Register 28 (Address 0x1C)

Figure 43:  
Register 28

| Bit | Bit Name | Settings  | Bit Description  |
|-----|----------|---|--|
| 7:0 | WD_DIS   | 0x5A : Watchdog disabled (off)<br>0x00 (recommended) /<br>else : Watchdog enabled | Watchdog Disable, to disable Watchdog 0x5A has to be written to this register. The watchdog period is between 9s and 15s |

### Configuration Register 29 (Address 0x1D)

Figure 44:  
Register 29

| Bit | Bit Name       | Settings                     | Bit Description  |
|-----|----------------|------------------------------|--|
| 7:4 | DSP_STARTONPIN | Bitwise PG0 to PG3           | Pin mask for starting the DSP - This mask permits assigning one or more GPIO pins to start the DSP |
| 3:0 | DSP_FF_IN      | Bitwise DSP_IN_0 to DSP_IN_3 | Pin mask for flip-flop activation  |

### Configuration Register 30 (Address 0x1E)

Figure 45:  
Register 30

| Bit | Bit Name     | Settings  | Bit Description          |
|-----|--------------|---|--------------------------|
| 7   | PG5_INTN_EN  | 0 : PG5 normal operation<br>1 : PG5 <== INTN  | Route INTN Signal to PG5 |
| 6   | PG4_INTN_EN  | 0 : PG4 normal operation<br>1 : PG4 <== INTN  | Route INTN Signal to PG4 |
| 5:3 | -            | -   | -                        |
| 2:0 | DSP_START_EN | 'bxxx1 : Trigger by end of CDC<br>'bxx1x : Trigger by end of RDC (recommended)<br>'bx1xx : Trigger by timer | DSP Trigger Enable       |

**Configuration Register 31 (Address 0x1F)**
**Figure 46:**  
**Register x**

| Bit | Bit Name      | Settings   | Bit Description   |
|-----|---------------|--|---|
| 7   | PI1_TOGGLE_EN | 0 : Normal operation<br>1 : Toggle flip flop active  | Activates toggle flip flop at Pulse Interface 1 Output especially for PDM to create 1:1 duty factor |
| 6   | PI0_TOGGLE_EN | 0 : Normal operation<br>1 : Toggle flip flop active  | Activates toggle flip flop at Pulse Interface 0 Output especially for PDM to create 1:1 duty factor |
| 5:4 | PI0_RES       | 0 : 10 bit<br>1 : 12 bit<br>2 : 14 bit<br>3 : 16 bit   | Resolution of the pulse-code interfaces   |
| 3   | PI0_PDM_SEL   | 0 : PWM<br>1 : PDM   | Pulse Interface 0<br>PWM / PDM switch   |
| 2:0 | PI0_CLK_SEL   | 0 : Off<br>1 : OLF / 1<br>2 : OLF / 2<br>3 : OLF / 4<br>4 : OX / 1<br>5 : OX / 2<br>6 : OX / 4<br>7 : n.d. | Pulse Interface 0<br>Clock Select   |

### Configuration Register 32 (Address 0x20)

Figure 47:  
Register 32

| Bit | Bit Name    | Settings   | Bit Description                         |
|-----|-------------|--|---|
| 7:6 | -           | -  | -                                       |
| 5:4 | PI1_RES     | 0 : Off<br>1 : OLF / 1<br>2 : OLF / 2<br>3 : OLF / 4<br>4 : OX / 1<br>5 : OX / 2<br>6 : OX / 4<br>7 : n.d. | Pulse Interface 1<br>Clock Select       |
| 3   | PI1_PDM_SEL | 0 : PWM<br>1 : PDM   | Resolution of the pulse-code interfaces |
| 2:0 | PI1_CLK_SEL | 0 : 10 bit<br>1 : 12 bit<br>2 : 14 bit<br>3 : 16 bit   | Pulse Interface 1<br>PWM / PDM switch   |

### Configuration Register 33 (Address 0x21)

Figure 48:  
Register 33

| Bit | Bit Name  | Settings                                   | Bit Description   |
|-----|-----------|--|---|
| 7:4 | PG_DIR_IN | 0 : Output<br>1 : Input                    | Toggles general-purpose port direction between input and output<br>#4: PG0<br>#5: PG1<br>#6: PG2<br>#7: PG3 |
| 3:0 | PG_PU     | 0 : Pull-up disabled<br>1 : Pull-up active | Activates protective pull-up resistors at general-purpose ports<br>#0: PG0<br>#1: PG1<br>#2: PG2<br>#3: PG3 |

### Configuration Register 34 (Address 0x22)

Figure 49:  
Register 34

| Bit | Bit Name    | Settings  | Bit Description  |
|-----|-------------|---|--|
| 7   | INT_TRIG_BG | 0 : Disabled<br>1 : Enabled                         | End of Reading triggers Bandgap  |
| 6   | DSP_TRIG_BG | 0 : Disabled<br>1 : Enabled                         | Bandgap refresh is triggered by the DSP bit setting  |
| 5   | BG_PERM     | 1 : Bandgap permanent enabled<br>0 : Bandgap pulsed | Activate Bandgap permanently. With BG_PERM = 1 the current consumption rises by approx. 20µA |
| 4   | AUTOSTART   | 0 : Disabled<br>1 : CDC trigger after Power On      | For standalone operation, triggers CDC after Power On  |
| 3:0 | -           | Mandatory : 7                                       | ams internal bit   |

### Configuration Register 35 (Address 0x23)

Figure 50:  
Register 35

| Bit | Bit Name           | Settings                     | Bit Description  |
|-----|--------------------|------------------------------|--|
| 7:0 | CDC_GAIN_CORR[7:0] | Recommended 1.25 ==><br>0x40 | Firmware defined configuration of the gain correction factor.<br>Bits 0 to 7 of 8fpp<br>0 : 0<br>n : 1 + n/256 |

### Configuration Register 36 (Address 0x24)

Figure 51:  
Register 36

| Bit | Bit Name | Settings | Bit Description |
|-----|----------|----------|-----------------|
| 7:0 | -        |          | Not used        |

**Configuration Register 37 (Address 0x25)****Figure 52:  
Register 37**

| Bit | Bit Name | Settings | Bit Description |
|-----|----------|----------|-----------------|
| 7:0 | -        |          | Not used        |

**Configuration Register 38 (Address 0x26)****Figure 53:  
Register 38**

| Bit | Bit Name | Settings       | Bit Description  |
|-----|----------|----------------|------------------|
| 7:0 | BG_TIME  | 0: Recommended | Firmware defined |

**Configuration Register 39 (Address 0x27)****Figure 54:  
Register 39**

| Bit | Bit Name   | Settings   | Bit Description                                |
|-----|------------|--|--|
| 7:4 | PULSE_SEL1 | 0 to 5 : Res0 to Res5 (C0..5 /Cref @PCap04_standard)<br>6 : Res6 (PT1/Ref @PCap04_standard)<br>7 : Res7 (Alu/Ref @PCap04_standard) | Firmware defined, select source for Pulse IF 1 |
| 3:0 | PULSE_SELO | 0 to 5 : Res0 to Res5 (C0..5 /Cref @PCap04_standard)<br>6 : Res6 (PT1/Ref @PCap04_standard)<br>7 : Res7 (Alu/Ref @PCap04_standard) | Firmware defined, select source for Pulse IF 0 |

**Configuration Register 40 (Address 0x28)****Figure 55:  
Register 40**

| Bit | Bit Name    | Settings   | Bit Description  |
|-----|-------------|--|------------------|
| 7:0 | C_SENSE_SEL | PCap04_linearize firmware only,<br>select C ratio for linearization<br>0..5 : C0 to 5 / Cref | Firmware defined |

### Configuration Register 41 (Address 0x29)

Figure 56:  
Register 41

| Bit | Bit Name    | Settings   | Bit Description  |
|-----|-------------|--|------------------|
| 7:0 | R_SENSE_SEL | PCap04_linearize firmware only, select R ratio for temperature determination | Firmware defined |

### Configuration Register 42 (Address 0x30)

Figure 57:  
Register 42

| Bit | Bit Name      | Settings  | Bit Description  |
|-----|---------------|---|--|
| 7   | -             | PCap04_linearize firmware only<br>Polarity<br>Select 0 : Z; 1 : Theta | Firmware defined. Select source of alarm signals at PG0 and PG1<br>Active High               |
| 6   | ALARM1_SELECT |   |  |
| 5   | -             |   |  |
| 4   | ALARM0_SELECT |   |  |
| 3   | EN_ASYNC_READ | 1 : Active  | Values in result registers Res0 to Res7 are only updated if the previous value has been read |
| 2   | HS_MODE_SEL   | 0 : Mandatory   | <b>ams</b> internal bit  |
| 1   | R_MEDIAN_EN   | PCap04_linearize firmware only  | Enable median filters for ci/ri in linearize firmware  |
| 0   | C_MEDIAN_EN   |   |  |

### Configuration Register 47 (Address 0x2F)

Figure 58:  
Register 47

| Bit | Bit Name | Settings  | Bit Description  |
|-----|----------|---|--|
| 7:1 | Not used |   |  |
| 0   | RUNBIT   | 0 : Off = the chip system is idle and protected<br>1 : On = the protection is removed, and the system may run | On/off switch for front-end and DSP: It should be "off" during programming and any registry modification, thus protecting the chip from any undesirable/unspecified states |



### Configuration Register 48 (Address 0x30)

Figure 59:  
Register 48

| Bit | Bit Name | Settings  | Bit Description  |
|-----|----------|---|--|
| 7:4 | Not used |   |  |
| 3:0 | MEM_LOCK | 'bxxx1 : NVRAM range `d0 to 703<br>'bxx1x : NVRAM range `d704 to 831<br>'bx1xx : NVRAM range `d832 to 959<br>'b1xxx : NVRAM range `d960 to 1007 and<br>NVRAM range `d1022 to 1023 | Data secure function to safe parts of NVRAM from reading and writing via SIF |

### Configuration Register 49 (Address 0x31)

Figure 60:  
Register 49

| Bit | Bit Name           | Settings   | Bit Description                       |
|-----|--------------------|--|---------------------------------------|
| 7:0 | SERIAL_NUMBER[7:0] | Free disposal for customer. Could only be written as far as the byte is zero. Afterwards it could just be cleared by an complete Erase | Lower byte reserved for serial number |

### Configuration Register 50 (Address 0x32)

Figure 61:  
Register 50

| Bit | Bit Name            | Settings   | Bit Description                        |
|-----|---------------------|--|--|
| 7:0 | SERIAL_NUMBER[15:8] | Free disposal for customer. Could only be written as far as the byte is zero. Afterwards it could just be cleared by an complete Erase | Higher byte reserved for serial number |

**Configuration Register 51 to 53: ams internal Registers, 0x00 mandatory**

**Configuration Register 54 (Address 0x36)**

Figure 62:  
Register 54

| Bit | Bit Name | Settings  | Bit Description |
|-----|----------|---|-----------------|
| 7:0 | MEM_CTRL | 0x2d : NVRAM store enable<br>0x59 : NVRAM recall enable<br>0xb8 : NVRAM erase<br>Register is reset automatically after following SIF activity | Memory control  |

**Configuration Register 55 to 61: ams internal Registers, 0x00 mandatory**

**Configuration Register 62 (Address 0x3e)**

Figure 63:  
Register 62

| Bit | Bit Name         | Settings   | Bit Description                      |
|-----|------------------|--|--------------------------------------|
| 7:0 | CHARGE_PUMP[7:0] | Individual, device-specific setting. Not allowed to be changed | Lower byte of NVRAM charge pump trim |

**Configuration Register 63 (Address 0x3f)**

Figure 64:  
Register 63

| Bit | Bit Name          | Settings   | Bit Description                       |
|-----|-------------------|--|---------------------------------------|
| 7:0 | CHARGE_PUMP[15:8] | Individual, device-specific setting. Not allowed to be changed | Higher byte of NVRAM charge pump trim |

**Important Note:** We guarantee the data for data retention and endurance only under the assumption, that the customer does **not** change the registers 62 and 63. In addition, it is mandatory to follow the given procedure for ERASE NVRAM as described in section [NVRAM and ROM](#) precisely. Otherwise, **we do no longer guarantee** the data retention time and endurance cycles.

## Read Registers

PCap04 has 35 byte of RAM for read access, combined as quadruples of 4 byte.

**Figure 65:**  
Read Registers

| Addr | Name | <D7> | <D6> | <D5> | <D4> | <D3> | <D2> | <D1> | <D0> |
|------|------|------|------|------|------|------|------|------|------|
| 0    | RES0 | 7    |      |      |      |      |      |      | 0    |
| 1    |      | 15   |      |      |      |      |      |      | 8    |
| 2    |      | 23   |      |      |      |      |      |      | 16   |
| 3    |      | 31   |      |      |      |      |      |      | 24   |
| 4    | RES1 | 7    |      |      |      |      |      |      | 0    |
| 5    |      | 15   |      |      |      |      |      |      | 8    |
| 6    |      | 23   |      |      |      |      |      |      | 16   |
| 7    |      | 31   |      |      |      |      |      |      | 24   |
| 8    | RES2 | 7    |      |      |      |      |      |      | 0    |
| 9    |      | 15   |      |      |      |      |      |      | 8    |
| 10   |      | 23   |      |      |      |      |      |      | 16   |
| 11   |      | 31   |      |      |      |      |      |      | 24   |
| 12   | RES3 | 7    |      |      |      |      |      |      | 0    |
| 13   |      | 15   |      |      |      |      |      |      | 8    |
| 14   |      | 23   |      |      |      |      |      |      | 16   |
| 15   |      | 31   |      |      |      |      |      |      | 24   |
| 16   | RES4 | 7    |      |      |      |      |      |      | 0    |
| 17   |      | 15   |      |      |      |      |      |      | 8    |
| 18   |      | 23   |      |      |      |      |      |      | 16   |
| 19   |      | 31   |      |      |      |      |      |      | 24   |
| 20   | RES5 | 7    |      |      |      |      |      |      | 0    |
| 21   |      | 15   |      |      |      |      |      |      | 8    |
| 22   |      | 23   |      |      |      |      |      |      | 16   |
| 23   |      | 31   |      |      |      |      |      |      | 24   |

| Addr | Name     | <D7>         | <D6>            | <D5>         | <D4>        | <D3>        | <D2>        | <D1>        | <D0>        |
|------|----------|--------------|-----------------|--------------|-------------|-------------|-------------|-------------|-------------|
| 24   | RES6     | 7            |                 |              |             |             |             |             | 0           |
| 25   |          | 15           |                 |              |             |             |             |             | 8           |
| 26   |          | 23           |                 |              |             |             |             |             | 16          |
| 27   |          | 31           |                 |              |             |             |             |             | 24          |
| 28   | RES7     | 7            |                 |              |             |             |             |             | 0           |
| 29   |          | 15           |                 |              |             |             |             |             | 8           |
| 30   |          | 23           |                 |              |             |             |             |             | 16          |
| 31   |          | 31           |                 |              |             |             |             |             | 24          |
| 32   | STATUS_0 | POR_FLAG_WDG | POR_FLAG_CONFIG | IR_FLAG_COLL | AUTOBOOT    |             | RDC_READY   | CDC_ACTIVE  | RUNBIT      |
| 33   | STATUS_1 |              |                 |              |             | RDC_ERR     | MUP_ERR     | ERR_OVFL    | COMB_ERR    |
| 34   | STATUS_2 |              | C_PORTER_R_INT  | C_PORTER_R5  | C_PORTER_R4 | C_PORTER_R3 | C_PORTER_R2 | C_PORTER_R1 | C_PORTER_R0 |

The read registers are made of 7 result registers. Addresses 32 to 34 contain the status register.

### Result Registers

The content of the results registers depend on the firmware. The following describes the result registers as they are used by the standard firmware.

**Figure 66:**  
Result Registers

| Name | Length  | Format   | Bit Description  |                  |
|------|---------|--|------------------|------------------|
|      |         |  | Grounded C       | Differential C   |
| Res0 | 32 bits | E.g. C0 = Unsigned fixed-point number:<br>5 bits integer<br>27 bits fractional<br>Min = 0x0 =<br>= 0.0000000<br>Max = 0xFFFFFFFF =<br>= 31.9999995 | Ratio C0 / Cref  | Ratio C1 / C0    |
| Res1 |         |  | Ratio C1 / Cref  | C3 / C2          |
| Res2 |         |  | C2 / Cref        | C5 / C4          |
| Res3 |         |  | C3 / Cref        |                  |
| Res4 |         |  | C4 / Cref        | Zero             |
| Res5 |         |  | C5 / Cref        | Zero             |
| Res6 |         |  | PT1/PTref        | PT1/PTref        |
| Res7 |         |  | PTinternal/PTref | PTinternal/PTref |

The user is free to assign any data to the results registers in his own firmware.

### Status Registers

**Figure 67:**  
STATUS\_0 (Address 32)

| Bit | Bit Name         | Bit Description  |
|-----|------------------|--|
| 7   | POR_Flag_Wdog    | A watchdog overflow has been detected and has provoked a power-on reset. Perhaps the firmware has hung up in an unwanted endless loop or, more likely, a CDC/RDC trigger signal has been lost. |
| 6   | POR_Flag_Config  | One or more configuration bits toggled by interferences and has provoked a power-on-reset.   |
| 5   | POR_CDC_DSP_COLL | If a CDC sequence is triggered while DSP is still active an Initial Reset is provoked.   |
| 4   | AutoBoot busy    |  |
| 3   |                  |  |
| 2   | RDC ready        |  |
| 1   | CDC active       | <b>Warning:</b> Traffic on interface may enhance noise in measurement  |
| 0   | RUNBIT           | The RUNBIT from write register 47 is mirrored here   |

**Figure 68:**  
STATUS\_1 (Address 33)

| Bit | Bit Name | Bit Description   |
|-----|----------|---|
| 7:4 | n.c.     | Test bits   |
| 3   | RDC_Err  | Some kind of error occurred when the RDC unit was busy                  |
| 2   | Mup_Err  | A particular kind of TDC error occurred when the CDC unit was busy      |
| 1   | Err_Ovfl | An overflow error occurred when the CDC unit was busy                   |
| 0   | Comb_Err | All error bits, from here onward, disjunctively combined (using bit-or) |

**Figure 69:**  
STATUS\_2 (Address 34)

| Bit | Bit Name                          | Port            | Bit Description   |
|-----|-----------------------------------|-----------------|---|
| 6   | C_PortError<br>Internal Reference | PC internal ref | In the CDC unit, one or several ports are affected by some error like a short-circuit to ground. May also be a charge/ discharge resistivity too big, a capacitance too big, or an ill-defined precharge/fullcharge/discharge time. |
| 5   | C_PortError5                      | PC5             |   |
| 4   | C_PortError4                      | PC4             |   |
| 3   | C_PortError3                      | PC3             |   |
| 2   | C_PortError2                      | PC2             |   |
| 1   | C_PortError1                      | PC1             |   |
| 0   | C_PortError0                      | PC0             |   |

## Principles of Operation

### Converter Frontend

The device uses “discharge time measurement” as a principle for measuring either capacitances (CDC unit) or resistances (RDC unit). It addresses all ports (PC...,PT...) in time multiplex, the time measurement being done by means of a high-resolution TDC (time-to-digital converter).

### Capacitance-to-Digital Converter (CDC)

#### Measuring Principle

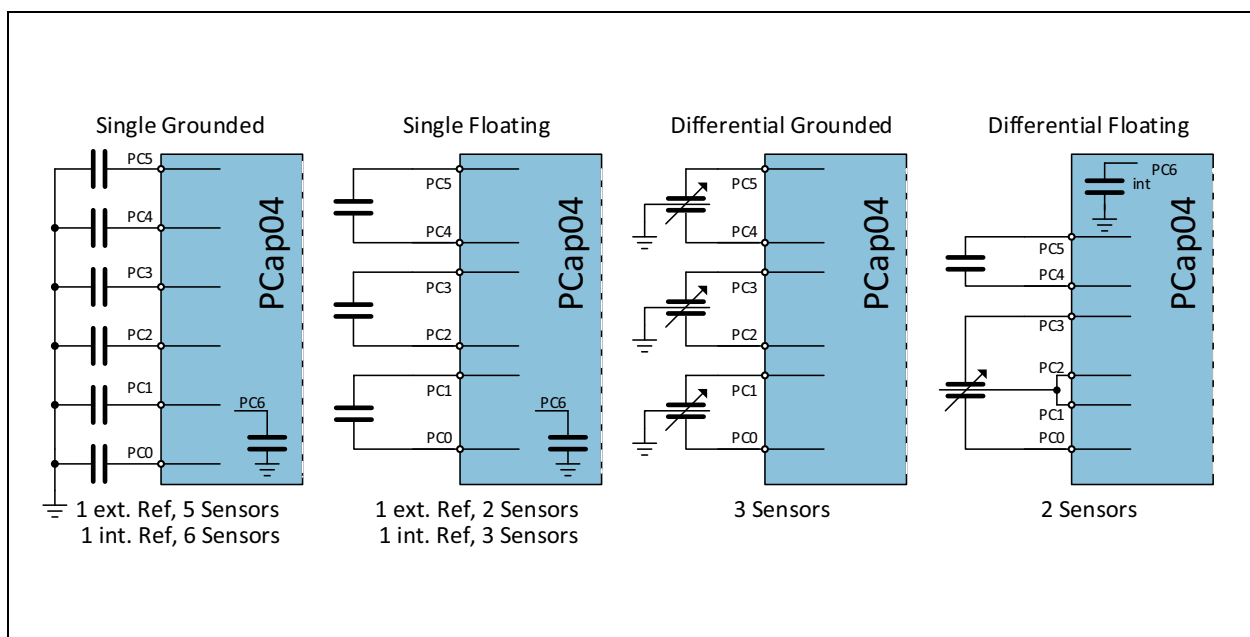
In PCap04, capacitance measurement is done by measuring discharge times of RC-networks. The measurements are ratiometric. This means that the capacitors are compared to a fixed reference or, like in differential sensors, to capacitors with change in opposite direction. Thanks to the short time intervals and special compensation methods, the ratio of discharge times is directly proportional to the ratio of capacitors. The discharge time is defined by the capacitor and the selected discharge resistor.

$$(EQ1) \quad \frac{\tau_N}{\tau_{ref}} = \frac{C_N}{C_{ref}} \quad \tau = k \times R \times C$$

#### Connecting Sensors

PCap04 can handle single and differential sensors in grounded or floating connection. PCap04 has integrated reference capacitors. They are programmable in a range from 1pF to 31pF in steps of 1pF.

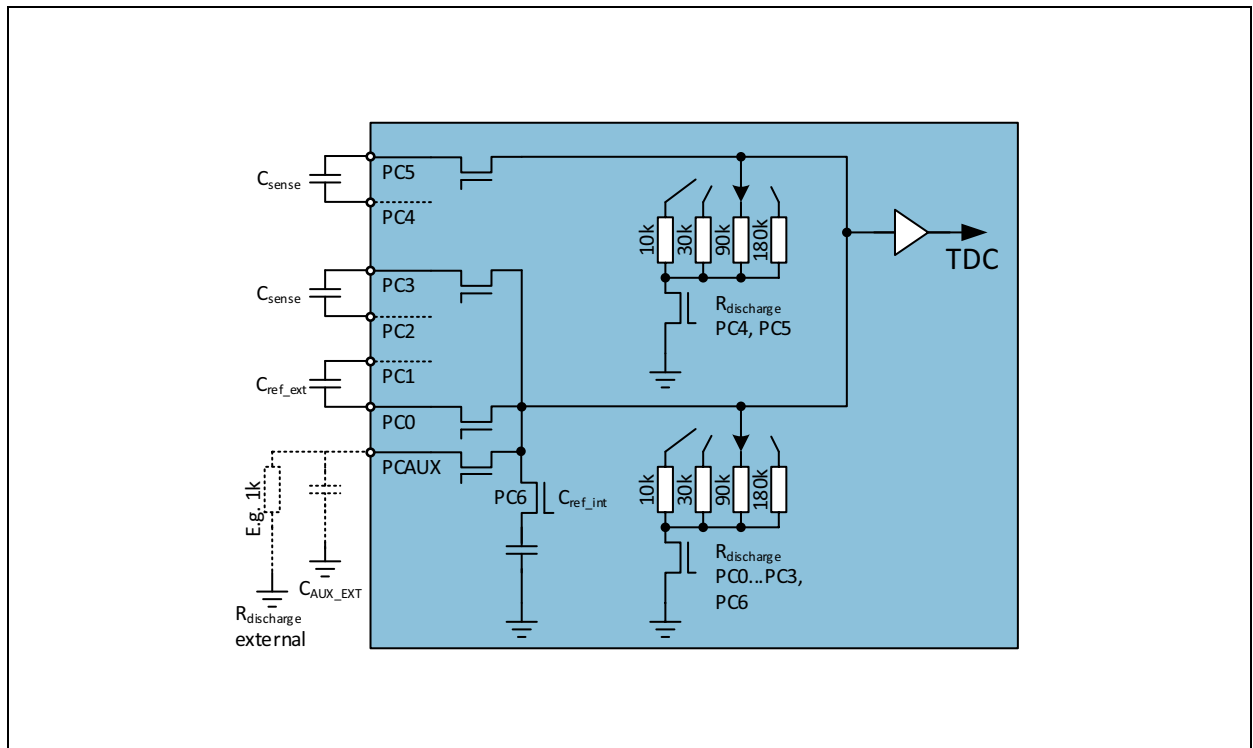
Figure 70:  
Connecting Sensors



**Discharge Resistors**

The PCap04 has two sets of discharge resistors already integrated. One resistor set (10k / 30k / 90k / 180k ohm) is for measurements on port PC0 to PC3 and the internal reference port PC6. The other resistor set (10k / 30k / 90k / 180k ohm) is for ports PC4 and PC5. This way, it is possible to measure different sensors with strongly deviated capacitance like pressure and humidity with one and the same chip. Parameters RCHG\_xxx select the resistors.

**Figure 71:**  
Integrated Discharge Resistors



The user can define which resistor he uses for the internal compensation measurement. It is selected by **DSP\_RDCHG\_COMP\_INT\_SEL** (0 = RCHG0 (default), 1 = RCHG1).

There is the possibility to use an external discharge resistor for handling big capacitances.

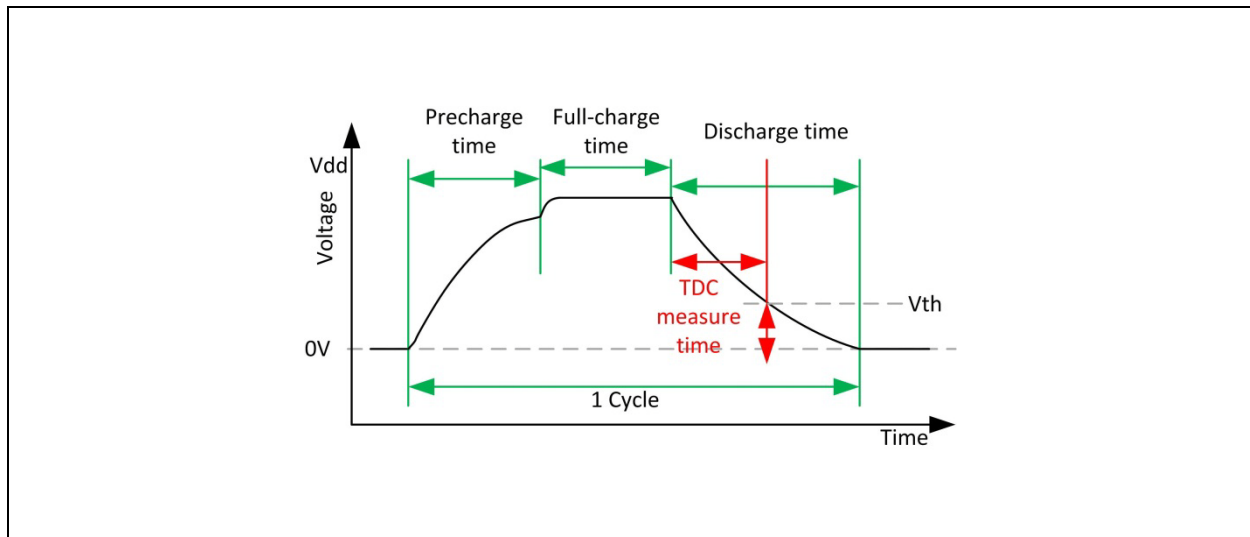


### Cycle

In PCap04 the measurement principle is based on a three-step cycle.

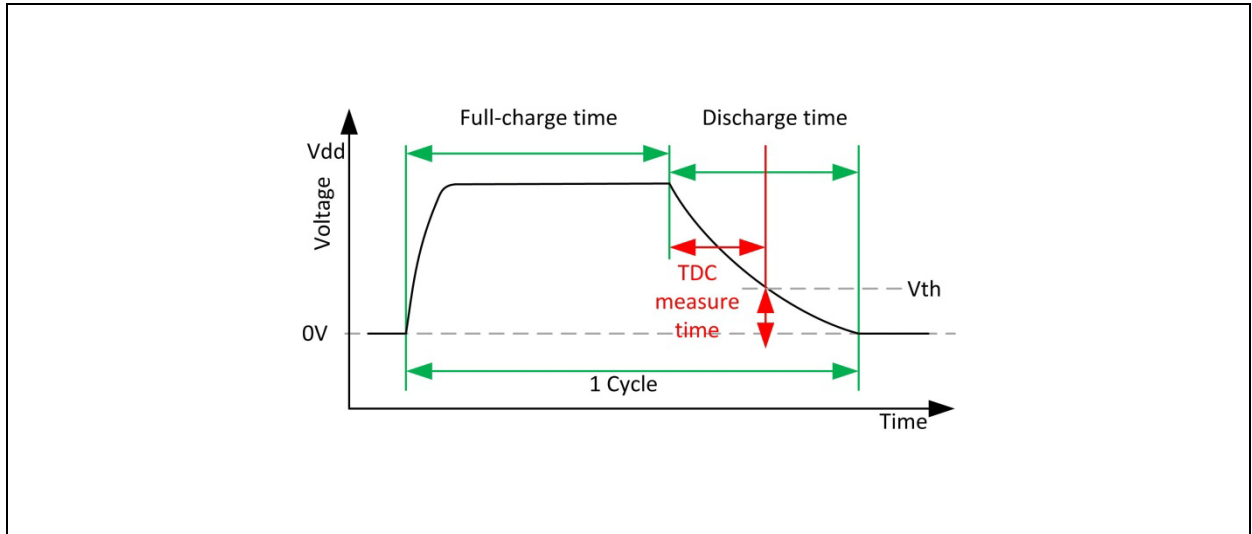
- In the pre-charge phase the capacitor is charged up via a series resistor to a level close to Vdd. The resistor reduces the charge current and reduces the mechanical stress on the sensing capacitor. This can be necessary in some MEMS applications. Further, this is a measure to detect a short circuit and to limit the current even in such an error case.
- In full-charge phase, the capacitor is charged up finally to Vdd without any series resistor.
- Then, in the third step, the capacitor is discharged via the discharge resistor down to 0V. The CDC measures the time interval until a trigger level is reached. All this is called a single "cycle".

**Figure 72:**  
Single Cycle Timing



In applications that do not need the slow charge up but high conversion rate, it is possible to disable the pre-charge option and to start charge up directly without any series resistor.

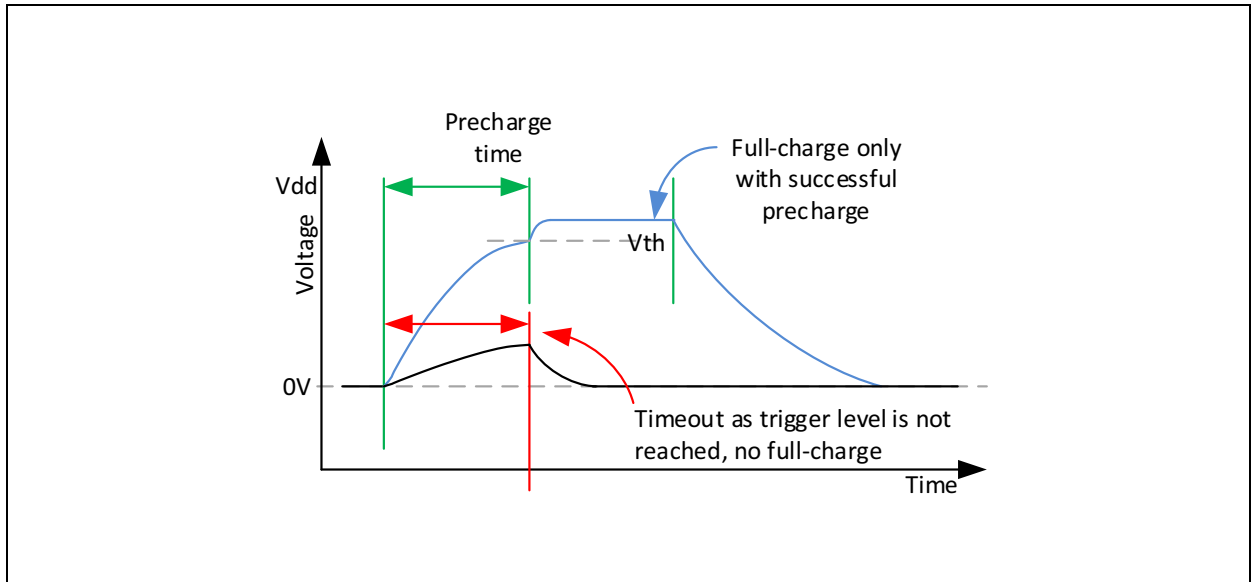
**Figure 73:**  
Single Cycle, Fast Charge



In both cases the capacitors are discharged for the full discharge time period and then connected to GND.

In case of a short circuit, the voltage never reaches the trigger level of the comparator. In such a case the measurement cycle is stopped, no low-resistive full-charge follows. This way the current in a short-circuit case is limited.

**Figure 74:**  
Short Circuit Detection



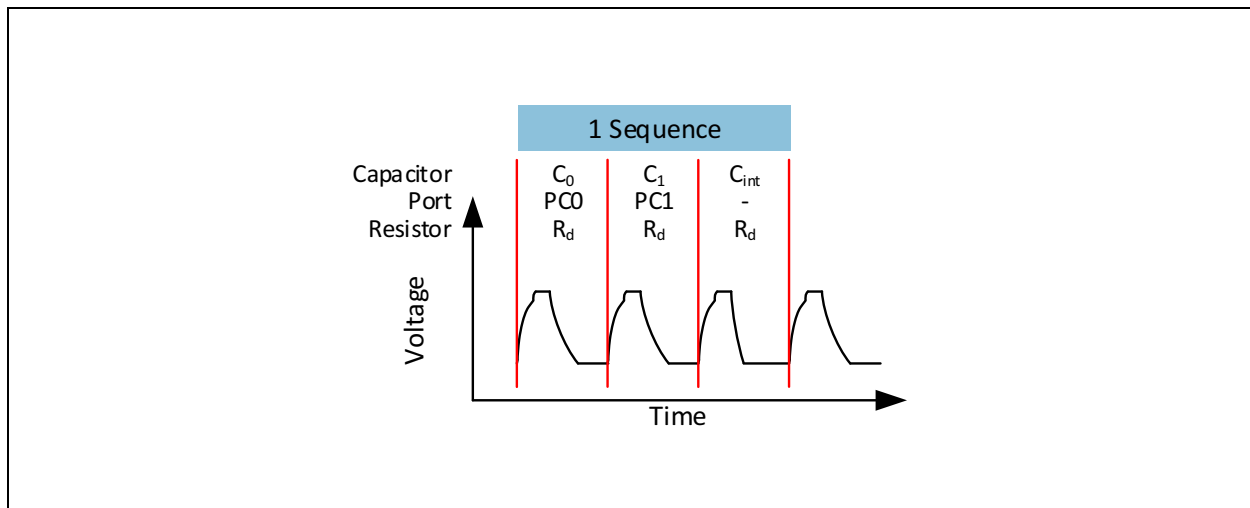
### Sequence

A “sequence” is made of a set of cycles, namely those for the various active ports as well as combinations of them as given by the compensation measurements. The number and kind of single cycles depend on the way of how the sensors are connected, the number of capacitors and the selected compensation options.

For **grounded** sensors, the sequence starts always with PC0 (reference) and then one or more of the other 5 ports. Normally, internal compensation is activated. So the sequence ends with the measurement  $C_{int}$  of the internal stray capacitance/delays. For compensating internal parasitic capacitance and the comparator delay the CDC measures the discharge time with all ports being off ( $C_{int}$ ).

The following figure shows the sequence for a grounded sensor with internal compensation.

**Figure 75:**  
Sequence Grounded

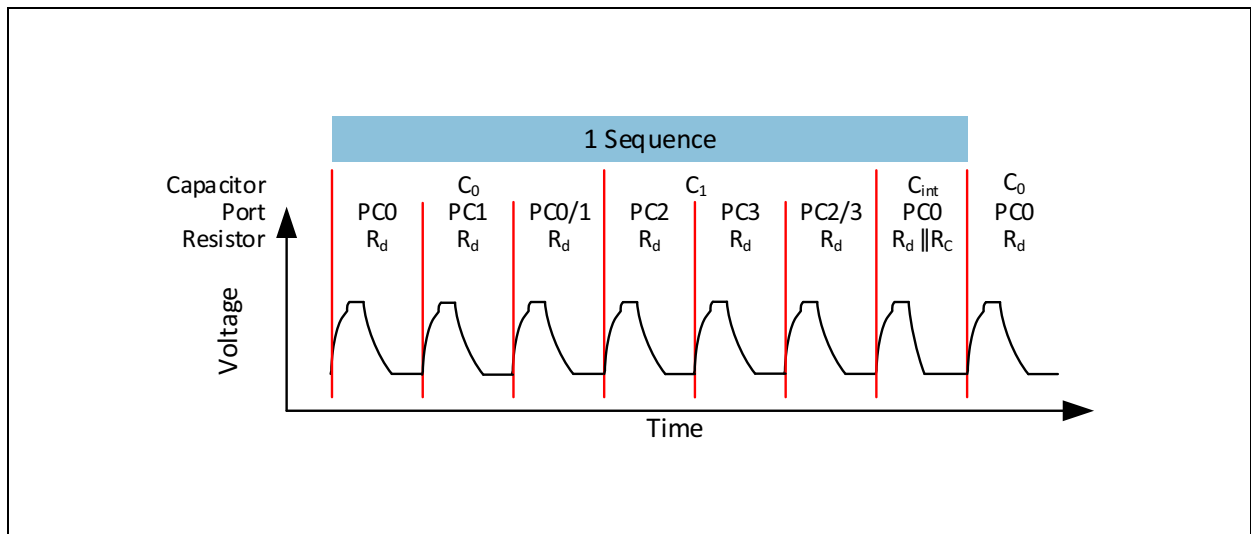


**Sequence** for 1 reference and 1 sensor in grounded connection, compensated for internal capacitance

For **floating** sensors, the sequence starts always with PC0/PC1 (reference), followed by one to three pairs of ports for the sensors. Normally, both compensations (internal and external) are activated.

For compensation of external parasitic capacitances the CDC makes a measurement for each capacitor with both ports being opened. So, for each capacitor 3 measurements are made, e.g. PC0, PC1 and PC0+PC1. The sequence ends with the internal compensation measurement  $C_{int}$ . The following figures show the sequence for 1 floating sensor with full compensation.

**Figure 76:**  
Sequence Floating

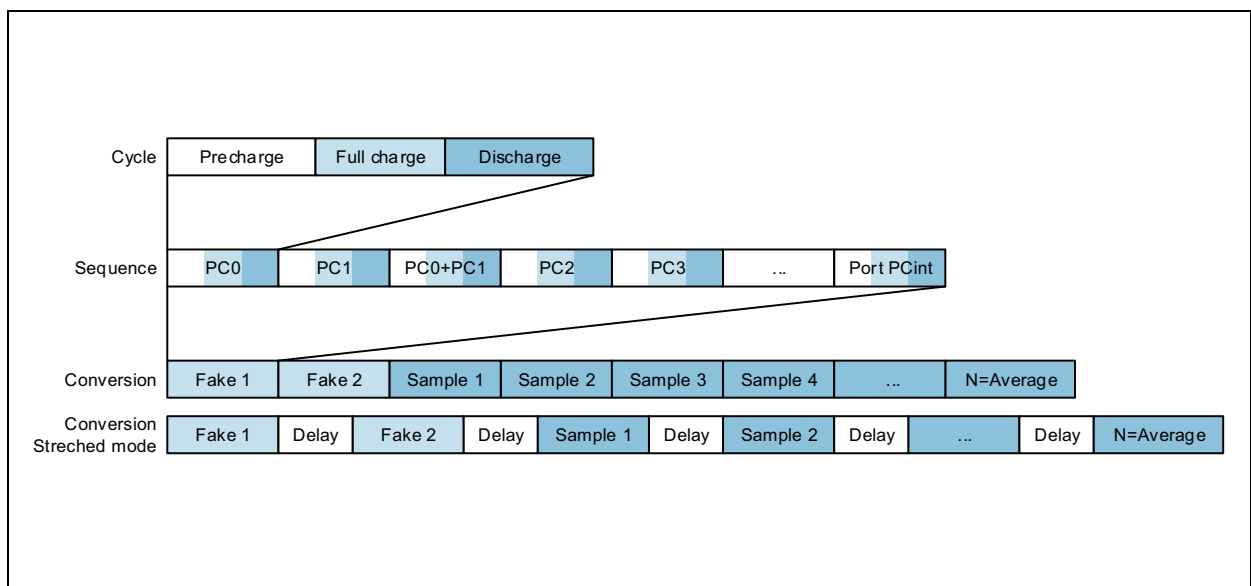


**Sequence** for 1 reference and 1 sensor in floating connection, fully compensated for parasitic capacitances

**Conversion**

Finally, the combination of various sequences and delays in between the sequences define a single “conversion”. Once triggered, a conversion is automatically completed, including all fake measurements and all real measurements defined by sample size for averaging. At the end of a conversion the measurement results are ready for further processing and readout. The end of the conversion is indicated by flag to the DSP and also the RDC unit.

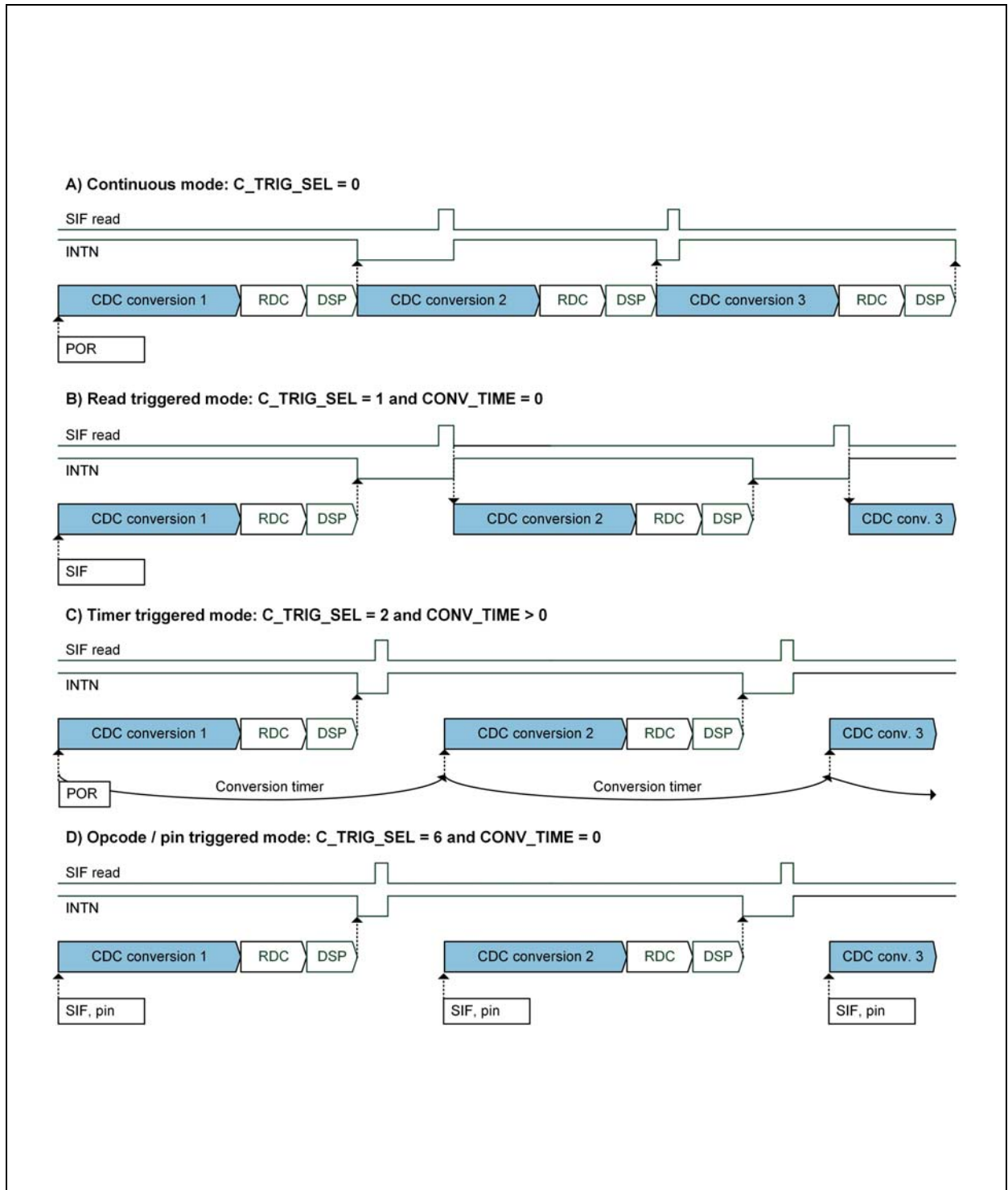
**Figure 77:**  
Cycle-Sequence-Conversion



Various sources are available to trigger the CDC.

- **Continuous mode:** In this mode the CDC starts automatically after POR. The end of the DSP processing triggers directly the next measurement. This mode allows the fastest sample rate, but with the risk that communication will continue while the next measurement is started. Noise will increase.
- **Read triggered mode:** In this mode the very first measurement is triggered via SIF. When the measurement is completed the interrupt goes LOW. An external microcontroller can react on this and read the data. The end of the read via SIF triggers the next measurement. This mode allows fastest measurement without having the interface disturb the measurement.
- **Timer triggered mode:** In this mode the PCAP04 timer triggers the measurements. This is preferred in applications with low sample rate.
- **Opcode or pin triggered mode:** In this mode the external microcontroller has full control on when measurements are triggered. This might be valuable when the measurement needs to be synchronized with other tasks.

Figure 78:  
Conversion Modes



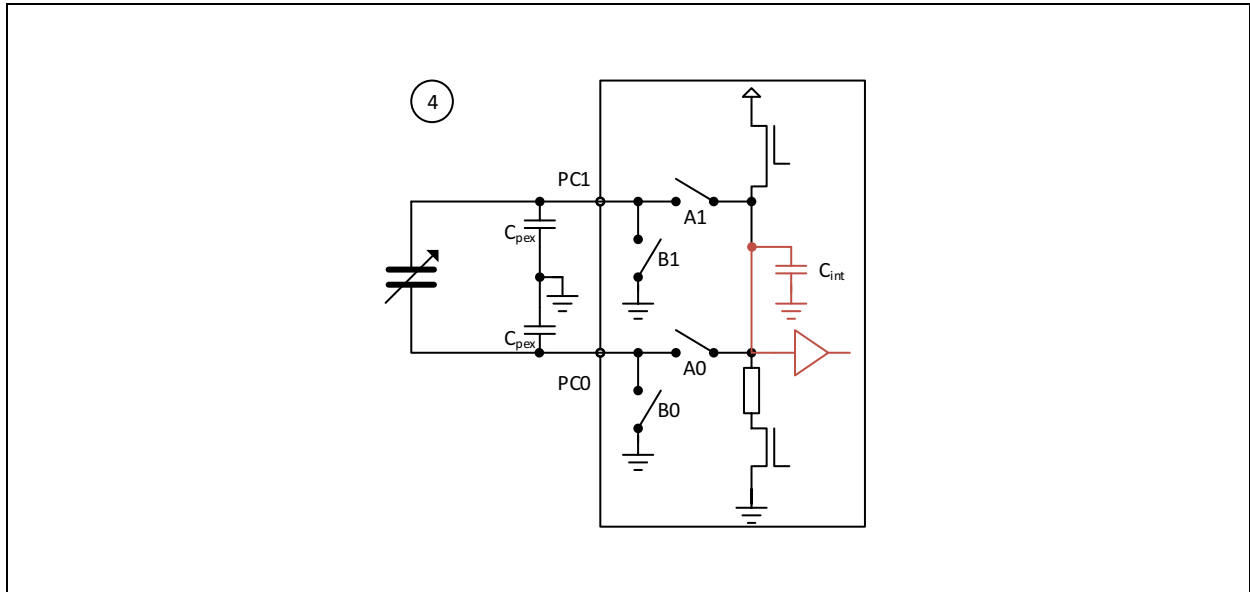
### CDC Compensation Options

#### Internal Compensation

For the internal compensation measurement, both switches A1 and A0 are open. Only the internal parasitic capacitance and the comparator propagation delay will thus be measured.

It is recommended to have internal compensation active in any application.

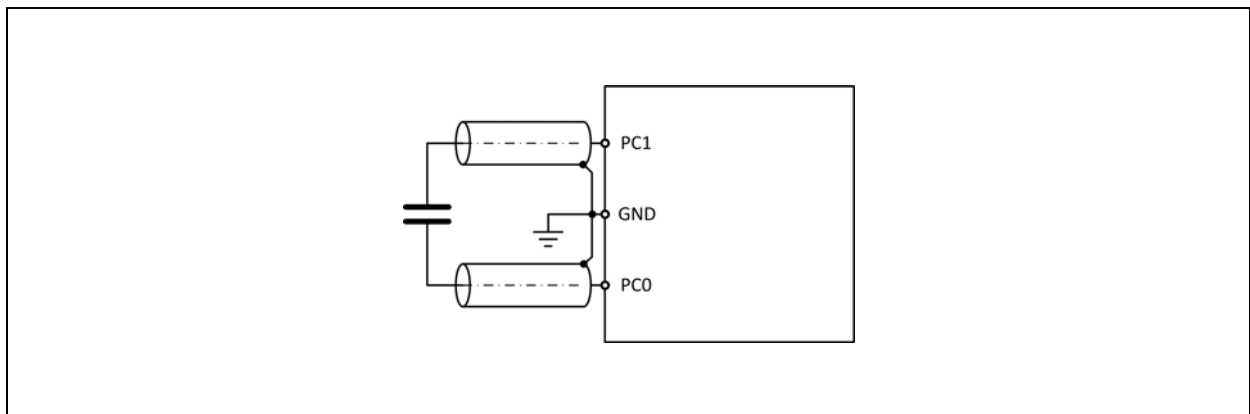
**Figure 79:**  
Internal Compensation Measurement



#### External Compensation

With floating capacitors we have the additional option to compensate external parasitic capacitances against ground. On the PCB, the wire capacitance typically refers to ground. For long wires, it is recommended to use shields which should be grounded at their PCB side.

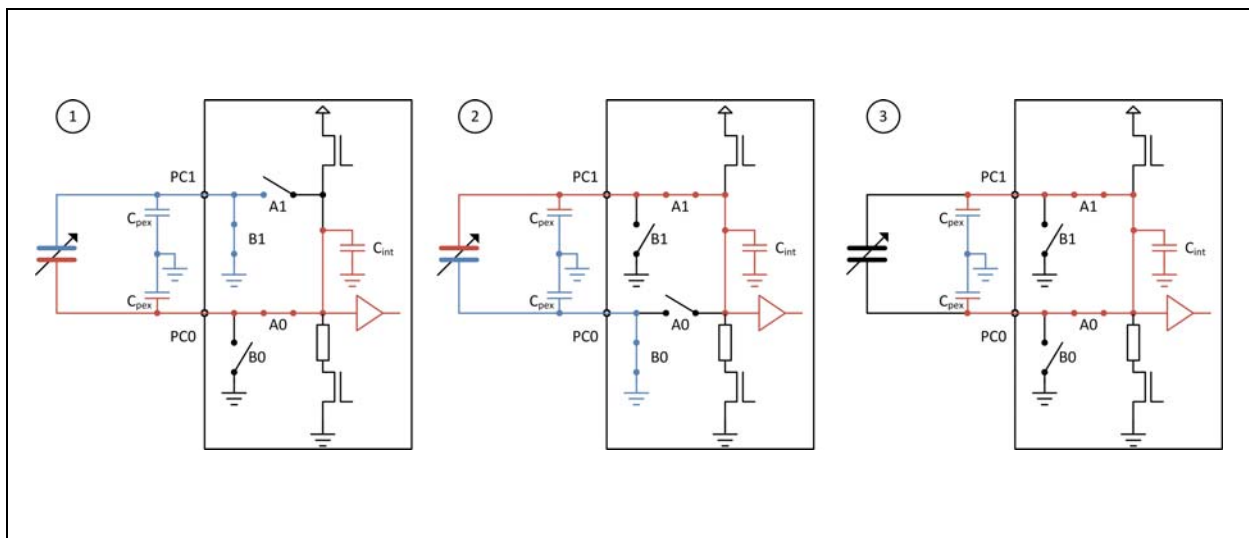
**Figure 80:**  
Shielded Cables



**How to connect** shielded cables for compensation of the external parasitic capacitances.

Three measurements are necessary for each capacitor in case of floating sensors. This is shown in the [Figure 81](#). First, the electrode at PC0 is loaded to VDD18. The discharge time is defined by the sensor capacitance, the parasitic capacitance of the connection, including the chip pad, and the internal capacitance. Second, the same measurement is done for the electrode at PC1. Third, both electrodes are set at VDD18. Therefore the field across the sensor is zero and has no impact. The discharge time includes only the connection and pad capacitance as well as the internal capacitance. Now it is possible to correct mathematically for the parasitic capacitance. This correction is covered by the **ams** firmware.

**Figure 81:**  
External Compensation



Floating capacitors, external compensation measurements, the three measurements that are made for each floating capacitor.

#### DC Balance

When driving floating sensors the sensors' supply is typically DC free.

With differential floating sensors symmetry would be broken. Therefore, PCap04 has the possibility to change the port controlling that the sensors are operated DC free (set by **C\_DC\_BALANCE**).

In applications with grounded sensors the sensors cannot be DC free by principle.



**Gain Correction**

Comparable to classical A/D converters, the PCap04 shows a gain error. But in case of PCap04 the gain error is mainly given by internal parasitic capacitances and the propagation delay of the internal comparator. With internal compensation being active this delay is subtracted from the original measurement. The temperature drift can be approximated linearly and corrected mathematically just by a gain factor. In the standard firmware the gain correction factor could be set in register 35 with fpp 8. The correction factor depends on the discharge time and therefore the RC combination. It has to be evaluated individually for every single application. E.g., with 22pF and 30kOhm the correction factor is 1.25 (0x40).

$$CDC\_GAIN\_CORR = (gain\_corr-1)*256$$

**Empirical method to find the right gain correction factor:**

Replace the sensor with a temperature stable capacitor of the same size (ceramic C0G) as your reference capacitor. (Therefore: quotient = 1, gain = 0). Set the gain correction factor to 1.0. Put the system (PCap04 on PCB) into a temperature chamber and measure the offset drift over temperature. Add an additional temperature stable capacitor to simulate your gain. Measure the gain drift. Increase the gain correction factor and measure the gain drift again. With a gain correction factor >1.0 the gain drift will decrease. If the gain correction factor is set too big then you will see a negative gain drift due to over compensation. The right gain correction factor is found, if the drift is reduced to what you measured at the initial offset drift measurement. Write back the new **CDC\_GAIN\_CORR** value into register 35.

**CDC Important Parameters**

**Cycle Clock**

The basic period  $t_{cycle}$  that defines the cycle time can be derived from the low frequency oscillator or the high frequency oscillator. Parameters **CY\_HFCLK\_SEL** selects in between the two, parameter **CY\_DIV4\_DIS** select between the original 2MHz or a 0.5MHz generated by a divider by 4.

**Figure 82:**  
Configuration of Cycle Clock

| CY_HFCLK_SEL | CY_DIV4_DIS | Cycle Time Base   |
|--------------|-------------|---|
| 0            | 0           | $t_{cycle} = t_{OLF}$ ; $t_{OLF}$ = period low-frequency oscillator     |
| 1            | 0           | $t_{cycle} = 4*t_{OHF}$ ; $t_{OHF}$ = period high-frequency oscillator. |
| 1            | 1           | $t_{cycle} = t_{OHF}$ ; $t_{OHF}$ = period high-frequency oscillator.   |

**Cycle Time**

The pre-charge, full-charge and discharge times of a single cycle are defined in multiples of  $t_{cycle}$ . Those are selected by:

**Figure 83:**  
Cycle Time Configuration

| Reg.  | Configuration Parameter | Description  |
|-------|-------------------------|--|
| 14,15 | PRECHARGE_TIME          | Time to charge via resistor for current limitation. Depends on the cycle clock and, with OHF, on the FULLCHARGE_TIME.<br>1023 = No pre-charge phase<br>OLF: 0 to 1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 1) * t_{cycle}$<br>OHF & FULLCHARGE_TIME = 1023:<br>0 to 1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 2) * t_{cycle}$<br>OHF & FULLCHARGE_TIME != 1023:<br>0 to 1022: $T_{precharge} = (\text{PRECHARGE\_TIME} + 1) * t_{cycle}$ |
| 16,17 | FULLCHARGE_TIME         | Time for final charge without current limitation. Depends on the cycle clock.<br>1023 = No full-charge phase<br>OLF: 0 to 1022: $T_{fullcharge} = (\text{FULLCHARGE\_TIME} + 1) * t_{cycle}$<br>OHF: 0 to 1022: $T_{fullcharge} = (\text{FULLCHARGE\_TIME} + 2) * t_{cycle}$   |
| 12,13 | DISCHARGE_TIME          | Time to discharge the capacitor. Depends on the cycle clock.<br>OLF: 0 to 1023: $T_{discharge} = (\text{DISCHARGE\_TIME} + 1) * t_{cycle}$<br>OHF: 0 to 1023: $T_{discharge} = (\text{DISCHARGE\_TIME} + 0) * t_{cycle}$   |

### Sequence

The length of a sequence depends on the kind and number of sensors, the selected compensation methods and the averaging sample size. The following parameters affect the sequence:

**Figure 84:**  
Sequence Configuration

| Reg. | Configuration Parameter | Description   |
|------|-------------------------|---|
| 6    | <b>C_PORT_EN</b>        | Bitwise enable of the capacitance ports PC0 to PC5<br>0 : Port disabled<br>1 : Port active  |
| 4    | <b>C_REF_INT</b>        | Switches between external and internal reference capacitors. Cannot be used with differential sensors.<br>0 : External, PC0 or PC0 & PC1<br>1 : Internal, PC6 |
| 4    | <b>C_DIFFERENTIAL</b>   | Switches between single and differential sensors<br>0 : Single<br>1 : Differential  |
| 4    | <b>C_FLOATING</b>       | Switches between grounded and floating sensors<br>0 : Grounded<br>1 : Floating  |
| 4    | <b>C_COMP_INT</b>       | Turns on compensation of internal capacitances/delays<br>0 : Off<br>1 : On, recommended   |
| 4    | <b>C_COMP_EXT</b>       | Turns on compensation of external parasitic capacitances. Available only with floating sensors.<br>0 : Off<br>1 : On, recommended                             |
| 5    | <b>C_DC_BALANCE</b>     | Changes Port Control for Differential Floating Mode<br>0 : Off (single HighZ)<br>1 : DC-Free (both HighZ)   |

**Conversion**

The duration of a full conversion has a lower limit given by the number of fake measurements, the averaging and eventually an inter-sequence delay:

**Figure 85:**  
Conversion Configuration

| Reg. | Configuration Parameter | Description  |
|------|-------------------------|--|
| 15   | <b>C_FAKE</b>           | Number of fake measurements (cycles with results being ignored)<br>0 : No dummy cycles<br>1 : 1 dummy cycle<br>...<br>15 : 15 dummy cycles |
| 7, 8 | <b>C_AVRG</b>           | Sample size for averaging within one conversion.<br>0 := No averaging<br>...<br>8191 : Maximum sample size                                 |

The Start of the next conversion depends on the selection of the measurement trigger:

**Figure 86:**  
Conversion Configuration

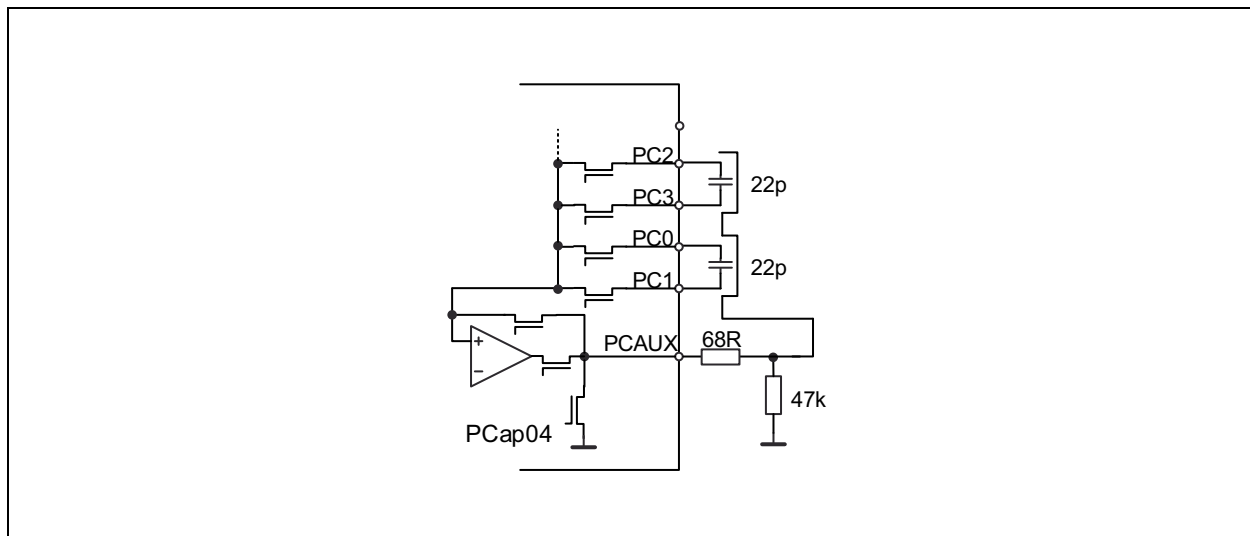
| Reg.      | Configuration Parameter | Description   |
|-----------|-------------------------|---|
| 13        | <b>C_TRIG_SEL</b>       | First trigger selection for CDC trigger<br>0 : Continuous<br>1 : Read triggered<br>2 : Timer triggered<br>3 : Timer triggered (stretched)<br>4 : n.d.<br>5 : Pin triggered<br>6 : Opcode triggered<br>(7 : continuous_exp, not recommended) |
| 13        | <b>C_STARTONPIN</b>     | Selects the GPIO that triggers the CDC measurement  |
| 9, 10, 11 | <b>CONV_TIME</b>        | Sets the conversion time in multiples of twice the period of the low-frequency clock.<br>$t_{conv} = 2 * \mathbf{CONV\_TIME} * t_{ofl}$   |

### Guarding

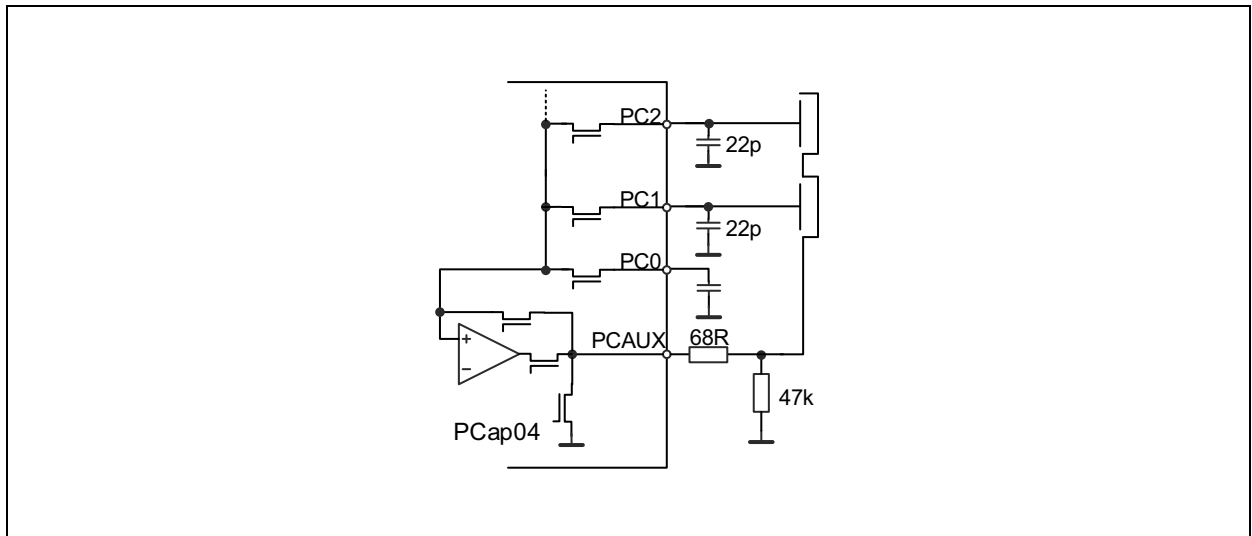
Depending on the sensor topology it may be necessary to add an active shield to suppress disturbing capacitances. The guard electrode is an additional metal area behind the sensing electrode, which is kept at the same potential as the sensing electrode. So all material in between the sensing electrode and the guard electrode are potential-free and therefore have no capacitive effect. It eliminates material-dependent temperature drifts of the sensor PCB. Further, things behind the shield are separated from the sensor by the potential-free zone.

In PCap04 the driver for the guard is integrated. This amplifier needs to have a low-capacitive input to not disturb the measurement path. Gain is ideally one. The gain can be set  $> 1$  (overcompensation) by means of  $C\_G\_OP\_VU$  so that in combination with an external voltage divider it is possible to match the port and wire resistance. The guard is connected to pin PCAUX (other functions of this pin are then not available). In-active ports are also switched to guard so that there is no additional capacitance seen between guard and inactive ports.

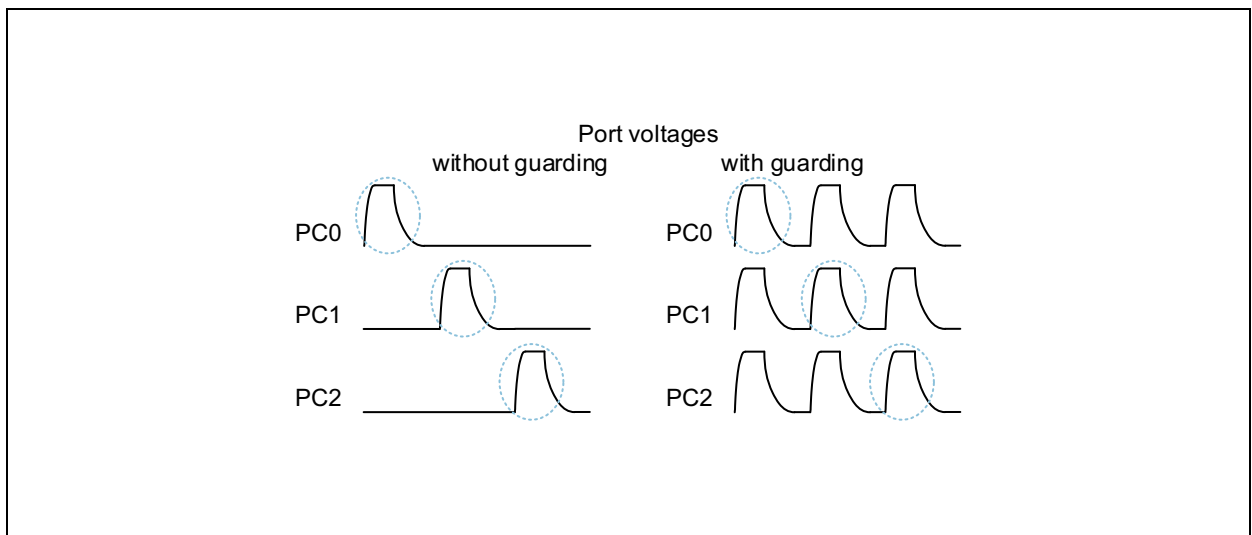
**Figure 87:**  
Guarding with Floating Capacitors



**Figure 88:**  
Guarding with Grounded Capacitors

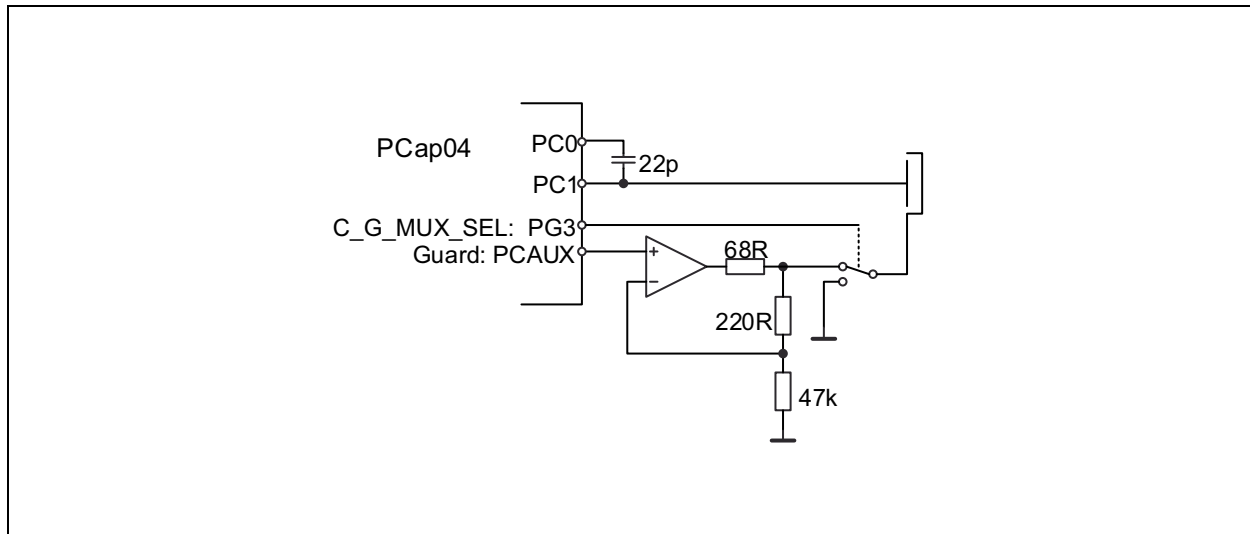


**Figure 89:**  
Port Voltages with Guarding



With guarding, also ports are set to VDD18, even the ones that are not measured. This ensures that electric fields between those ports and the active port are zero. Internally, those ports are not connected to the time measuring path but to the guard driver. If the guard electrode is so big that it cannot be driven by the internal amplifier it is possible to add an external one. An external analog multiplexer will be needed, too. The external amplifier is connected to PCAUX. The SEL port of the multiplexer is connected to PG3.

**Figure 90:**  
Guarding with External Amplifier



**Figure 91:**  
Important Parameters are Set in Register 18:

| Reg. | Configuration Parameter | Description   |
|------|-------------------------|---|
| 18   | <b>C_G_EN[5:0]</b>      | Individual guard enable for each port PC0 to PC5  |
| 18   | <b>C_G_OP_RUN</b>       | 0 : Permanent: Guarding OP is permanent activated (additional power consumption)<br>1 : Pulsed: Guarding OP set to sleep mode between CDC conversions |
| 19   | <b>C_G_OP_ATTN</b>      | Capacitive attenuation of guarding OP:<br>0 : 0.5pF<br>1 : 1.0pF<br>2 : 1.5pF<br>3 : 2.0pF  |
| 19   | <b>C_G_OP_VU</b>        | OP Gain (from sense port to guard)<br>0 : 1.00<br>1 : 1.01<br>2 : 1.02<br>3 : 1.03  |
| 20   | <b>C_G_OP_TR</b>        | Trim power consumption and driving strength of guarding OP<br>0 : Min.<br>...<br>7 : Max.   |
| 18   | <b>C_G_OP_EXT</b>       | Switch between internal guarding OP and an optional external OP<br>0 : Internal OP<br>1 : External OP, PG3 as <b>C_G_MUX_SEL</b>                      |
| 19   | <b>C_G_TIME</b>         | $t_{pp1} = t_{OHF} \cdot C\_G\_TIME$  |

- **C\_G\_EN[5:0]** guarding is activated individually for each port PC0 to PC5. One or more ports can be enabled. The guard output (PCAUX) drives voltage the enabled ports are enabled.
- **C\_G\_TIME** controls the pre-charge phase. Because internal circuits the pre-charge phase is divided into two phases:
  - Pre-charge phase 1: PCAUX is directly connected with active PC.
  - Pre-charge phase 2: PCAUX is driven by OP,  $V_{PCAUX} = gain_{guard} \cdot OP \cdot V_{PCactive}$ . PCAUX is driven by OP until finishing current port cycle.

**Attention:** For guarding, the internal OX/OHF is mandatory:

- **OX\_RUN** > 0, **OX\_DIS** = 0; **OX\_STOP** = 0; **OX\_DIV4** = 0;
- **CY\_HFCLK\_SEL** = 1;

### RDC Resistance-to-Digital Converter

#### Measuring Principle

In PCap04 resistance measurement is also done by measuring discharge times. The measurements are ratiometric. This means, the temperature-sensitive resistances are compared with a fixed reference. The ratio of discharge times is directly proportional to the ratio of resistors. The discharge time is defined by the resistors and the load capacitance.

$$(EQ2) \quad \frac{\tau_N}{\tau_{ref}} = \frac{R_\theta}{R_{ref}} \quad \tau = k \times R \times C$$

#### Connecting Sensors

The chip device has two on-chip resistor elements for the measurement of temperature, an aluminum strip with  $TK \approx 2800 \text{ppm/K}$  as a sensor and a poly-silicon resistor with  $TK$  “close to zero” as a reference. In the range 0°C to 100°C the aluminum sensor can be well approximated by a linear function of temperature.

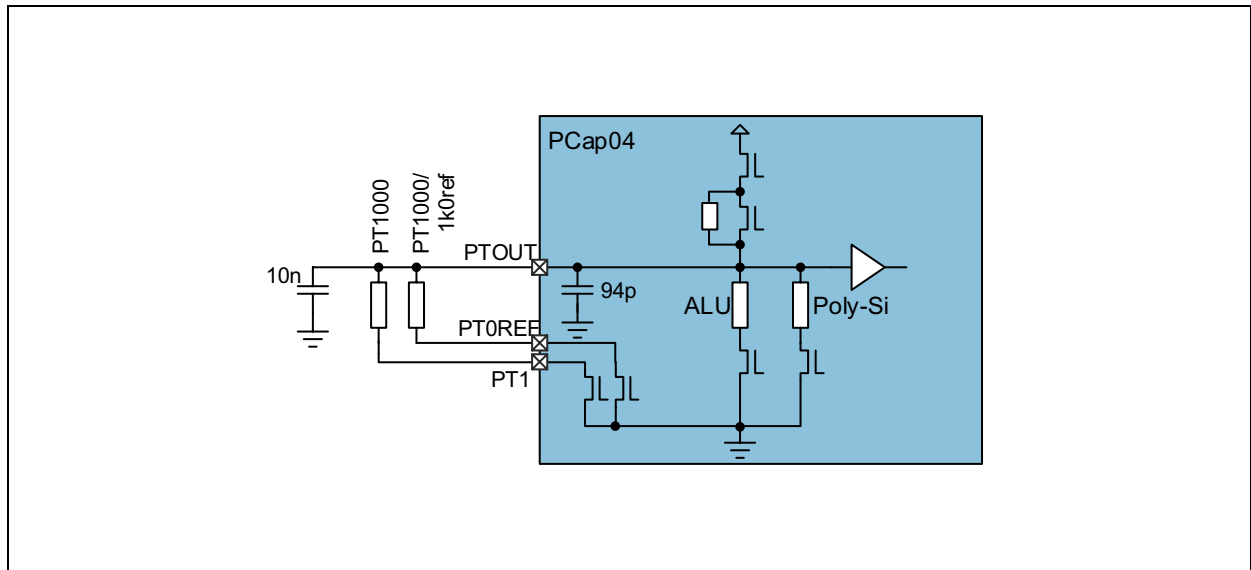
As an alternative, it is possible to connect up to two external sensors. One of those can be used as external reference alternately. External and internal thermometers/reference may be mixed, e.g. an external PT1000 may be compared to the internal Poly-Si resistor.

The chip has an internal capacitor of about 94pF. In combination with the internal resistors the discharge time is about 500ns and the typical resolution of the resistance ratio is better than 13 bits. For precision measurements we recommend to connect an external capacitor of about 10nF. With 10nF the discharge time is in the order of 20 μs and the resolution in the order of 15 bits.

Discharge time must not exceed 20μs. For the capacitor, C0G ceramics yields best performance, while X7R material yields fair results.



**Figure 92:**  
Connecting Temperature Sensors



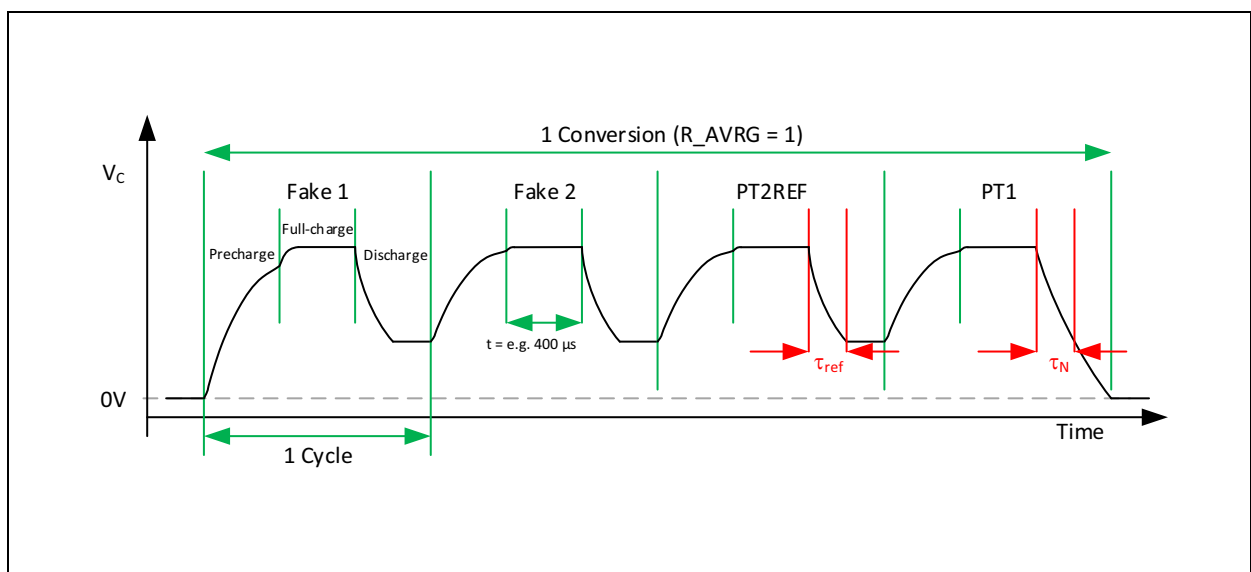
**Note(s):**

1. The RDC measurement is based on an AC principle. Long cables with their parasitic capacitance and resistance will disturb and it is recommended to have short cables ( $\leq 0.5\text{m}$ ), ideally twisted and shielded.

**Cycle & Conversion**

In PCap04 the resistance measurement is running in three phases, like in capacitance measurement: Pre-charge – Full-charge – Discharge. The timing is based on the internal low-frequency oscillator (OLF). The duration of full and discharge phases can be 1 or 2 periods of this reference. The conversion starts with 2 or 8 fake measurements to improve the stability of data. For each single conversion the averaging can be selected with sample size 1, 4, 8 or 16.

**Figure 93:**  
RDC Conversion



**RDC Conversion:**  $R\_AVRG = 1$ , Reference and sensor, 2 fake measurements

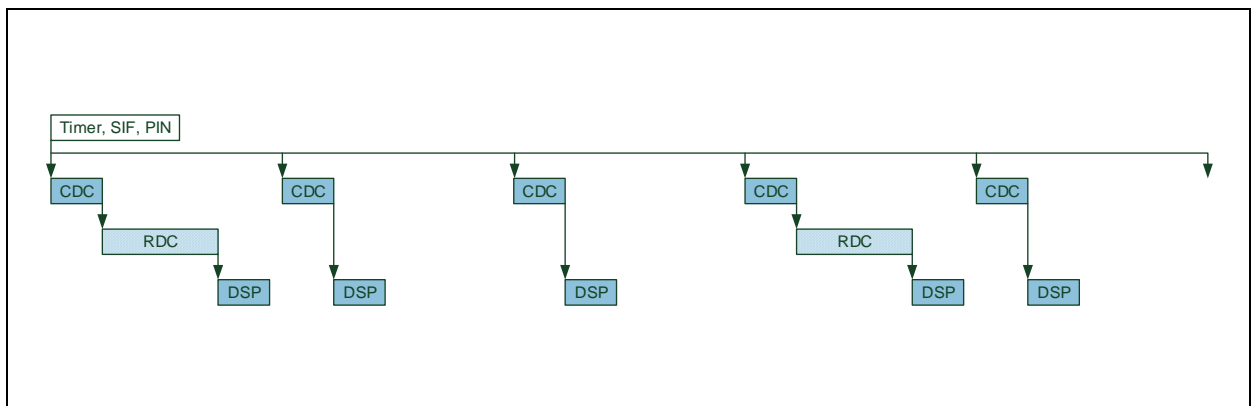
**Trigger**

Various sources can trigger the RDC. The trigger rate can be set to a divider of the CDC trigger rate by means of parameter R\_TRIG\_PREDIV (1 to 1023).

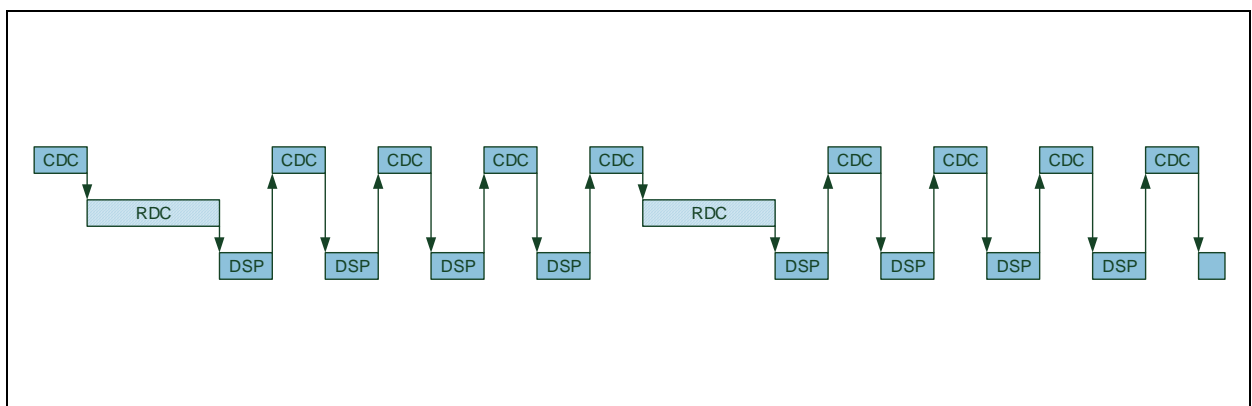
Parameter R\_TRIG\_SEL defines the various possibilities to trigger a resistance measurement:

- Serial Interface command, PIN or DSP (Figure 64)
- Timer triggered, based on the OLF (Figure 64)
- CDC end of conversion
  - Asynchronous: The DSP is triggered by the RDC end of conversion. If RDC rate is less than CDC rate the DSP is triggered directly from the CDC for inactive RDC conversions (Figure 95).
  - Synchronous: The DSP is triggered by the RDC end of conversion. Assuming that RDC rate is less than the CDC rate, the inactive RDC conversions are replaced by a delay (Figure 96).

**Figure 94:**  
RDC Timing, Triggered by Timer, SIF or Pin

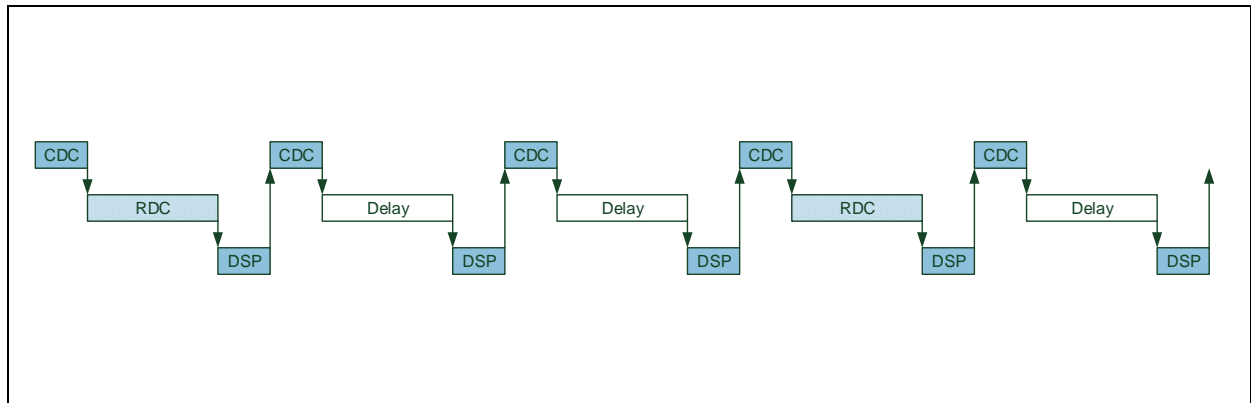


**Figure 95:**  
RDC Timing, Triggered by DSP in Asynchronous Mode



R\_TRIG\_PREDIV = 3, R\_TRIG\_SEL = 3'b101, DSP\_START\_EN: CDC\_TRIG\_EN = 0, RDC\_TRIG\_EN = 1

**Figure 96:**  
RDC Timing, Triggered by DSP in Synchronous Mode



`R_TRIG_PREDIV = 3, R_TRIG_SEL = 3'b110, DSP_START_EN: CDC_TRIG_EN = 1, RDC_TRIG_EN = 0`

**RDC Important Parameters**

*Cycle Clock*

The base frequency for the temperature measurement is the low frequency oscillator (OLF). A further bit, **R\_CY**, specifies whether 1 or 2 periods define the length of the three phases.

**Figure 97:**  
Cycle Clock Configuration

| OLF Frequency | $t_{\text{precharge}} = t_{\text{fullcharge}} = t_{\text{discharge}}$ |          |
|---------------|---|----------|
|               | R_CY = 0  | R_CY = 1 |
| 10 kHz        | 100µs   | 200µs    |
| 50 kHz        | 20µs  | 40µs     |
| 100 kHz       | 10µs  | 20µs     |
| 200 kHz       | 20µs  | 40µs     |

**Sequence**

The major settings for the sequence are the number of ports, the fakes, the reference and averaging.

**Figure 98:**  
Sequence Configuration

| Reg. | Configuration Parameter | Description                            |
|------|-------------------------|--|
| 23   | <b>R_PORT_EN</b>        | Enable ports PT0REF, PT1               |
| 23   | <b>R_PORT_EN_IREF</b>   | Enable the internal reference resistor |
| 23   | <b>R_PORT_EN_IMES</b>   | Enable the internal temperature sensor |
| 22   | <b>R_AVRG</b>           | Set averaging for T measurement        |
| 23   | <b>R_FAKE</b>           | Set number of fake measurements        |

**Conversion**

**Figure 99:**  
RDC Trigger Configuration

| Reg.   | Configuration Parameter | Description   |
|--------|-------------------------|---|
| 22     | <b>R_TRIG_SEL</b>       | Selection of trigger source for RDC unit  |
| 21, 22 | <b>R_TRIG_PREDIV</b>    | Pre-divider to set the RDC rate as fraction of the CDC rate but also to the <b>OLF_CLK</b> when <b>OLF_CLK</b> is selected as RDC Trigger<br>0 = 1 : RDC conversion with each CDC conversion<br>2 : RDC conversion every second CDC conversion<br>...<br>1023 : Maximum setting |
| 23     | <b>R_STARTONPIN</b>     | Select Pin for pin triggered  |

**RDC Results, Ratios**

PCap04\_standard and PCap04\_linearize firmware are determining ratios between sense ports and reference ports.

If the internal reference port is activated (**R\_PROT\_EN\_IREF**: 1) for all other ports (internal sense, PT0 and PT1) the internal reference is automatically selected for ratios.

If **R\_PORT\_EN\_IREF**: 0, external port PT0/Ref is selected as reference value for all other ports (internal sense, PT1)

## Interfaces (Serial & PDM/PWM)

### Serial Interfaces (SIF)

Two types of serial interfaces are available for communication with a microcontroller and for programming the device: SPI and IIC. Only one interface is available at a time, selected by pin IIC\_EN. On both interfaces, the PCap04 can operate as slave only.

**Figure 100:**  
Serial Interface Selection

| Pin             | Description   |
|-----------------|---|
| IIC_EN = GROUND | 4-wire SPI interface<br>General-purpose I/O pins PG0 and PG1 are not available  |
| IIC_EN = VDD    | 2-wire I <sup>2</sup> C interface<br>All general-purpose I/O pins are available |

IIC\_EN may not be floating. Connect IIC\_EN to VDD if there is no need for a controller interface.

The serial interfaces allow read access to the read registers (results and status), read/write access to the configuration registers and read/write access to the NVRAM (explicitly the SRAM part of the NVRAM).

All commands for write or read to memory or configuration / read registers may use explicit addressing or address auto-increment.

Opcodes

Figure 101:  
Opcodes

| Description              | Byte 2 |   |                               |   |   |   | Byte 1                    |   |   |   | Byte 0                    |             |   |   |   |   |  |
|--------------------------|--------|---|-------------------------------|---|---|---|---------------------------|---|---|---|---------------------------|-------------|---|---|---|---|--|
| wr_mem (NVRAM)           | 1      | 0 | 1                             | 0 | 0 | 0 | add<9...0> <sup>(2)</sup> |   |   |   | data<7...0>               |             |   |   |   |   |  |
| rd_mem (NVRAM)           | 0      | 0 | 1                             | 0 | 0 | 0 | add<9...0> <sup>(2)</sup> |   |   |   | data<7...0>               |             |   |   |   |   |  |
| Write configuration      | 1      | 0 | 1                             | 0 | 0 | 0 | 1                         | 1 | 1 | 1 | add<5...0> <sup>(2)</sup> | data<7...0> |   |   |   |   |  |
| Read configuration       | 0      | 0 | 1                             | 0 | 0 | 0 | 1                         | 1 | 1 | 1 | add<5...0> <sup>(2)</sup> | data<7...0> |   |   |   |   |  |
| Read result (RAM)        | 0      | 1 | add<5...0> <sup>(2) (3)</sup> |   |   |   | data<7...0>               |   |   |   |                           |             |   |   |   |   |  |
| POR (Power-on Reset)     | 1      | 0 | 0                             | 0 | 1 | 0 | 0                         | 0 |   |   |                           |             |   |   |   |   |  |
| Initialize               | 1      | 0 | 0                             | 0 | 1 | 0 | 1                         | 0 |   |   |                           |             |   |   |   |   |  |
| CDC Start conversion     | 1      | 0 | 0                             | 0 | 1 | 1 | 0                         | 0 |   |   |                           |             |   |   |   |   |  |
| RDC Start conversion     | 1      | 0 | 0                             | 0 | 1 | 1 | 1                         | 0 |   |   |                           |             |   |   |   |   |  |
| dsp_trig                 | 1      | 0 | 0                             | 0 | 1 | 1 | 0                         | 1 |   |   |                           |             |   |   |   |   |  |
| nv_store <sup>(1)</sup>  | 1      | 0 | 0                             | 1 | 0 | 1 | 1                         | 0 |   |   |                           |             |   |   |   |   |  |
| nv_recall <sup>(1)</sup> | 1      | 0 | 0                             | 1 | 1 | 0 | 0                         | 1 |   |   |                           |             |   |   |   |   |  |
| nv_erase <sup>(1)</sup>  | 1      | 0 | 0                             | 1 | 1 | 1 | 0                         | 0 |   |   |                           |             |   |   |   |   |  |
| Test read                | 0      | 1 | 1                             | 1 | 1 | 1 | 1                         | 0 | 0 | 0 | 0                         | 1           | 0 | 0 | 0 | 1 |  |

Note(s):

1. Set MEM\_CTRL before using nv\_store, nv\_recall or nv\_erase
2. Auto-incremental write/read is supported by both, SPI and I<sup>2</sup>C
3. Address range for Read result is 0 to 24 (8 x 32bit result registers and 3 x 8 bit status registers)

The serial interface is tested most easily by performing a test read:

Write opcode 0x7e to SIF and read 1 byte. Compare this byte to following patterns:

- 0x11 : Expected value, read cycle performed correctly
- 0x88 : Failure: there is a big/little-endian swap
- 0xEE : Failure: during read cycle all bits are inverted
- 0x77 : Failure: inverted bits and bit/little-endian swap

**Synchronous Read**

For best results it is recommended to read all values from the result registers synchronized by the INTN signal. The INTN signal can be routed to PG4 or PG5 by **PG4\_INTN\_EN** and **PG5\_INTN\_EN** at Register 30. The INTN signal is low active, which means the negative edge of INTN signals new values are available at Res0 to Res7. The INTN is set back to High by a positive edge at SSN (SPI) or a stop condition (I<sup>2</sup>C).

**Asynchronous Read**

If it is not possible to read synchronously as described above for any reason, asynchronous read (**EN\_ASYNC\_RD**) has to be enabled in register 42. In this mode values in result registers Res0 to Res7 are only updated if the previous value has been read (INTN is reset to High by a positive edge of SSN or a stop condition).

**I<sup>2</sup>C Compatible Interface**

The present paragraph outlines the PCap04 device specific use of the I<sup>2</sup>C interface. The external I<sup>2</sup>C master begins the communication by creating a start condition, a falling edge on the SDA line while SCL is HIGH. It stops the communication by a stop condition, a rising edge on the SDA line while SCK is high. Data bits are transferred with the rising edge of SCK.

On I<sup>2</sup>C buses, every slave holds an individual 7-bit device address with 5 fixed and 2 configurable bits. This address has always to be sent as the first byte after the start condition, the eighth bit indicating the direction of the following data transfer (R=read=1 and W=write=0).

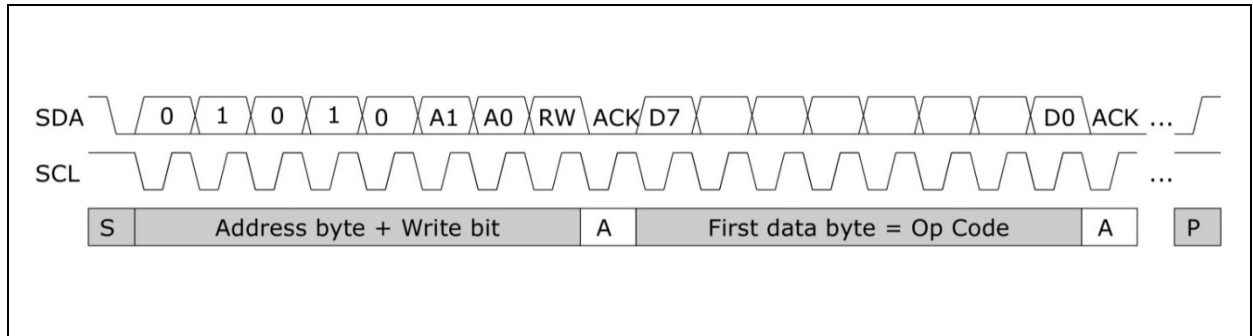
**Figure 102:**  
**Address Byte**

|            |   |   |   |   |          |    |            |
|------------|---|---|---|---|----------|----|------------|
| <b>MSB</b> |   |   |   |   |          |    | <b>LSB</b> |
| 0          | 1 | 0 | 1 | 0 | A1       | A0 | R/W        |
| Fixed      |   |   |   |   | Variable |    | key        |

**I<sup>2</sup>C Timing**

The address byte is followed by the opcode and eventually the payload. Each byte is followed by an acknowledge bit (= 0, when a slave acknowledges).

**Figure 103:**  
I<sup>2</sup>C Typical Sequence

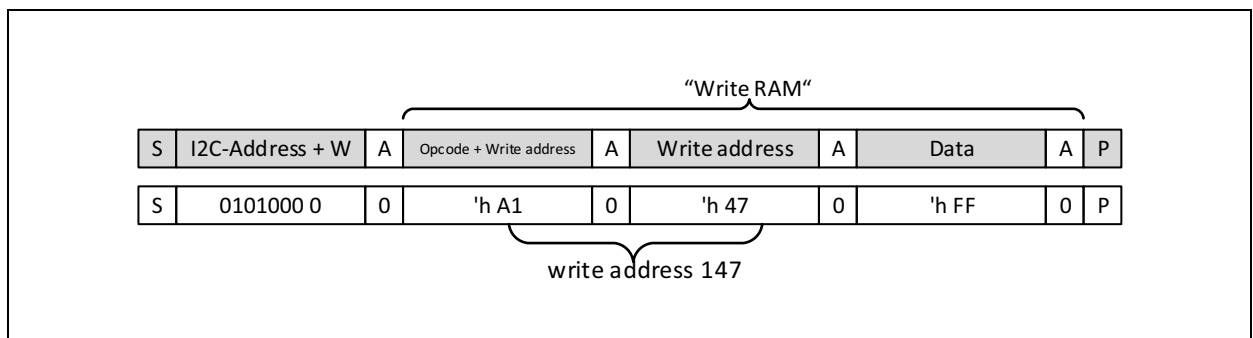


**I<sup>2</sup>C Write**

During write transactions, the master alone sends data, the addressed slave just sends the acknowledge bits. The master first sends the slave address plus the write bit. Then it sends the PCap04 specific opcode including the register address in the slave. Finally it sends the payload (“Data”).

Incremental writing is possible, means, for a consecutive set of data only the start address has to be sent and a various number of data could be sent in one row.

**Figure 104:**  
I<sup>2</sup>C Write Procedure



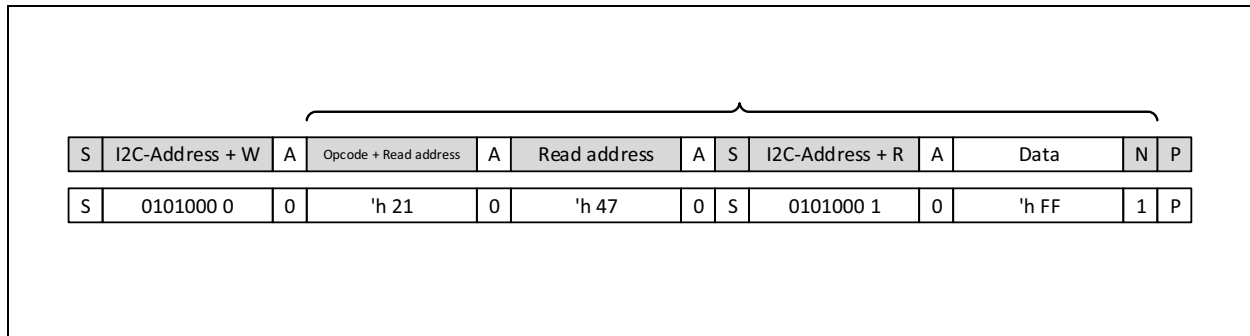
**Example:** Write 0xFF as a datum to the SRAM at address 0x147.



### I<sup>2</sup>C Read

During read transactions, the direction of communication has to be commuted. Therefore, the master creates again a start condition (resp. restart: start without stop condition in between) and sends the slave address plus the read bit to switch into read mode. Figure 105 shows an example with op code “read from SRAM”.

**Figure 105:**  
I<sup>2</sup>C Read Procedure



**Example:** Read from SRAM address 0x147, we find 0xFF having been programmed before.

After arrival of the first (or any) data byte, the master may either signal

- Not-Acknowledge = N = 1 to indicate “end read”, “stop sending” to the slave, or
- Acknowledge = A = 0 to indicate “continue in automatic address-increment mode” and thus receive many bytes in a row. As one can see, automatic address increment is particularly useful and efficient with the I<sup>2</sup>C interface.

### SPI Interface

Clock Polarity, Clock Phase and Bit Order: The following choices are necessary for successful operation.

**Figure 106:**  
SPI Settings

| SPI - Parameter | Description        | Setting      |
|-----------------|--------------------|--------------|
| CPOL            | Clock polarity     | 0            |
| CPHA            | Clock phase        | 1            |
| Mode            | SPI Mode           | 1            |
| DORD            | Bit sequence order | 0, MSB first |

SPI Timing

Figure 107:  
SPI Write

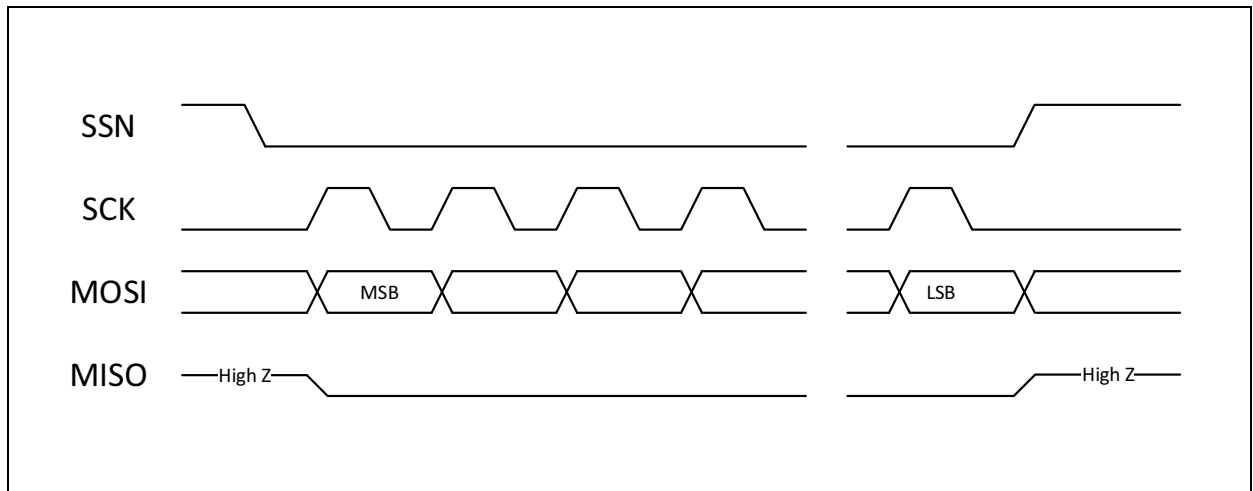
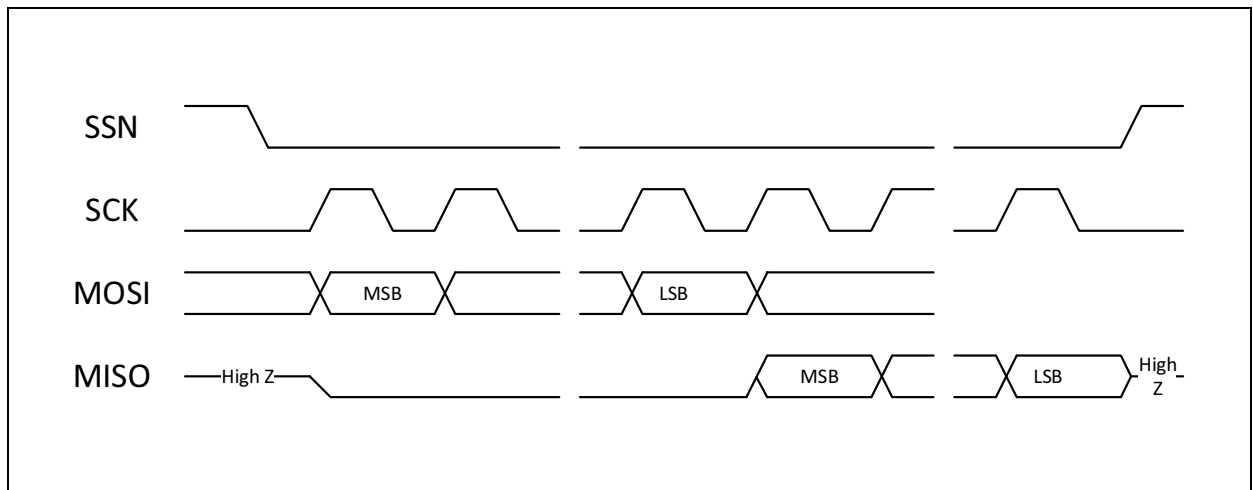


Figure 108:  
SPI Read



**Figure 109:**  
**SPI Timing**

| Name                                 | Symbol               | VDD=2.2V | VDD=3.0V | VDD=3.6V | Units |
|--------------------------------------|----------------------|----------|----------|----------|-------|
| Serial clock frequency               | $f_{\text{SPI-bus}}$ | 10       | 17       | 20       | MHz   |
| Serial clock pulse width HI state    | $t_{\text{pwh}}$     | 50       | 30       | 25       | ns    |
| Serial clock pulse width LO state    | $t_{\text{pwl}}$     | 50       | 30       | 25       | ns    |
| SSN enable-to-valid latch            | $t_{\text{susssn}}$  | 10       | 8        | 7        | ns    |
| SSN pulse width between write cycles | $t_{\text{pwssn}}$   | 50       | 30       | 25       | ns    |
| Data setup time prior to clock edge  | $t_{\text{sud}}$     | 7        | 6        | 5        | ns    |
| Data hold time after clock edge      | $t_{\text{hd}}$      | 5        | 4        | 3        | ns    |
| Data valid after clock edge          | $t_{\text{vd}}$      | 40       | 26       | 16       | ns    |

### ***GPIO and PDM/PWM***

This section is about the general purpose ports and their use as Pulse-Density / Pulse Width Modulated outputs (PDM/PWM). PCap04 is very flexible with assignment of the various GPIO pins to the DSP inputs/outputs. The following table shows the 6 general purpose ports and their possible assignment.

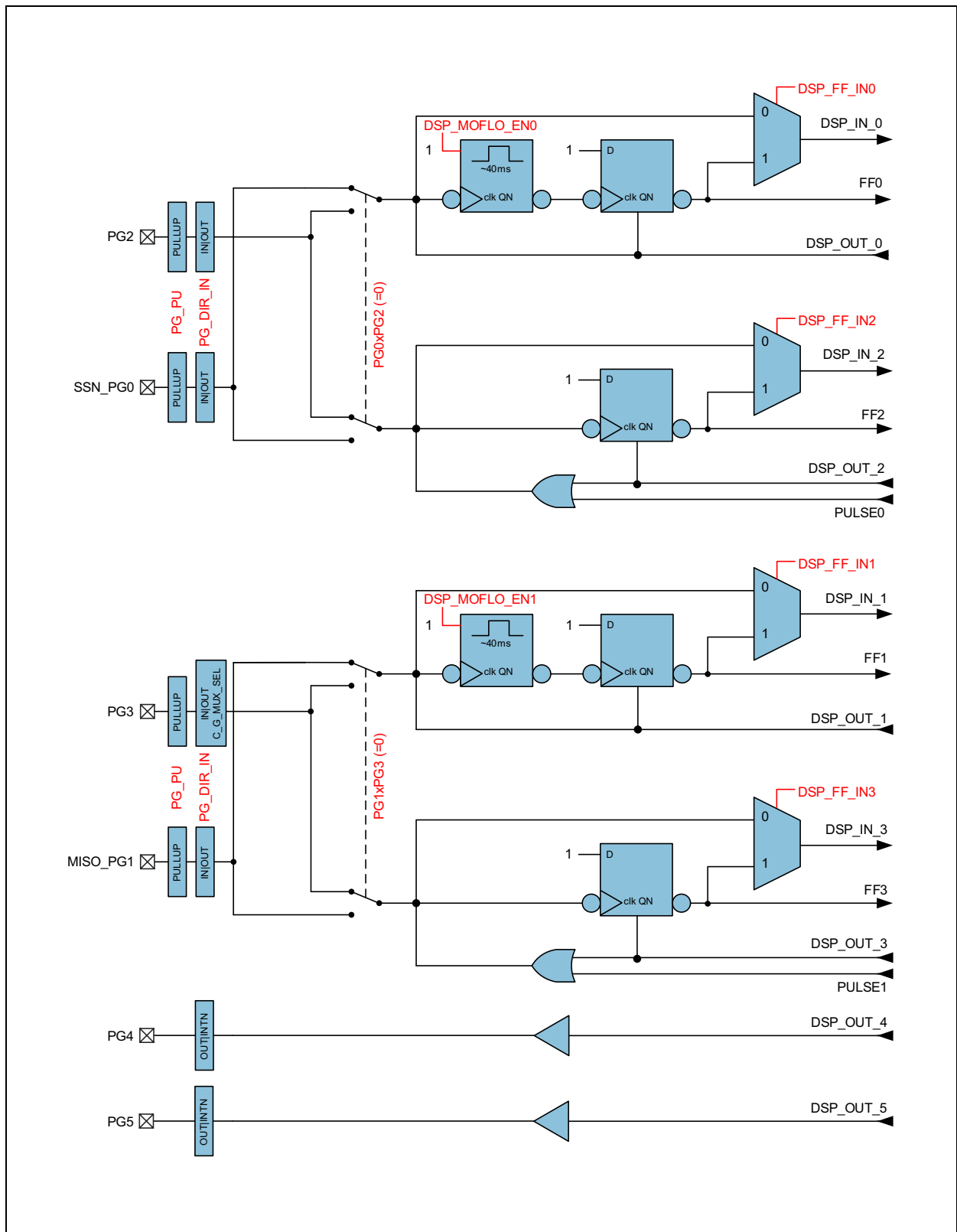
**Figure 110:**  
**General Purpose Port Assignment**

| External Port Name | Description                                | Direction In or Out     |
|--------------------|--|-------------------------|
| PG0                | SSN (in SPI-Mode), serial select           | In                      |
|                    | DSP0 or DSP2, I/O for the DSP              | In <sup>(1)</sup> / Out |
|                    | FF0 or FF2, I/O for the DSP with Flip-Flop | In <sup>(1)</sup>       |
|                    | Pulse0, PDM or PWM output                  | Out                     |
| PG1                | MISO (in SPI-Mode)                         | Out                     |
|                    | DSP1 or DSP3, I/O for the DSP              | In <sup>(1)</sup> / Out |
|                    | FF1 or FF3, I/O for the DSP with Flip-Flop | In <sup>(1)</sup>       |
|                    | Pulse1, PDM or PWM output                  | Out                     |
| PG2                | DSP0 or DSP2, I/O for the DSP              | In <sup>(1)</sup> / Out |
|                    | FF0 or FF2, I/O for the DSP with Flip-Flop | In <sup>(1)</sup>       |
|                    | Pulse0, PDM or PWM output                  | Out                     |
| PG3                | DSP1 or DSP3, I/O for the DSP              | In <sup>(1)</sup> / Out |
|                    | FF1 or FF3, I/O for the DSP with Flip-Flop | In <sup>(1)</sup>       |
|                    | Pulse1, PDM or PWM output                  | Out                     |
|                    | C_G_MUX_SEL output                         | Out                     |
| PG4                | DSP4 (output only)                         | Out                     |
|                    | INTN                                       | Out                     |
| PG5                | DSP5 (output only)                         | Out                     |
|                    | INTN                                       | Out                     |

**Note(s):**

1. These ports provide an optional debouncing filter and an optional pull-up resistor.

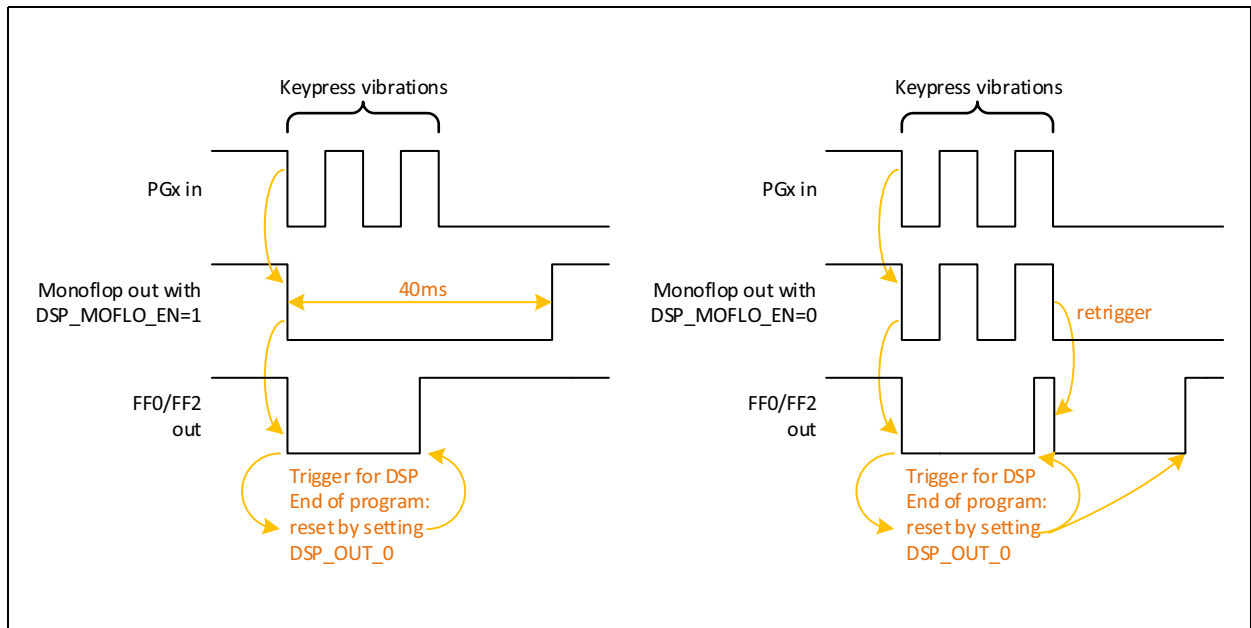
**Figure 111:**  
GPIO Assignment



**Debouncing Filter**

There is a possibility to activate a 40ms debounce filter (“monoflop”) for the ports in case these are used as push button inputs. This might be useful especially in case the DSP is started by the pins (signals FF0, FF2). Figure 112 shows the effect of the monoflop filter.

**Figure 112:**  
**Port Trigger Timing**



**PDM and PWM**

There is a possibility to generate two pulse density modulated output signals. In general, PDM is preferred because of better noise behavior. The output is based on the content of RAM registers PI0\_REF, PI1\_REF (DSP write addresses 109, 110. Width 16 bit each). The content of those RAM cells depends on the firmware. The description in this datasheet is based on the standard firmware, which writes the capacitance ratio to PI0\_REF, the Resistance ratio to PI1\_REF.

The pulse interfaces can be switched on individually. The resolution can be programmed from 10 to 16 bit. There is a broad range of clock signals that can be selected as base for the pulse interfaces, derived from the 50kHz low-frequency oscillator or the 2MHz internal oscillator. The output pins may be PG0 or PG2 and PG1 or PG3.

The PDM signal can be converted into an analog voltage by means of a simple RC-filter. A first-order filter made of 220kΩ / 100nF is sufficient. By the choice of the resistor and capacitor values the user can optimize for reaction time vs. ripple.

Filter configuration instructions:

The resistor should be ≥ 50kOhm. The internal DC resistance of the output buffer is typ. 100Ω

## 1. Settling time (for PDM and PWM)

If the output value changes, the settling time to reach 90% is  $2.3 \times \text{Tau}$ ,  $\text{Tau} = R \times C$

Example:  $200\text{k}\Omega \times 100\text{nF} \times 2.3 = 50\text{ms}$

The smaller is Tau the faster is the settling but the higher is the ripple.

## 2. Voltage Ripple

**Calculation Method:**

$$V_{DD} \cdot \left(1 - e^{-\frac{1}{f_0 \cdot R \cdot C}}\right) \text{ with } \left(f_0 \ll \frac{1}{R \cdot C}\right)$$

$$v_{pp} = \frac{V_{DD}}{f_0 \cdot R \cdot C}$$

$v_{pp}$  = ripple voltage (peak to peak)

$$f_0 = \text{for PWM} : \frac{1}{\text{period}} = f_{clk} / 2^{PWM\text{resolution|bit}}$$

$$\text{for PDM} : \frac{1}{t_{pulsewidth}}$$

In the standard firmware, the result of measurement from capacitance or temperature is a 32-bit value. The DSP linearizes this 32-bit result to a value according to the resolution settings of the pulse interface. The parameters  $pi\langle n \rangle\_result0$ ,  $pi\langle n \rangle\_result1$ ,  $pi\langle n \rangle\_pulse0$  and  $pi\langle n \rangle\_pulse1$  of the linear function are configurable in NVRAM, calibration space 800 to 822. The parameters are describing the edges for a simple scaling (1<sup>st</sup> order linearization), whereby  $pi\langle n \rangle\_pulse0$  is also the minimum clipping values and  $pi\langle n \rangle\_pulse1$  the maximum clipping value for the pulse output.  $pi\langle n \rangle\_pulse0$  must be always smaller than  $pi\langle n \rangle\_pulse1$ . For negative slopes just  $pi\langle n \rangle\_result0$  has to be larger than  $pi\langle n \rangle\_result1$ . A 12-bit resolution thus limits the result value between 0 and 4096. For lower-bit resolutions, the range reduces accordingly.

The  $pulse\_out$  is determined like this:

$$pi_n^{out} = \frac{pi_n^{pulse1} - pi_n^{pulse0}}{pi_n^{result1} - pi_n^{result0}} \cdot (result - pi_n^{result0}) + pi_n^{pulse0}$$

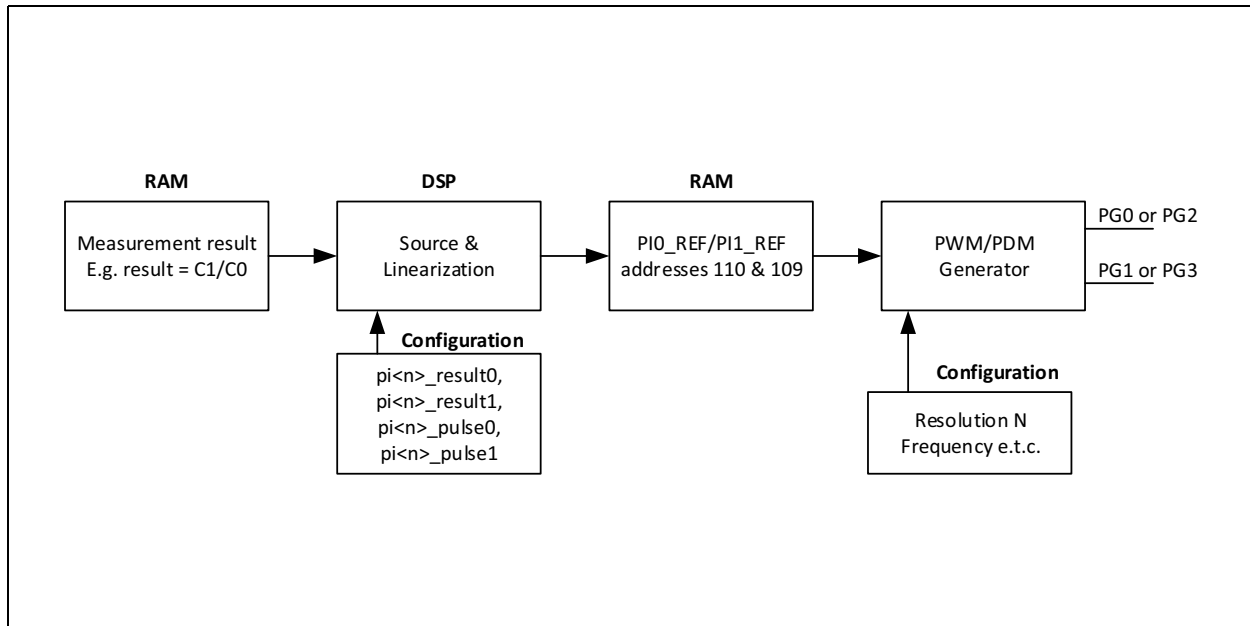
$$pi_n^{result1} > pi_n^{result0}$$

$$0 \leq pi_n^{pulse1} \leq 2^{pulse\ resolution}$$

$$0 \leq pi_n^{pulse0} \leq 2^{pulse\ resolution}$$

The following figure depicts how the result is processed to generate the pulsed output.

**Figure 113:**  
PDM and PWM Pulse Generation



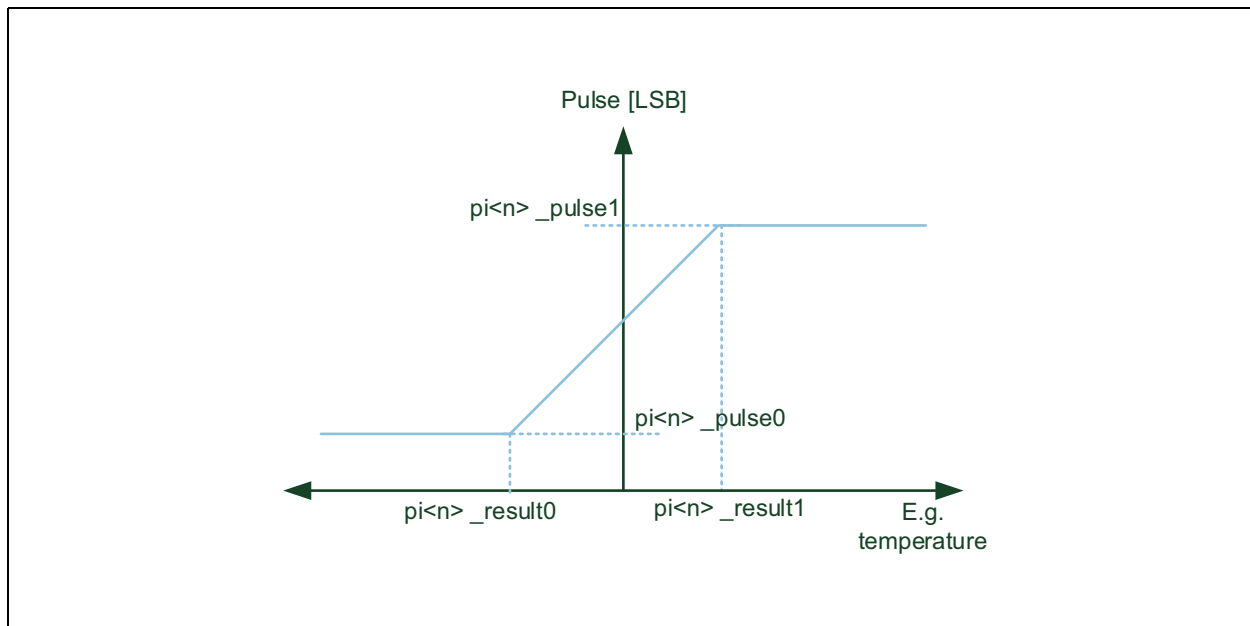
The following figure shows a sample linear function and its parameters graphically. In this graph, the result C1/C0 has been taken on the x-axis, assuming that this result is to be pulse modulated. A 12 bit resolution has been configured.

The settings for the PDM and PWM interface are made in configuration registers 27 and 29 to 33.

The lower limit (pi<n>\_pulse0) of the valid range corresponds to 0% modulation (all bits are 0), The upper limit (pi<n>\_pulse1) of the valid range corresponds to 100% modulation (all bits are 1), and this is the maximum possible value of output. 12 bit resolution implies that this maximum value is 4095. For lower-bit resolutions, this maximum value will come down accordingly. In terms of voltage, the two limits correspond to 0V and VDD.



**Figure 114:**  
**PDM and PWM Linearization**



#### Applications:

- A typical case would be outputting capacitance results through PG0 and temperature results through PG1. Calculation and transfer to the output registers will be performed by firmware.
- Main applications will be that an analog interface is demanded by the final customer.
- Applications where the serial interface cannot be used due to speed limitations or other reasons.
- Finally, a temperature-coded pulse stream could be low-pass filtered and then directly used for temperature control.

Please note that the entire linearization task as described here is performed by firmware, especially the PCap04\_standard and PCap04\_linearization firmware.

### Oscillators

PCap04 offers a low frequency oscillator (**OLF\_CLK**) and an integrated 2MHz high frequency oscillator (**OHF\_CLK**). **OLF\_CLK** is running all the time and cannot be turned off.

**OLF\_CLK** is used for:

- CDC cycle time
- RDC cycle time
- PDM/PWM time base
- Watchdog for Standalone Applications

The **OHF\_CLK** can be used alternatively for

- CDC cycle time
- PDM/PWM time base

The **OLF\_CLK** can be trimmed for various typical frequencies :

Figure 115:  
OLF Trimming

| OLF_CTUNE    | OLF_FTUNE | OLF Frequency |
|--------------|-----------|---------------|
| 3 : (10kHz)  | 1         | 5kHz          |
| 3 : (10kHz)  | 7         | 10kHz         |
| 2 : (50kHz)  | 0         | 28kHz         |
| 2 : (50kHz)  | 3         | 48kHz         |
| 1 : (100kHz) | 4         | 100kHz        |
| 0 : (200kHz) | 5         | 200kHz        |

**Note(s):** The internal oscillators are not very precise and stable. The frequency varies from chip to chip, with temperature and voltage.

- Variation over batch  $\pm 20\%$
- Variation with temperature  $\pm 5\%$ ,
- Variation with voltage.  $VDD \pm 2\%$

The **OHF\_CLK** can be switched off, turned on with delay before further tasks like measurement follow, or turned on continuously:

**OX\_RUN[2:0]0** : Generator off

6 : OX latency =  $1 / f_{OLF}$

3 : OX latency =  $2 / f_{OLF}$

2 : OX latency =  $31 / f_{OLF}$

1 : OX runs in permanence

By means of **OX\_DIV4** it can be divided by 4 to generate 500 kHz,

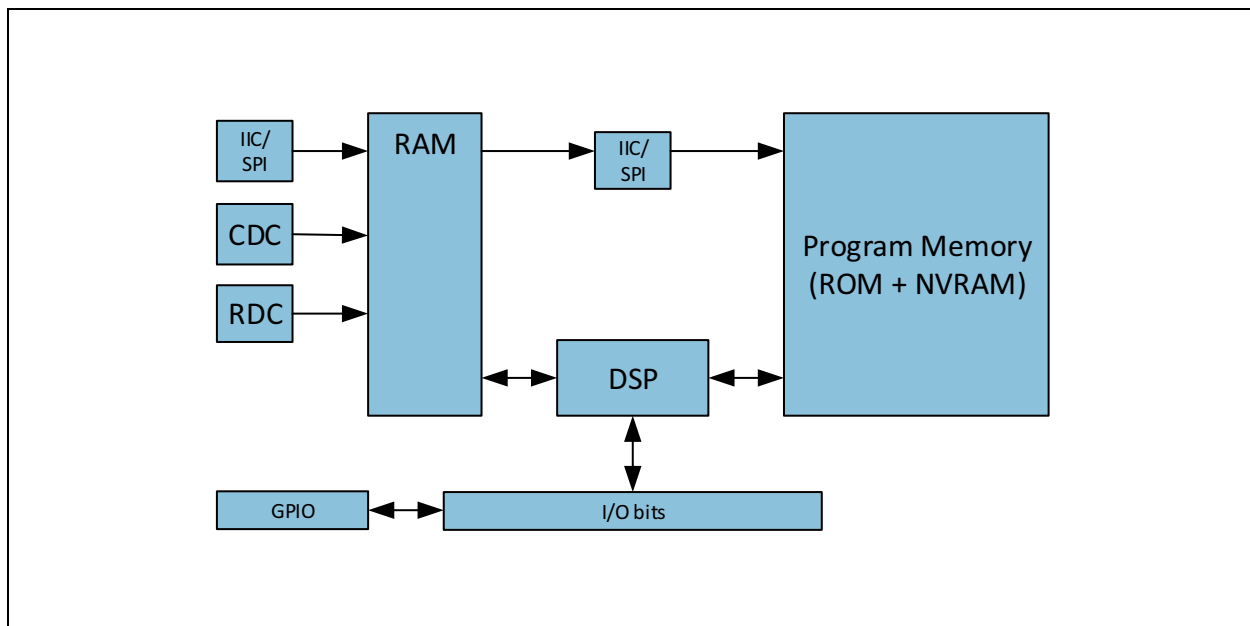
### DSP & Memory

This section describes the 32bit-DSP of the PCap04.

A 32-bit digital signal processor (DSP) in Harvard architecture was integrated to the PCap04. It is responsible for taking the information from the CDC and RDC measuring units, for processing the data and making them available to the user interface. Both, the CDC/RDC raw data as well as the data processed by the DSP are stored in the RAM. The program for the DSP is stored either in the NVRAM. The DSP can collect various status information from a set of 64 I/O Bits and write back 16 of those. This way the DSP can react on and also control the GPIO pins of PCap04. The DSP is internally clocked at approximately 60MHz. The internal clock is stopped through a firmware command, to save power. The DSP starts again upon a GPIO signal or an “end of measurement” condition.

In its simplest form, the DSP transfers the pure time measurement information from the CDC/RDC to the read registers without any further processing. The next higher step is to calculate the capacitance ratios including the information from the compensation measurements, as it is provided in **ams**’ standard firmware version PCap04\_standard\_v01.hex. Finally, **ams** provides a ready-made linearize firmware that performs a linearization via polynomial of third degree and temperature compensation via polynomial of second degree. Many functional blocks for the linearization firmware are implemented as ROM code. This way, the main firmware can be very compact and can fit into the 1k NVRAM.

**Figure 116:**  
DSP Embedding



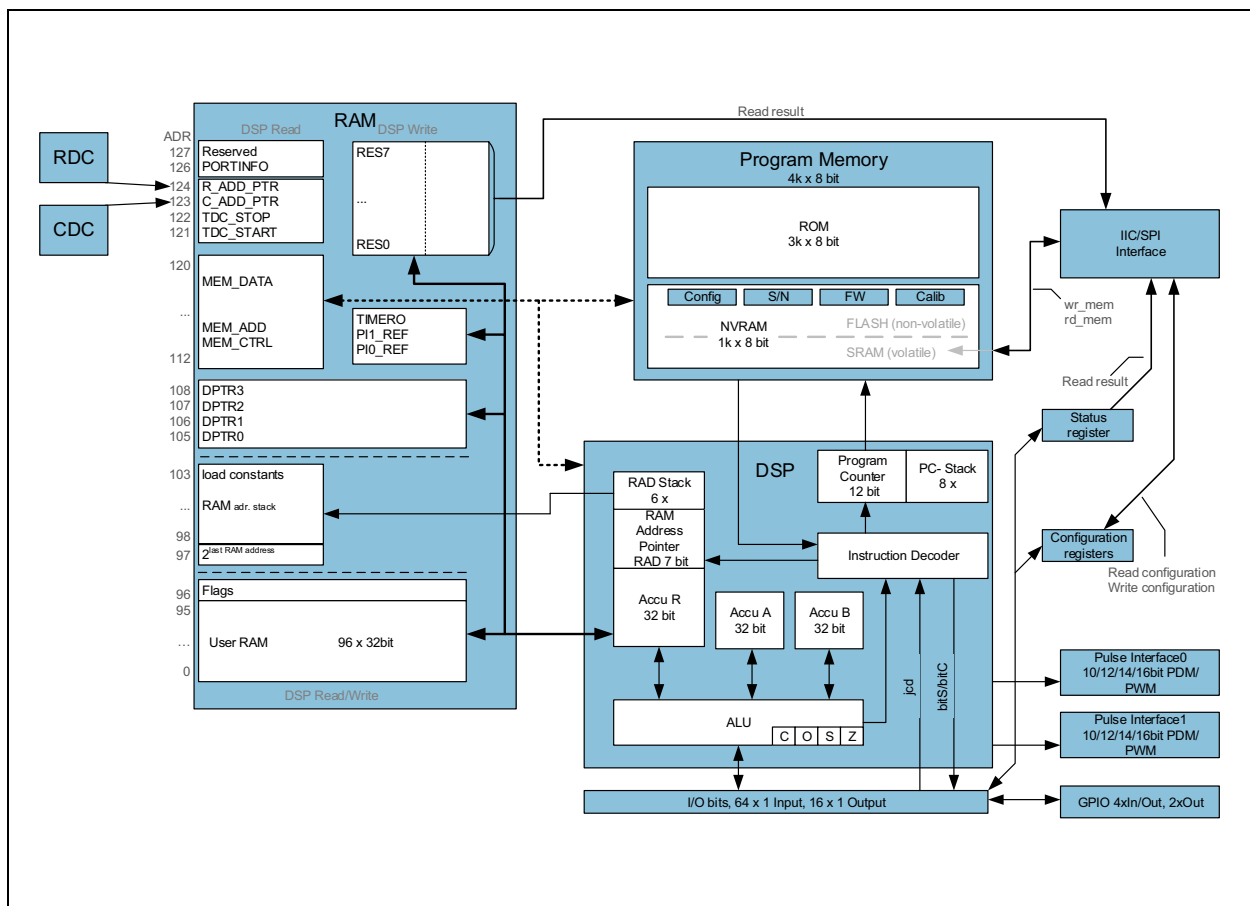
The content of the read registers will always depend on the firmware in use. With the standard firmware it will be the pure capacitance and resistance ratios. With the linearization firmware it might be the linearized and calibrated result, e.g. a pressure given in Pascal or humidity given in percent.

The DSP is **ams** proprietary to cover low-power tasks as well as very high data rates. It is programmed in Assembler. A user-friendly assembler software with a graphical interface, help text pop-ups as well as sample code sustain programming efforts.

### DSP & Environment

The DSP reads the RDC and CDC raw data from the RAM, processes them and writes the results back to the RAM. The program is stored in the NVRAM. It may use subroutines that are available from the ROM. The DSP reacts on flags and controls flags. It controls the GPIO and accordingly the PDM/PWM interface.

Figure 117:  
DSP Environment



The DSP is designed in Harvard architecture for 32 bit wide parallel data processing. It is coupled to a 128 x 32 bit RAM, 96 x 32 bit thereof freely accessible. In read access, the DSP can get the MEM\_DATA from address space 112 to 120, the CDC- and

RDC-frontend control from address space 121 to 124. By write access the DSP provides the output data to the PDM/PWM interfaces (addresses 109, 110).

A detailed description of the RAM is given in section 2.1. The DSP operates with two accumulators A and B and has direct access to the RAM, which can be seen as a third accumulator. The RAM address pointer is of 7 bit size, and there is a 6-fold stack for RAM addresses.

The program counter has 12 bit and there is an 8-fold stack for the program counter.

Finally, the DSP can get a lot of information from the 64 I/O bits. The read information covers the ALU status, trigger information, some of the configuration bits and the information about the status of the GPIOs. 16 of those bits can be used as outputs, setting the GPIOs and also some internal information. The DSP can read these bits by means of instruction jcd (conditional jump) and set those bits by means of instructions bitS/bitC (bit Set/Clear).

The ALU flags overflow, carry, equal/not equal and pos./neg. are used directly as condition for the jcd instructions and are also mirrored in the I/O bits.

### **RAM Structure**

The RAM plays a key role. It is made of 128 words with size of maximum 32 bit. The DSP has free write and read access to registers address 0 to 96 of those words, all 32 bits wide. The RAM space addresses 97 to 104, 109 to 111 and 115 and higher has different meaning for read and write operations.

The main data in the read section are the raw data as they come from the CDC and the RDC. Also the parameters are in the RAM as part of the configuration registers and they are set via the serial interface or copied from the NVRAM.

The DSP reads the raw data, does the data processing and writes back the results into the write section of the RAM. From there, the user can read the final results through the serial interface.

Some of the RAM cells are dedicated to special functions and will be described in the following in detail.

**Figure 118:**  
**RAM Structure in Detail**

| RAM: DSP Read |                          |      | RAM: DSP Write |             |      |
|---------------|--------------------------|------|----------------|-------------|------|
| Addr          | Description              | Bits | Addr           | Description | Bits |
| 127           | Reserved                 |      |                |             | -    |
| 126           | PORTINFO                 | 24   |                |             |      |
| 124           | R_ADD_PTR                | 2    |                |             |      |
| 123           | C_ADD_PTR                | 4    |                |             |      |
| 122           | TDC_STOP                 | 27   |                |             |      |
| 121           | TDC_START                | 27   |                |             |      |
| 120           | MEM_DATA_u08b            | 32   |                |             |      |
| 119           | MEM_DATA_u16b            | 32   |                |             |      |
| 118           | MEM_DATA_u24b            | 32   |                |             |      |
| 117           | MEM_DATA_s08b            | 32   |                |             |      |
| 116           | MEM_DATA_s16b            | 32   |                |             |      |
| 115           | MEM_DATA_s24b            | 32   |                |             |      |
| 114           | MEM_DATA (wr & 4Byte rd) |      |                |             | 32   |
| 113           | MEM_ADD                  |      |                |             | 10   |
| 112           | MEM_CTRL                 |      |                |             | 16   |
|               |                          |      | 111            | TIMERO      | 16   |
|               |                          |      | 110            | PI1_REF     | 16   |
|               |                          |      | 109            | PIO_REF     | 16   |
| 108           | DPTR3                    |      |                |             | 7    |
| 107           | DPTR2                    |      |                |             | 7    |
| 106           | DPTR1                    |      |                |             | 7    |
| 105           | DPTR0                    |      |                |             | 7    |
|               |                          |      | 104            | RES7        | 32   |
| 103           | rad_stack_6b             | 6    | 103            | RES6        | 32   |
| 102           | rad_stack_12b            | 12   | 102            | RES5        | 32   |
| 101           | rad_stack_18b            | 18   | 101            | RES4        | 32   |
| 100           | rad_stack_24b            | 24   | 100            | RES3        | 32   |

| RAM: DSP Read |                               |      | RAM: DSP Write |             |      |
|---------------|-------------------------------|------|----------------|-------------|------|
| Addr          | Description                   | Bits | Addr           | Description | Bits |
| 99            | rad_stack_30b                 | 30   | 99             | RES2        | 32   |
| 98            | rad_stack_32b                 | 32   | 98             | RES1        | 32   |
| 97            | 2 <sup>last_RAM_address</sup> | 32   | 97             | RES0        | 32   |
| 96            | Flags & extended GPIO's       |      |                |             | 32   |
| 95            | (free) User RAM               |      |                |             | 32   |
| ...           | ...                           |      |                |             | 32   |
| 0             | (free) User RAM               |      |                |             | 32   |

**Registers 0 to 95, User RAM**

This is normal RAM space without any special functions. It is readable and writable via instruction rad.

Example:

Add content of RAM address 12 and 13 and write the result into RAM address 13

```
rad 12
move a, r
rad 13
add r, a
```

Register 96, Flags & Internal Control Signals

Figure 119:  
Flags

| Bit    | Flag Name         | Default (after Reset) | Description   |
|--------|-------------------|-----------------------|---|
| 0      | FIRSTSTART_N      | 0                     | Indicates first DSP-Trigger until set to 1 by firmware  |
| 1..2   | Free to use       | 0                     |   |
| 3      | RDCHG_COM_INT_SEL | 0                     | 0 : Use RDCHG_IN_SEL0<br>1 : Use RDCHG_IN_SEL1<br><br>For internal compensation   |
| 4      | Free to use       | 0                     |   |
| 5      | FLAG_CDC_INV      | 0                     | Temporary parameter to ROM routine _ROM_cdc_<br>0 : Determine inverse ratios reference/sense<br>1 : Determine ratios sense/reference  |
|        | SIGNED_VALUE_NV   | 0                     | Temporary parameter to ROM routines<br>_ROM_NVblock_copy_32b_<br>_ROM_NVblock_copy_24b_<br>_ROM_NVblock_copy_16b_<br>_ROM_NVblock_copy_08b_<br>0 : Treat data from NVRAM as unsigned<br>1 : Treat data from NVRAM as signed |
| 6..7   | Reserved          | 0                     | Temporarily used flags within ROM routines  |
| 8      | RST_RDC           | Pulsed                | Temperature reset. This flag has to be set 1, after each RDC measurement. Otherwise a new RDC measurement is not possible. This flag is set back to 0 automatically   |
| 9..15  | Reserved          |                       |   |
| 16..31 | Free to use       | Unknown               |   |

DSP Read Register 97

This register is there to get the N-th power of 2. The exponent N needs to be written to the RAD stack. The result can be read from register 81. In the assembler, the necessary three instructions are merged into one:

load2exp a, 10 ; a = 2<sup>10</sup> = 1024

A very simple and efficient method to set an accumulator = 1 is

load2exp b, 0 ; b = 2<sup>0</sup> = 1



**DSP Read Registers 98 to 103**

These registers contain the content of the RAM address stack. The 32 bit data is made of the 6 last 6-bit RAM addresses. This address can be used to load 32 bit constants from the program memory into the data space. The necessary instructions are merged into one single instruction by the assembler. (Hint: The assembler accepts negative values as well as decimal and hexadecimal numbers. Depending on the constant to be load, the assembler converts this instruction into 3 to 8 operations)

load a, 1715956 ; a = 1715956

is the same as

rad 0x06 ; 0x06 \* 2<sup>18</sup>

rad 0x22 ;+ 0x22 \* 2<sup>12</sup>

rad 0x3b ;+ 0x3b \* 2<sup>6</sup>

rad 0x34 ;+ 0x34 = 1715956

rad 100 ; rad\_stack\_24b

move a, r

**DSP Read/Write Registers 105 to 108, Data Pointer**

These registers may be used for indirect addressing. They are 7 bits wide.

Load a register with the address you want to manipulate:

|   |   |
|---|---|
| load a, <myaddress><br>rad DPTR0<br>move r, a | load a, <myaddress><br>rad 105<br>move r, a |
|---|---|

Load a RAM address pointer with content of DPTR0:

rad \_at\_DPTR0; now ram address pointer is set to content of DPTR0

**Hint:** In the <pcap\_standard.h> "\_at\_DPTR0" to "\_at\_DPTR3" are set to values of 284 to 287. These are no valid RAM addresses but just indicators to the assembler to generate the corresponding opcodes.

Example direct memory address: Copy a memory block from one address to another:

|  |  |  |
|--|--|--|
| <pre> __sub_dma__: not b inc b __sub_dma_loop__: rad _at_DPTR1 move a, r rad _at_DPTR0 move r, a  rad DPTR0 inc r rad DPTR1 inc r inc b  jne __sub_dma_loop__ jrt         </pre> | <pre> ; initialize loop counter ; with - &lt;length&gt; ; ; copy a : @DPTR1  ; copy @DPTR0 : a  ; increment destination ; address ; increment source add ; ; increment loop ; counter ; loop body         </pre> | <pre> ; DPTR1 : source_address; DPTR0 : destination address; b: length of dma         </pre> |
|--|--|--|

**DSP Read Register 126, PORTINFO (PORTERR<7...0>, PORTMASK<7...0>)**

The low 8 bits mirror the port enable setting as defined by configuration parameter C\_PORT\_EN in register 12.

Bits 8 to 17 are error flags for the capacitance ports including the internal reference ports.

**DSP Write Registers 97 to 104, RES00...RES07**

These are the result registers to which the DSP has to write the output data so that the user can read those through the SPI/IIC interface as Res 0 to Res 7.

All addresses are 32 bit wide.

**!!! Attention:** These Registers are write only! The DSP cannot read from these Registers !!!

**DSP Write Registers 109, 110, PIO\_REF...PI1\_REF**

These registers contain the data that is used to generate the PWM/PDM output signals. After the DSP has calculated and scaled the output data, it writes those into these two registers. The data are 16 bit wide.

**DSP Write Register 111, TIMERO**

The DSP has a 16-bit timer based on the OLF clock. This timer may be used to generate long delays while the DSP is halted. Bit #3 (timer) in DSP\_START\_EN must be set!

By writing a value to Register 111 the timer starts to count up from 0 each OLF-clock cycle until the written value has been reached. Then a DSP\_START\_TRIG is generated.

If the DSP is not halted the TIMERO\_IRQ\_N Flag could be tested anyway.

## Example 1 (without halting DSP):

```

CONST    wait_time_1ms 50      ; 50*20µs (@50kHz)
...
load a, wait_time_1ms
rad TIMER0
move r, a

timer_wait_loop:
jcd TIMER0_IRQ_N, timer_wait_loop

```

## Example 2 (with halting DSP, DSP run on internal oscillator):

```

CONST    wait_time_1ms 50      ; 50*20µs (@50kHz)
...
ORG 0
jcd TIMER0_IRQ_N, Skip_Timer0_process
    jsb Triggered_by_Timer0
Skip_Timer0_process:
...
load a, wait_time
rad TIMER0
move r, a
stop

Triggered_by_Timer0:      ; subroutine
...

```

**DSP Read/Write Registers 112 to 120, MEM\_CTRL, MEM\_ADD, MEM\_DATA**

Those registers are used by ROM routines for the transfer of data between NVRAM and RAM. It is possible to transfer data between NVRAM and accumulators a and b, from 1 to 4 bytes, signed and unsigned.

- MEM\_CTRL: defines the operation. The four options are
  - MEM\_STORE: write to NVRAM
  - MEM\_RECALL: read from NVRAM
  - MEM\_WE: enable writing
  - MEM\_WR\_PROTECT: protect against arbitrary writing
- MEM\_ADD: defines the target address in the NVRAM
- MEM\_DATA\_xxx: registers for write or read data. Data that shall be written into NVRAM data need to be in RAM address 114. Data that have been read from NVRAM can be found in addresses 114 to 120, depending on the format.

These calls are used e.g. to copy constants and calibration data from the NVRAM to the RAM.

Example: Copying the NV\_C\_sens\_sel register to RAM

```
load    a, NV_C_sens_sel    ; mem_add : NV_C_sens_sel
rad     mem_add
move    r, a
jsb     __ROM_memory_rd_a_u08b__ ; A contains
content of NV_C_sens_sel
rad     RAM_C_sens_sel
move    r, a
```

These registers may be used by the DSP to change the configuration on the fly. Important note: After writing to the NVRAM it is necessary to do an ini\_reset. Therefore the DSP has to do the following sequence of setting flags DSP\_6 and DSP\_7.

```
; Initreset
```

```
    bitC 7
    bitC 6
    bitS 6
    bitC 6
    bitS 6
```

*DSP Read Registers 121 to 124, TDC\_START, TDC\_STOP, C\_ADD\_PTR, R\_ADD\_PTR*

**ams** internal data, used by ROM routine \_\_tdc\_dispatch\_\_

### **NVRAM and ROM**

The total program memory is made of 1k NVRAM and 3k ROM. The NVRAM holds configuration data, 960 byte of user code and some special registers. The ROM holds useful mathematical routines that make programming very efficient.

### **NVRAM Structure**

The user space is split in three sections. The reason is that the NVRAM can be read/write protected by section. A big section of 704 byte is for program code, two smaller sections of 128 byte may be used for calibration data or additional firmware.

**Figure 120:  
NVRAM Organization**

| Address      |             | NVRAM (1k x 8 bit)     |               | Memory Lock |
|--------------|-------------|------------------------|---------------|-------------|
| Decimal      | Hexadecimal | Contents               | Length [Byte] | Settings    |
| 1023 to 1022 | 3FF to 3FE  | CHARGE_PUMP            | 2             | MEM_LOCK<3> |
| 1021 to 1011 | 3FD to 3F3  | Reserved               | 11            |             |
| 1010 to 1009 | 3F2 to 3F1  | S/N customer           | 2             |             |
| 1008         | 3F0         | MEM_LOCK               | 1             |             |
| 1007 to 960  | 3EF to 3C0  | Configuration Registry | 48            | MEM_LOCK<3> |
| 959 to 832   | 3BF to 340  | User Space (FW/CAL1)   | 128           | MEM_LOCK<2> |
| 831 to 704   | 33F to 2C0  | User Space (FW/CAL0)   | 128           | MEM_LOCK<1> |
| 703 to 0     | 2BF to 0    | User Space (FW)        | 704           | MEM_LOCK<0> |

The NVRAM consists of two parts: a volatile SRAM and a non-volatile memory (FLASH). There is a store/recall method to copy (store) the complete SRAM content to FLASH or to recall it from FLASH back to SRAM.

Different methods of operation apply:

- **Stand Alone:**  
Configuration data, firmware and calibration values are stored once to non-volatile memory and autoboot is selected. After a power-on the device starts immediately with the measurement.
- **Pre-Configured:**  
Configuration data, firmware and calibration values are stored once to nonvolatile memory, RUNBIT and autoboot are disabled. After power-on, the device is programmed and configured, but in idle mode, waiting for instructions.
- **Pure Slave:**  
Configuration data, firmware and calibration values are written to the SRAM (volatile memory) after each power on by an external  $\mu$ C.

### *NVRAM Access*

There are three commands available to handle the NVRAM, Store, Recall & Erase, each one protected to avoid accidental trigger during communication over the serial interface. It is mandatory to send first an activation code to register 54 (MEM\_CTRL). This is followed by the according opcode (|| = termination of SIF, e.g. setting SSN = HIGH):

Store SRAM content into NVRAM: Activation code in MEM\_CTRL: 0x2D  
 Store NVRAM opcode: 0x96  
 (Send via SIF: 0xA3F62D || 0x96, wait minimum 12 ms)

Recall from NVRAM into SRAM: Activation in MEM\_CTRL: 0x59  
 Recall from NVRAM opcode: 0x99  
 (Send via SIF: 0xA3F659 || 0x99)

Erase NVRAM: Read trim bits adr 1022&1023 and Unique ID from adr 954 to 959  
 Activation in MEM\_CTR: 0xB8  
 Erase NVRAM opcode: 0x9C  
 (Send via SIF: 0xA3F6B8 || 0x9C, wait minimum 12ms)  
 Write back trim bits adr 1022&1023 and Unique ID to adr 954 to 959  
 Activation code in MEM\_CTRL: 0x2D  
 Store NVRAM opcode: 0x96

**Important Note:** We guarantee the data for data retention and endurance only under the assumption, that the customer does **not** change the registers 62 and 63 and NVRAM adr 954 to 959 (Unique ID). In addition, it is mandatory to follow the given procedure for ERASE NVRAM as described in section [NVRAM and ROM](#) precisely. Otherwise, **we do no longer guarantee** the data retention time and endurance cycles.

### *ROM Structure*

The limitation in size for the NVRAM is compensated by having many functions integrated hard-wired in 3k ROM. The ROM routines range from simple shift functions over filters to polynomial linearization of 4<sup>th</sup> degree. This allows to keep the user code very compact.

The assembler comes with header file PCap04\_ROM\_addresses\_standard.h that lists the jump-in addresses for the various ROM routines. For details see section Sample Code / Libraries.

### *DSP Inputs & Outputs*

The DSP has access to 64 bits of information on ALU status, start trigger, configuration, input / output pins.

This information can be interpreted by means of instructions `jcd` or conditional jump.

Instruction conditional jump looks like:

`jcd p1,p2: if p1 ==1 then jump to p2, p1 = flag number`

16 of those bits can be set by the DSP, e.g. to set a GPIO or to select between RDC and CDC data. The bits are controlled by means of instructions `bitS` / `bitC` (bit Set/bit Clear).

**Figure 121:**  
DSP Inputs / Outputs

| Bit Name                         | Description  | Type          | Read Bit # | Write Bit # |
|----------------------------------|--|---------------|------------|-------------|
| DSP_OUT<7...0>                   | Status feedback of the 8 general DSP outputs (Write bits 0 to 7).  | IN            | 56 to 63   |             |
| SIF_TRIGGERED_N <sup>(1)</sup>   | Flag = LOW indicates that a falling edge at a pin or an SPI/IIC opcode has started the DSP. This flag is reset by a STOP instruction at the end of the firmware.   | Start trigger | 55         |             |
| PIN_TRIGGERED_N <sup>(1)</sup>   | Flag = LOW indicates a GPIO has started the DSP  |               | 54         |             |
|                                  |  |               |            |             |
| TDC_TRIGGERED_R_N <sup>(1)</sup> | Flag = Low indicates that a single time-values from Resistance (Temperature) measurement are available and must be processed (done by ROM routine <code>_ROM_tdc_dispatch__</code> )                                     |               |            |             |
| TDC_TRIGGERED_C_N <sup>(1)</sup> | Flag = LOW indicates that time-values from Capacitance measurement are available and must be processed (done by ROM routine <code>_ROM_tdc_dispatch__</code> )   | Start trigger | 52         |             |
| INTN_TRIGGERED_N <sup>(1)</sup>  | Flag = LOW indicates the DSP is started by rising edge of INTN-Signal  | Start trigger | 51         |             |
| TIMERO_IRQ_N <sup>(1)</sup>      | Flag = LOW indicates the DSP is started by the internal timer  | Start trigger | 50         |             |
| RDC_TRIGGERED_N <sup>(1)</sup>   | Flag = LOW indicates that an RDC measurement has started the DSP. Therefore, <code>DSP_STARTONTEMP</code> has to be set (configuration register 8). This flag is reset by a STOP instruction at the end of the firmware. | Start trigger | 49         |             |
| CDC_TRIGGERED_N <sup>(1)</sup>   | Indicates the DSP is started by the end of the capacitance conversion.   | Start trigger | 48         |             |

| Bit Name            | Description  | Type          | Read Bit # | Write Bit # |
|---------------------|--|---------------|------------|-------------|
| ALU_OFL_N           | ALU flags for overflow, carry, equal and sign. The ALU flags are used by the jump instruction of the assembler   | Status        | 47         |             |
| ALU_OFL             |  | Status        | 46         |             |
| ALU_CAR_N           |  | Status        | 45         |             |
| ALU_CAR             |  | Status        | 44         |             |
| ALU_EQ / ALU_ZERO   |  | Status        | 43         |             |
| ALU_NE / ALU_ZERO_N |  | Status        | 42         |             |
| ALU_POS             |  | Status        | 41         |             |
| ALU_NEG             |  | Status        | 40         |             |
| FLAGREG_N[7:0]      | Lower 8bits, inverted Flags from FLAGREG (register 96)   | Flag          | 32..39     |             |
| AWAKE_TRIGGERED_N   | After setting RUNBIT to 1 the DSP is triggered immediately. This flag shows this trigger source. Used for initialize raw result registers before first measurement (used by ROM routine _ROM_tdc_dispatch__) | Start Trigger | 31         |             |
|                     |  |               |            |             |
| TDC_RDY             | Flag = Low indicates TDC-Ring oscillator is running  | Status        | 28         |             |
| POR_CDC_DSP_COLL    | Flag = Low indicates reset forced by a CDC / DSP collision   | Status        | 27         |             |
| LAST_CYCLE_ACTIVE_N | Flag = Low indicates that this is the last CDC measurement in the current sequence (used for ROM routine _ROM_tdc_dispatch__)  | Status        | 26         |             |
|                     |  |               |            |             |
| CYC_ACTIVE          | Flag = bit 23 of status register. Indicates that the CDC frontend is active. (not negated)   | Status        | 25         |             |
| POR_FLAG_WD         | Flag = Low indicates a reset was forced by watchdog timeout  | Status        | 24         |             |
| POR_FLAG_PARITY     | Flag = Low indicates a reset was forced by one or more configuration bits toggled by interferences.  | Status        | 23         |             |
| CONTINUOUS_N        | Low : Continuous mode is activated   | Config Reg    | 22         |             |
| AUTOSTART_N         | Bit from configuration register  | Config Reg    | 21         |             |
| C_REF_INT           | Bit from configuration register  | Config Reg    | 20         |             |



| Bit Name                    | Description   | Type       | Read Bit # | Write Bit # |
|-----------------------------|---|------------|------------|-------------|
| TIMER_TRIG_DSP              |   |            | 19         |             |
| (TRUE)                      | Constant 1, usable for "goto" jcd TRUE, <jump_address>  |            | 18         |             |
| INT_TRIG_BG_N               | Bit from configuration register   | Config Reg | 17         |             |
| CDC_TRIG_BG_N               | Bit from configuration register   | Config Reg | 16         |             |
| C_COMP_EXT_N                | Bit from configuration register   | Config Reg | 15         |             |
| C_COMP_IN_N                 | Bit from configuration register   | Config Reg | 14         |             |
| C_SINGLE / C_DIFFERENTIAL_N | Bit from configuration register   | Config Reg | 13         |             |
| C_GROUNDED / C_FLOATING_N   | Bit from configuration register   | Config Reg | 12         |             |
| ERR_OVFLN                   | Flag = bit 16 of status register. Indicates an overflow or other error in the TDC.  | Status     | 11         |             |
| COMB_ERRN                   | Flag = bit 16 of status register. This is a combined condition of all known error conditions.   | Status     | 10         |             |
| CYC_ACTIVE_N                | Flag = bit 23 of status register. Indicates that the CDC frontend is active. (negated)  | Status     | 9          |             |
| SIF_RES_RD_BSY              |   | Status     | 8          |             |
| RAM_BUSY                    | Indicates, NVRAM is busy  | Status     | 7          |             |
| Interrupt_In                | Port INTN will be reset by a positive edge on SSN (SPI) or a stop condition (I <sup>2</sup> C), with this flag the current status of INTN can be detected | Status     | 6          |             |
| TEMPERR_N                   | Flag = bit 3 of status register 1. Indicates whether an error occurred during the temperature measurement.<br>0 : Error, 1 : No error                     | Status     | 5          |             |
| RDC_BUSY                    | Flag = bit 2 of status register. Indicates RDC unit is busy.<br>0 : Measurement done, 1 : Measurement running.  | Status     | 4          |             |
| TRIG_BG                     | This parameter starts the Bandgap (to synchronize with measurement) (pulse, automatically set to 0)   | Out        |            | 15          |
| (MEM_PUSH)                  | Reserved, only usable by ROM routines   | Out        |            | 14          |

| Bit Name      | Description  | Type   | Read Bit # | Write Bit # |
|---------------|--|--------|------------|-------------|
| RST_CDC       | CDC reset. This flag has to be set 1, after each CDC measurement. Otherwise, a new CDC measurement is not possible. This flag is set back to 0 automatically   | Out    |            | 13          |
| (MEM_RD)      | Reserved, only usable by ROM routines  | Out    |            | 12          |
| Interrupt_Out | Sets the interrupt (pin PG4 or PG5, see register 30) (pulse, automatically set to 0)   | Out    |            | 11          |
| (PAGE)        | Reserved, do not use   | Out    |            | 10          |
| TRIG_RDC      | This bit starts a new RDC measurement. (pulsed, automatically set to 0)  | Out    |            | 9           |
| TRIG_CDC      | This bit starts a new CDC measurement (pulsed, automatically set to 0)   | Out    |            | 8           |
| DSP_7         | Those two outputs are used by the DSP for<br>- Reset watchdog<br>- INI_RESET by DSP<br>(Pattern combination of both Outputs are used to prevent these actions triggered accidentally)<br>; Inireset          | Out    |            | 7           |
| DSP_6         | bitC 7<br>bitC 6<br>bitS 6<br>bitC 6<br>bitS 6   | Out    |            | 6           |
| DSP_5         | Sets the general purpose output pin PG5  | Out    |            | 5           |
| DSP_4         | Sets the general purpose output pin PG4  | Out    |            | 4           |
| DSP_3         | When the Pulse1 is switched OFF then this bit can be used to set and clear the general purpose output pin PG3. When the Pulse1 is ON then this bit must be cleared so that the Pulse1 output appears on PG3. | In/Out | 3          | 3           |
| DSP_2         | When the Pulse0 is switched OFF then this bit can be used to set and clear the general purpose output pin PG2. When the Pulse0 is ON then this bit must be cleared so that the Pulse0 output appears on PG2  | In/Out | 2          | 2           |
| DSP_1         | Set or read the general purpose I/Os at pins PG0 & PG1. The assignment is programmable and shown in detail below.  | In/Out | 1          | 1           |
| DSP_0         |  | In/Out | 0          | 0           |

**Note(s):**

1. A negative edge on those inputs start the DSP. The status of the start trigger is memorized till the next reset or stop of the DSP. The start trigger information can be read from inputs 48 to 55 by jcd.

### ALU Flags

Every ALU operation sets flags. The ALU has four flags: overflow, carry, equal and sign. The following table shows an overview:

**Figure 122:**  
**ALU Flags**

| Flag | Description             | Format            | Modified by Instructions:                 | Interpreted by Instructions: | Range                                |
|------|-------------------------|-------------------|---|------------------------------|--------------------------------------|
| ON   | No Overflow             | signed            | add, sub, mult, div                       | jOvIC, jOvIS                 | $\geq -2^{31}$ and $\leq 2^{31} - 1$ |
| O    | Overflow                |                   |   |                              | $< -2^{31}$ and $> 2^{31} - 1$       |
| CN   | No Carry <sup>(1)</sup> | unsigned          | add, sub, mult, div                       | jCarC, jCarS                 | $< 2^{32}$                           |
| C    | Carry <sup>(1)</sup>    |                   |   |                              | $\geq 2^{32}$                        |
| Z    | Equal / Zero            | signed / unsigned | add, sub, mult, div, move, shiftL, shiftR | jEQ, jNE                     | $= 0$                                |
| ZN   | Not Equal / Not Zero    |                   |   |                              | $\neq 0$                             |
| S    | Positive                | signed            | add, sub, mult, div, move, shiftL, shiftR | jPos, jNeg                   | $\geq 0$                             |
| SN   | Negative                |                   |   |                              | $< 0$                                |

**Note(s):**

1. During addition, the carry C is set when a carry-over takes place from the most significant bit, else C remains at 0. During subtraction, carry C is by default 1. Carry C is cleared only when the minuend < subtrahend.

E.g. for  $A - B$ : if  $A \geq B \rightarrow C = 1$ ; if  $A < B \rightarrow C = 0$ .

In other words, the carry C is actually the status of the carry of the addition operation  $A + 2$ 's complement (B).

**DSPOUT – GPIO Assignment**

PCap04 is very flexible with assignment of the various GPIO pins to the DSP inputs/outputs. The following table shows the possible combinations.

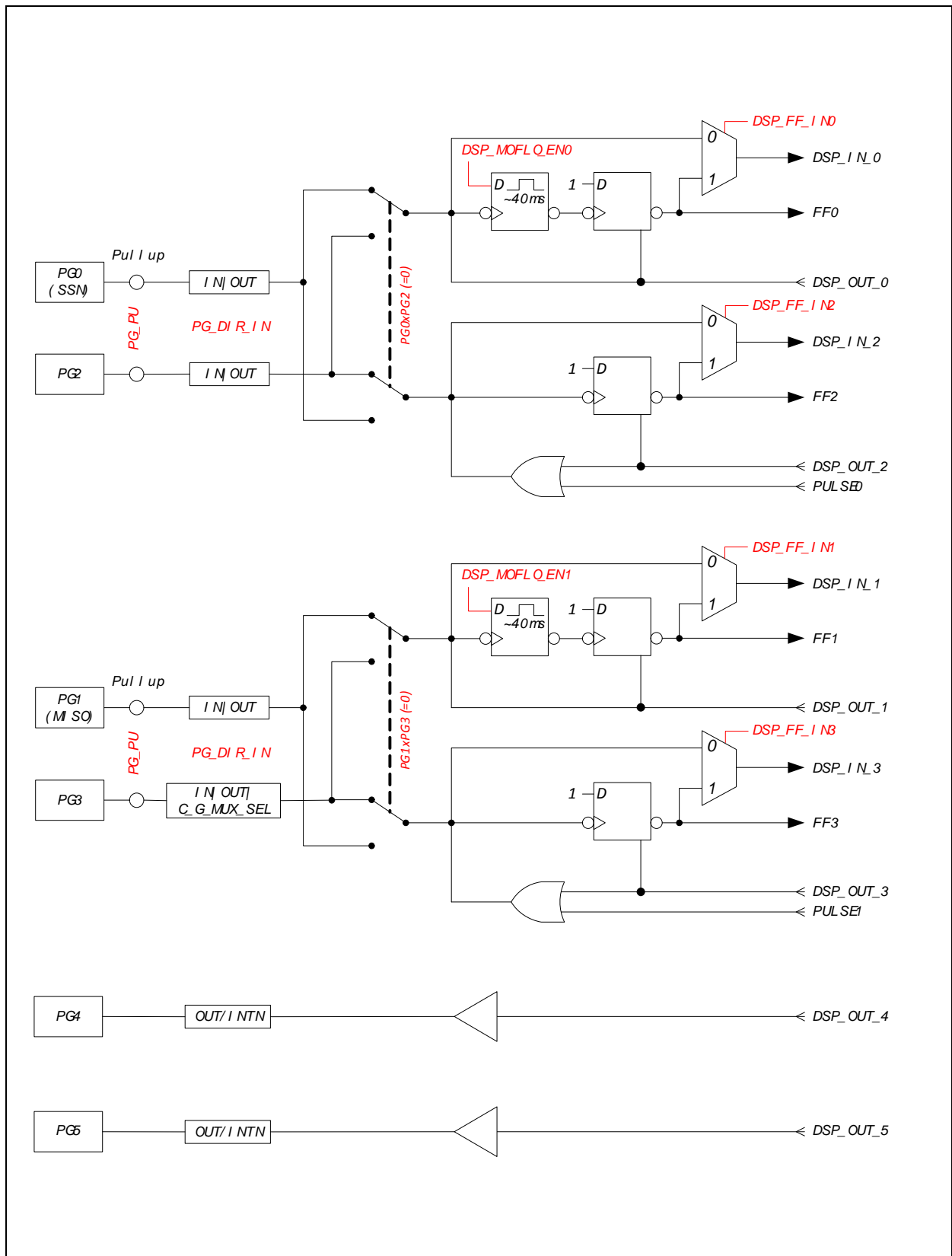
**Figure 123:**  
**Pin Assignment**

| External Port | Description             | In/Out                  |
|---------------|-------------------------|-------------------------|
| PG0           | SSN (in SPI-Mode)       | In                      |
|               | DSP_x_0 or DSP_x_2      | In <sup>(1)</sup> / Out |
|               | FF0 or FF2              | In <sup>(1)</sup>       |
|               | Pulse0                  | Out                     |
| PG1           | MISO (in SPI-Mode)      | Out                     |
|               | DSP_x_1 or DSP_x_3      | In <sup>(1)</sup> / Out |
|               | FF1 or FF3              | In <sup>(1)</sup>       |
|               | Pulse1                  | Out                     |
| PG2           | DSP_x_0 or DSP_x_2      | In <sup>(1)</sup> / Out |
|               | FF0 or FF2              | In <sup>(1)</sup>       |
|               | Pulse0                  | Out                     |
| PG3           | DSP_x_1 or DSP_x_3      | In <sup>(1)</sup> / Out |
|               | FF1 or FF3              | In <sup>(1)</sup>       |
|               | Pulse1                  | Out                     |
|               | C_G_MUX_SEL             | Out                     |
| PG4           | DSP_OUT_4 (output only) | Out                     |
|               | INTN                    | Out                     |
| PG5           | DSP_OUT_5 (output only) | Out                     |
|               | INTN                    | Out                     |

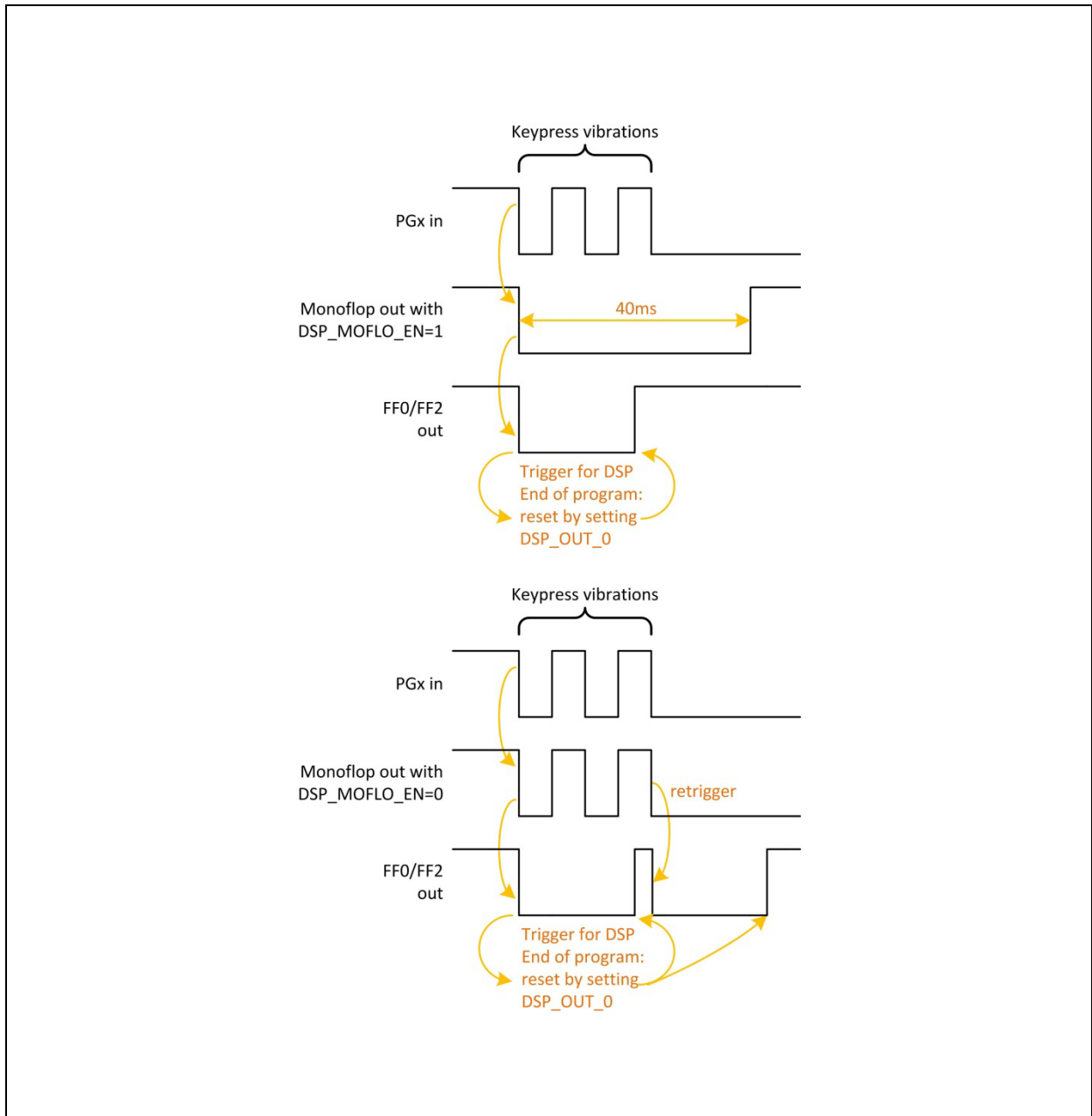
**Note(s):**

1. These ports provide an optional debouncing filter and an optional pull-up resistor.

**Figure 124:**  
GPIO Assignment



**Figure 125:**  
**Port Trigger Timing**



There is a possibility to activate a 40ms debounce filter (“monoflop”) for the ports in case these are used as inputs. This might be useful especially in case the DSP is started by the pins (signals FF0, FF2). [Figure 125](#) shows the effect of the monoflop filter.

The settings for this are made in the following configuration registers:

**Figure 126:**  
**Ports for Pulse Output**

| Reg | Parameter    | Settings   | Description  |
|-----|--------------|--|--|
| 27  | DSP_MOFLO_EN | Bit 6 for PG0<br>Bit 7 for PG1                           | Activates anti-bouncing filter in PG0 and PG1 lines  |
| 27  | PG0xPG2      | 0 : PG0<br>1 : PG2                                       | The pulse codes can be output at ports PG0 & PG1 or PG2 & PG3. In I <sup>2</sup> C mode they can be optionally given out on PG2 and PG3, instead of PG0 and PG1. |
| 27  | PG1xPG3      | 0 : PG1<br>1 : PG3                                       |  |
| 29  | DSP_FF_IN    | Bit 0 : PG0<br>Bit 1 : PG1<br>Bit 2 : PG2<br>Bit 3 : PG3 | Pin mask for latching flip-flop activation   |
| 30  | PG4_INTN_EN  | Bit 6  | Activates INTN at port PG4   |
| 30  | PG5_INTN_EN  | Bit 7  | Activates INTN at port PG5   |
| 33  | PG_DIR_IN    | 0 : Output<br>1 : Input                                  | Toggles outputs to inputs (PG3/bit7 to PG0/bit4).  |
| 33  | PG_PU        | Bit 0 : PG0<br>Bit 1 : PG1<br>Bit 2 : PG2<br>Bit 3 : PG3 | Activates pull-up resistors in PG0 to PG3 lines; useful for mechanical switches.   |

### DSP Configuration

Configuration register 8 defines the DSP operation. Relevant bits are:

DSP\_SRAM\_SEL, DSP\_START, DSP\_STARTONOVL, DSP\_STARTONTEMP, DSP\_STARTPIN, DSP\_WATCHDOG\_LENGTH, DSP\_SPEED

**Figure 127:**  
DSP Configuration

| Reg | Parameter          | Settings   | Description  |
|-----|--------------------|--|--|
| 27  | DSP_SPEED          | 0 : Fastest<br>1 : Fast<br>2 : Recommended<br>3 : Low-current (slow) | Setting the DSP speed  |
| 29  | DSP_STARTONPIN     | 0 : FF0<br>1 : FF1<br>2 : FF2<br>3 : FF3                             | Pin mask for DSP trigger   |
| 30  | DSP_START_EN<2..0> |  | DSP trigger enable<br>'bxxx1 : Trigger by end of CDC<br>'bxx1x : Trigger by end of RDC (recommended)<br>'bx1xx : Trigger by timer<br>'b1xxx : Obsolete |
| 34  | DSP_TRIG_BG        | 0 : Disabled<br>1 : Enabled  | Bandgap refresh is triggered by start of DSP determination.  |



### DSP Start

There are various options to trigger the DSP.

In slave operation:

- Trigger by external controller. This is done by sending opcode "CDC Start conversion" or "DSP\_TRIG".

In stand-alone operation:

- Trigger by pin. The trigger pin is selected between pins PG0 to PG3 by configuration parameters DSP\_STARTPIN and PG0\_X\_PG2/PG1\_X\_PG3. Signal FFX triggers the DSP. FFX has to be reset in the firmware by setting DSP\_x, e.g.  
BitS DSP\_2  
BitC DSP\_2
- Trigger by the end of  
CDC  
RDC  
Timer  
Or by an interrupt. The option is selected by configuration parameter DSP\_START\_EN.

(Hint: DSP is also triggered by

- Toggling RUNBIT from 0 to 1. This is indicated by Flag "AWAKE\_TRIGGERED\_N".
- After each CDC or RDC cycle. This is indicated by Flags TDC\_C\_TRIGGERED\_N and TDC\_R\_TRIGGERED\_N)

### Watchdog

The watchdog is based on the OLF clock and counts always, even if the DSP is halted. If the DSP doesn't reset the Watchdog within 9s to 15s a power-on reset is generated => auto-boot. Status Flag POR\_FLAG\_Wdog is set.

The watchdog is implemented to handle situations where no CDC or RDC is running.

In applications as slave, the watchdog has to be disabled. This can be done by writing a 0x5A to WD\_DIS. If the watchdog is used disarm the watchdog in advance to any SIF-Communication.

### System Reset

In case the PCap04 is operated as a slave, not in self-boot mode, it is necessary to do the following actions after applying power:

1. Send opcode Power-on Reset via the serial interface, opcode 0x88.
2. Write the firmware into the SRAM by means of opcode "Write to SRAM".
3. Write the configuration registers by means of opcode "Write Config". Register 47 with the RUNBIT has to be the last one in order.
4. Send a start command, opcode 0x8C

**Instruction Set**

The complete instruction set of the PCap04 consists of 29 core instructions that have unique op-code decoded by the CPU. Further, **ams** offers a set of libraries including common constant definitions and mathematical operations

The library family is intended to be continuously expanded and be a great help during software development.

**Figure 128:**  
Instruction Set

| Simple Arithmetic | Miscellaneous | RAM Access         | Bitwise Operation |
|-------------------|---------------|--------------------|-------------------|
| add               | init          | rad                | not               |
| sign              | nop           | clear              | and               |
| sub               | rst           | load               | or                |
| inc               | stop          | load2exp           | xor               |
|                   | wdr           | mov<br>push<br>pop |                   |

| Complex Arithmetic | Shift & Rotate | Unconditional Jump | Bitwise |
|--------------------|----------------|--------------------|---------|
| div                | shiftL         | goto<br>jsb        | bitC    |
| mult               | shiftR         | jrt                | bitS    |

| Conditional Jump |      |       |
|------------------|------|-------|
| jcd              | jEQ  | jOfIC |
| jCarC            | jNE  | jOfIS |
| jCarS            | jNeg | jPOS  |

### Instructions

| <b>and</b>      | <b>Bitwise AND</b>                                 |
|-----------------|--|
| Syntax:         | and p1,p2  |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r]<br>p1 != p2 |
| Calculus:       | p1 : p1 & p2                                       |
| Flags affected: | C O S Z  |
| Bytes:          | 1  |
| Description:    | Bitwise AND (conjunction)                          |
| Category:       | Bitwise operation                                  |

| <b>add</b>      | <b>Addition</b>                        |
|-----------------|--|
| Syntax:         | add p1,p2                              |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r] |
| Calculus:       | p1 : p1 + p2                           |
| Flags affected: | C O S Z                                |
| Bytes:          | 1                                      |
| Description:    | Addition of two registers              |
| Category:       | Simple arithmetic                      |

| <b>bitC</b>     | <b>Clear single bit</b>                          |
|-----------------|--|
| Syntax:         | bitC p1  |
| Parameters:     | p1 = number 0 to 15                              |
| Calculus:       | Set bit number p1 of the DSP output bits bit = 0 |
| Flags affected: | -  |
| Bytes:          | 1  |
| Description:    | Clear a single bit in the DSP output bits        |
| Category:       | Bitwise  |

| <b>bitS</b>     | <b>Set single bit</b>                            |
|-----------------|--|
| Syntax:         | bitS p1  |
| Parameters:     | p1 = number 0 to 15                              |
| Calculus:       | Set bit number p1 of the DSP output bits bit = 1 |
| Flags affected: | -  |
| Bytes:          | 1  |
| Description:    | Set a single bit in the DSP output bits          |
| Category:       | Bitwise  |

| <b>clear</b>    | <b>Clear register</b>         |
|-----------------|-------------------------------|
| Syntax:         | clear p1                      |
| Parameters:     | p1 = ACCU [a,b,r]             |
| Calculus:       | p1 : 0                        |
| Flags affected: | S Z                           |
| Bytes:          | 2                             |
| Description:    | Clear addressed register to 0 |
| Category:       | RAM access                    |

| <b>div</b>      | <b>Unsigned division</b>  |
|-----------------|---|
| Syntax:         | div   |
| Parameters:     | -   |
| Calculus:       | Single div code: $b : (a/r), a : \text{Remainder} * 2$<br>N div codes: $b : (a/r)*2^{(N-1)}, a : \text{Remainder} * (2^N)$  |
| Flags affected: | S Z   |
| Bytes:          | 1   |
| Description:    | <p>Unsigned division of two 32-bits registers. When the div opcode is used once, the resulting quotient is assigned to register 'b'. The remainder can be calculated from 'a'. When N div opcodes are used one after another, the result in b : <math>(a/r)*2^{(N-1)}</math>. See also ROM routine div_xx.</p> <p>Before executing the first division step, the following conditions must be satisfied: 'b' = 0, and <math>0 &lt; 'a' &lt; 2 * 'r'</math>.</p> <p>If this condition is not satisfied, you can shift 'a' until this is satisfied. After shifting, if <math>a \rightarrow a * (2^{ea})</math> and <math>r \rightarrow r * (2^{er})</math>, then the resulting quotient b for N division steps is <math>b : (a/r) * 2^{(1+ea-er-N)}</math><br/><math>a = \text{Remainder} * (2^N)</math></p> |
| Category:       | Complex arithmetic  |

| <b>inc</b>      | <b>Increment register</b> |
|-----------------|---------------------------|
| Syntax:         | inc p1                    |
| Parameters:     | p1 = ACCU [a,b,r]         |
| Calculus:       | $p1 : p1 + 1$             |
| Flags affected: | C O S Z                   |
| Bytes:          | 1                         |
| Description:    | Increment register        |
| Category:       | Simple arithmetic         |

| <b>init</b>     | <b>Init reset</b>   |
|-----------------|---|
| Syntax:         | Init  |
| Parameters:     | -   |
| Calculus:       | -   |
| Flags affected: | C O S Z   |
| Bytes:          | 1   |
| Description:    | Initialization and reset. Sets back CDC, RDC and CPU. Copies configuration from NVRAM into configuration registers. |
| Category:       | Miscellaneous   |

| <b>jCarC</b>    | <b>Jump on Carry Clear</b>   |
|-----------------|--|
| Syntax:         | jCarC p1   |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | if (carry == 0) PC : p1  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Jump on carry clear. Program counter will be set to target address if carry is clear. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jCarS new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump   |

| <b>jCarS</b>    | <b>Jump on Carry Set</b>   |
|-----------------|--|
| Syntax:         | jCarS p1   |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | if (carry == 1) PC : p1  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Jump on carry set. Program counter will be set to target address if carry is set. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jCarC new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump   |

| <b>jcd</b>      | <b>Conditional Jump</b>  |
|-----------------|--|
| Syntax:         | jcd p1, p2   |
| Parameters:     | p1 = Flag or input port bit [63...0]. See section 2.3 for DSP Inputs.<br>p2 = jumplabel  |
| Calculus:       | If ( p1 == 1 ) PC : p2   |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Program counter is set to target address if the bit given by p1 is set to one. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jcd p1, new_label1 jsb new_label2 jrt new_label1: jsb p2 jrt new_label2: ;...</pre> |
| Category:       | Conditional jump   |

| <b>jEQ</b>      | <b>Jump on Equal</b>  |
|-----------------|---|
| Syntax:         | jEQ p1  |
| Parameters:     | p1 = jumplabel  |
| Calculus:       | if (Z == 0) PC : p1   |
| Flags affected: | -   |
| Bytes:          | 2   |
| Description:    | <p>Jump on equal resp. zero. Program counter will be set to target address if the foregoing result is zero. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jNE new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump  |

| <b>jNE</b>      | <b>Jump on Not Equal</b>  |
|-----------------|---|
| Syntax:         | jNE p1  |
| Parameters:     | p1 = jumplabel  |
| Calculus:       | if (Z == 1) PC : p1   |
| Flags affected: | -   |
| Bytes:          | 2   |
| Description:    | <p>Jump on not equal resp. not zero. Program counter will be set to target address if the foregoing result is zero. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jEQ new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump  |



| <b>jNeg</b>     | <b>Jump on Negative</b>  |
|-----------------|--|
| Syntax:         | jNeg p1  |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | if (S == 1) PC : p1  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Jump on negative. Program counter will be set to target address if the foregoing result is negative (Bit 31 == 1). The target address is given by using a jumplabel.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jPos new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump   |

| <b>jOvIC</b>    | <b>Jump on Overflow Clear</b>  |
|-----------------|--|
| Syntax:         | jOvIC p1   |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | if (O == 0) PC : p1  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Jump on overflow clear. Program counter will be set to target address if the overflow flag of the foregoing operation is clear. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jOfIS new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump   |

| <b>jOvIS</b>    | <b>Jump on Overflow Set</b>   |
|-----------------|---|
| Syntax:         | jOvIS p1  |
| Parameters:     | p1 = jumplabel  |
| Calculus:       | if (O == 1) PC : p1   |
| Flags affected: | -   |
| Bytes:          | 2   |
| Description:    | <p>Jump on overflow set. Program counter will be set to target address if the overflow flag of the foregoing operation is set. The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt. If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jOfIC new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump  |

| <b>jPos</b>     | <b>Jump on Positive</b>   |
|-----------------|---|
| Syntax:         | jPos p1   |
| Parameters:     | p1 = jumplabel  |
| Calculus:       | if (S == 0) PC : p1   |
| Flags affected: | -   |
| Bytes:          | 2   |
| Description:    | <p>Jump on positive. Program counter will be set to target address if the foregoing result is positive (Bit 31 == 0). The target address is given by using a jumplabel. The conditional jump does not serve the stack. Therefore it is not possible to return by jrt. If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>jNeg new_label jsb p1 jrt new_label: .....</pre> <p>In this case the stack will be loaded with p1, and therefore the stack capacity will be reduced by one.</p> |
| Category:       | Conditional jump  |

| <b>jrt</b>      | <b>Return from subroutine</b>   |
|-----------------|---|
| Syntax:         | jrt   |
| Parameters:     | -   |
| Calculus:       | PC : PC from jsb-call   |
| Flags affected: | -   |
| Bytes:          | 1   |
| Description:    | Return from subroutine. A subroutine can be called via 'jsb' and exited by using jrt. The program is continued at the next command following the jsb-call. You have to close a subroutine with jrt - otherwise there will be no jump back. The stack is decremented by 1. |
| Category:       | Unconditional Jump  |

| <b>goto</b>     | <b>Unconditional relative Jump</b>   |
|-----------------|--|
| Syntax:         | goto p1  |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | PC : p1  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | <p>Jump to jumplabel. Program counter will be set to target address. The target address is given by using a jumplabel. The goto command does not serve the stack. Therefore it is not possible to return by jrt.</p> <p>If the target address is beyond the range of current address (PC) -128/+127 bytes, then the assembler software will substitute this opcode for the following optimization:</p> <pre>goto new_label1 jsb new_label2 jrt new_label1: jsb p2 jrt new_label2: ;...</pre> |
| Category:       | Unconditional Jump   |

| <b>jsb</b>      | <b>Unconditional Jump</b>  |
|-----------------|--|
| Syntax:         | jsb p1   |
| Parameters:     | p1 = jumplabel   |
| Calculus:       | PC : PC from jsub-call   |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | Jump to subroutine without condition. The programm counter is loaded by the address given through the jumplabel. The subroutine is processed until the keyword 'jrt' 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).<br>The stack is incremented by 1. |
| Category:       | Unconditional Jump   |

| <b>load</b>     | <b>Load Accumulator</b>   |
|-----------------|---|
| Syntax:         | load p1,p2  |
| Parameters:     | p1 = ACCU [a,b]<br>p2 = 6..32-bit integer number (positive/negative, decimal or hexadecimal)  |
| Calculus:       | p1 : p2   |
| Flags affected: | S Z   |
| Bytes:          | 3..8 (depending on p2)  |
| Description:    | Move constant to p1 (p1=ACCU, p2=NUMBER)<br>The following instruction is not allowed:<br>load r, NUMBER<br>This instruction is a macro that is replaced by the following opcodes:<br>rad NUMBER[23:18]<br>rad NUMBER[17:12]<br>rad NUMBER[11:6]<br>rad NUMBER[5:0]<br>rad rad_stack_24b<br>move [a, b], r<br>Here the 24-bits number is split into four pieces, the symbol [xx:yy] indicates the individual bit range belonging to each piece. Please notice that the ram address pointer is changed during the operations, keep this in mind while coding. |
| Category:       | RAM access  |

| <b>load2exp</b> | <b>Load Accumulator with 2exp</b>  |
|-----------------|--|
| Syntax:         | load2exp p1,p2   |
| Parameters:     | p1 = ACCU [a,b]<br>p2 = 6-bit number   |
| Calculus:       | $p1 : 2^{p2}$  |
| Flags affected: | S Z  |
| Bytes:          | 2  |
| Description:    | <p>Move <math>2^{(p2)}</math> to p1 (p1=ACCU, p2=NUMBER)<br/>           The following instruction is not allowed:<br/>           load r, NUMBER<br/>           This instruction is a macro that is replaced by the following opcodes:<br/>           rad NUMBER[5:0]<br/>           rad load2exp<br/>           move [a, b], r</p> |
| Category:       | RAM access   |

| <b>mov</b>      | <b>Move</b>   |
|-----------------|---|
| Syntax:         | mov p1,p2   |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r]  |
| Calculus:       | $p1 : p2$   |
| Flags affected: | S Z   |
| Bytes:          | 1   |
| Description:    | <p>Move content of p2 to p1<br/>           Assembler will understand also the old opcode move</p> |
| Category:       | RAM access  |

| <b>mult</b>     | <b>Multiply</b>   |
|-----------------|---|
| Syntax:         | mult  |
| Parameters:     | -   |
| Calculus:       | $ab : (b * r)$  |
| Flags affected: | S Z   |
| Bytes:          | 1   |
| Description:    | <p>Unsigned multiplication of the content of ab and r registers.<br/>           ab is the composition of the registers a and b, forming an 64-bits long register, where 'a' takes the most significant bits, and register 'b' takes the less significant ones.<br/>           The result is stored in the composed register a and b. The register 'a' must be previously cleared.</p> <p>This instruction only executes one multiplication step, to execute a full 32-bits multiplication, this instruction must be executed 32 times. This has the disadvantage of being tedious to code, but also has the advantage of executing only the amount of arithmetic needed, if you do not need a 32-bits multiplication but N, where <math>N &lt; 32</math>, then you have only to execute N multiplication steps in order to complete the full N-bits multiplication.</p> <p>After one multiplication step, register 'a' contains <math>((a + (b[0] * r)) \gg 1)</math>, and register 'b' contains { a[0], b[3:1] }. For example: lets denote the individual bits of register 'a' as a[31], a[30], a[29].....a[2], a[1], a[0], and lets denote a range of bits of 'a' as: a[3:0], meaning the 4 less significant bits of register 'a'.</p> <p>Then, after one multiplication step, <math>a[30:0] = (a[31:0] + r[31:0] * b[0]) \gg 1</math>, where <math>\gg 1</math>, means right shift by one position; the value of a[31] is zero, and <math>b[31] = (a[0] + r[0] * b[0])</math>, and <math>b[30:0] = b[31]</math>. The register r remains unchanged.</p> |
| Category:       | Complex arithmetic  |

| <b>nop</b>      | <b>No operation</b>                             |
|-----------------|---|
| Syntax:         | -   |
| Parameters:     | -   |
| Calculus:       | -   |
| Flags affected: | -   |
| Bytes:          | 1   |
| Description:    | Placeholder code or timing adjust (no function) |
| Category:       | Miscellaneous                                   |

| <b>not</b>      | <b>Bitwise NOT</b>         |
|-----------------|----------------------------|
| Syntax:         | not p1                     |
| Parameters:     | p1 = ACCU [a,b,r]          |
| Calculus:       | p1 : ~ p1                  |
| Flags affected: | C O S Z                    |
| Bytes:          | 1                          |
| Description:    | Invert register (negation) |
| Category:       | Bitwise operation          |

| <b>or</b>       | <b>Bitwise OR</b>                                  |
|-----------------|--|
| Syntax:         | or p1,p2   |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r]<br>p1 != p2 |
| Calculus:       | p1 : p1   p2                                       |
| Flags affected: | C O S Z  |
| Bytes:          | 1  |
| Description:    | Bitwise OR (disjunction)                           |
| Category:       | Bitwise operation                                  |

| <b>pop</b>      | <b>Remove address</b>       |
|-----------------|-----------------------------|
| Syntax:         | pop                         |
| Parameters:     | -                           |
| Calculus:       | -                           |
| Flags affected: | -                           |
| Bytes:          | 1                           |
| Description:    | Roll back ram-address stack |
| Category:       | RAM access                  |

| <b>rst</b>      | <b>Power On Reset</b>   |
|-----------------|---|
| Syntax:         | rst   |
| Parameters:     | -   |
| Calculus:       | -   |
| Flags affected: | S Z   |
| Bytes:          | 5   |
| Description:    | <p>This is a symbolic opcode which is equivalent to the following instruction sequence:</p> <pre>bitC 54 bitC 55 bitS 55 bitS 54 bitC 55</pre> <p>The assembler understands also the old powerOnReset</p> |
| Category:       | Miscellaneous   |

| <b>push</b>     | <b>Put data into stack memory</b>   |
|-----------------|---|
| Syntax:         | push p1   |
| Parameters:     | p1 = NUMBER [6-bit]   |
| Calculus:       | -   |
| Flags affected: | -   |
| Bytes:          | 1   |
| Description:    | <p>Writes p1 to RAM address stack (range: 0 to 63). Commit constant value to ROM routines.</p> <pre>push 22 ; number of fractional digits for CDC ratios push 4 ; Port Number for reference value (PC4) jsb _ROM_CDC__</pre> <p>Note: for advanced users only. Better use rad</p> |
| Category:       | RAM access  |



| <b>rad</b>      | <b>Set RAM Address Pointer</b>   |
|-----------------|--|
| Syntax:         | rad p1   |
| Parameters:     | p1 = NUMBER [7-bit]  |
| Calculus:       | -  |
| Flags affected: | -  |
| Bytes:          | 2  |
| Description:    | Set pointer to RAM address (range: 0 to 127) in the RAM address stack.<br>Note: Internally the RAM is made of 2 pages, 64 words each. The assembler translates the combination of a bitS/C and a push instruction into the rad instruction.<br>rad 15<br>move r, b will move the content of b the address 15 |
| Category:       | RAM access   |

| <b>wdr</b>      | <b>Clear watch dog timer</b>   |
|-----------------|--|
| Syntax:         | wdr  |
| Parameters:     | -  |
| Calculus:       | -  |
| Flags affected: | -  |
| Bytes:          | 5  |
| Description:    | Clear watchdog timer.<br>This is a symbolic opcode which is equivalent to the following instruction sequence:<br>bitC 54<br>bitC 55<br>bitS 54<br>bitS 55<br>bitC 54<br>The assembler understands also the old resetWDG opcode |
| Category:       | Miscellaneous  |

| <b>shiftL</b>   | <b>Shift Left</b>   |
|-----------------|---|
| Syntax:         | shiftL p1   |
| Parameters:     | p1 = ACCU [a, b]  |
| Calculus:       | p1 : p1<< 1   |
| Flags affected: | S Z   |
| Bytes:          | 1   |
| Description:    | Shift p1 left --> shift p1 register to the left, fill LSB with 0, MSB is placed in carry register |
| Category:       | Shift and rotate  |

| <b>shiftR</b>   | <b>Shift Right</b>   |
|-----------------|--|
| Syntax:         | shiftR p1  |
| Parameters:     | p1 = ACCU [a, b]   |
| Calculus:       | p1: p1>> 1   |
| Flags affected: | S Z  |
| Bytes:          | 1  |
| Description:    | Signed shift right of p1 --> shift p1 right, MSB is duplicated according to whether the number is positive or negative |
| Category:       | Shift and rotate   |

| <b>sign</b>     | <b>Sign</b>  |
|-----------------|--|
| Syntax:         | sign p1  |
| Parameters:     | p1 = ACCU [a,b]  |
| Calculus:       | If SF = 0 => p1 :  p1 , SF : S(p1) If SF = 1 => p1 : -  p1 , SF : S(p1)  |
| Flags affected: | S Z SF   |
| Bytes:          | 1  |
| Description:    | The intention of this opcode is to take the absolute value of one parameter before multiplication or division and to restore the sign after this operation.<br>Assuming the Signum flag is zero, the absolute value of accumulator is taken and the sign from accumulator is stored to SF. At the second time this opcode is used the sign to p1 will be restored from SF<br>Zero is assumed to be positive. |
| Category:       | Simple arithmetic  |

| <b>stop</b>     | <b>Stop</b>  |
|-----------------|--|
| Syntax:         | stop   |
| Parameters:     | -  |
| Calculus:       | -  |
| Flags affected: | -  |
| Bytes:          | 1  |
| Description:    | Stop of the PCAP-Controller. The clock generator is stopped, the PCAP-Controller go to standby. A restart can be achieved by an external event like 'watchdog timer', 'external switch' or 'new capacitive measurement results'. Usually this opcode is the last command in the assembler listing. |
| Category:       | Miscellaneous  |

| <b>sub</b>      | <b>Subtraction</b>   |
|-----------------|--|
| Syntax:         | sub p1,p2  |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r]<br>p1 != p2   |
| Calculus:       | p1 : p1 – p2   |
| Flags affected: | C O S Z  |
| Bytes:          | 1  |
| Description:    | Subtraction of 2 registers.<br>The following instructions are not allowed: sub a,a. sub b,b. sub r,r |
| Category:       | Simple arithmetic  |

| <b>xor</b>      | <b>Bitwise XOR</b>                                 |
|-----------------|--|
| Syntax:         | xor p1,p2  |
| Parameters:     | p1 = ACCU [a,b,r]<br>p2 = ACCU [a,b,r]<br>p1 != p2 |
| Calculus:       | p1 : p1 ^ p2                                       |
| Flags affected: | C O S Z  |
| Bytes:          | 1  |
| Description:    | Bitwise XOR (antivalence)                          |
| Category:       | Bitwise operation                                  |

**Instruction Details****Pointer**

; Copying the Cratio results to the persistent bank in RAM

```
load b, 6
load a, __sub_cdc_CO_Ratio_temp
rad DPTR1
move r, a
load a, CO_Ratio_RAM
rad DPTR0
move r, a
jsb __ROM_dma__
```

rad\_at\_DPTR0 to rad\_at\_DPTR3 are special instructions for indirect addressing. \_at\_DPTR0 to \_at\_DPTR3 are special RAM addresses 284 and 287 that have been defined in the firmware.

RAM addresses 105 to 108 are used as data pointers, named DPTR0 to DPTR3.

By means of

```
rad DPTR0
move r, a
```

an address is loaded into DPTR0. With

```
rad _at_DPTR0
```

the address in DPTR0 is loaded.

Example 1: copy sequentially RAM-content from one address-space to another

```
Load a, CO_ratio
rad DPTR1
move r, a
load a, RES0
rad DPTR0
move r, a
load b, 8
jsb __ROM_dma__ ; call ROM routine
```

Example 2: Copying the Rratio results to the persistent bank into RAM

```
rad 4
rad rad_stack_6b
move b, r
load a, __sub_rdc_R0_Ratio_temp; Source for copy
rad DPTR1
move r, a

rad R0_Ratio_RAM ; Destination for copy
rad rad_stack_6b
move a, r
rad DPTR0
move r, a
jsb __ROM_dma__
```

### *Transfer Constants with Push and Pop*

Call of a subroutine:

```

push  FPP_CRATIO          ; Stack - 1 ---> Number of fpp in the result
push  C_REF_PORT_NUMBER; Stack - 0 ---> Reference Port Number
jsb   _ROM_cdc___        ; Calling ROM routine for Ratio calculation
    
```

```

_ROM_cdc__:
bitS PAGESEL_OUT
rad   rad_stack_6b; (Stack - 0) contains Reference Port Number
move  b, r
pop
rad   __sub_cdc_RefPort
move  r, b          ; Temporarily saving the Reference Port Number in RAM
pop
pop
rad   rad_stack_6b ; (Stack - 1) contains number of fractional digits in the
result (cdc_fpp)
move  b, r          ; B = The number of fractional digits , Result_fpp
    
```

### *mult*

The instruction “mult” is just a single multiplication step. To do a complete 32-bit multiplication this instruction has to be done 32 times. The multiplicands are in accumulators b and r. Every step takes the lowest bit of b. If it is one, r is added to accumulator a, else nothing is added. Thereafter a and b are shifted right. The lowest bit of a becomes the highest bit of b. Before the first step of the multiplication, a has to be cleared. The final result is spread over both accumulators a and b.

The use of mult is simplified by using the ROM routines mult\_01 to mult\_32.

In many cases it will not be necessary to do the full 32 multiplication steps but much fewer. The necessary number of steps is given by the number of significant bits of b and also the necessary significant number of bits of the result.

But, if the multiplication steps are less than 32, the result might be spread between accumulators a and b. Doing an appropriate right shift of the multiplicand in r, and the appropriate number of multiplication steps, it is possible to ensure that the result is either fully in a or in b.

### *Handover of Constants by Push & Pop*

A simple method to hand over constants with a value 0 to 63 is using push & pop instructions. The following shows an example for calling a subroutine.

Subroutine call:

```
push  FPP_CRATIO;      Stack - 1 -> Number of fpp in the result
push  C_REF_PORT_NUMBER; Stack - 0 -> Reference Port Number
jsb   _ROM_cdc__;      Calling ROM routine for Ratio calculation
```

\_ROM\_cdc\_\_:

```
rad   rad_stack_6b; (Stack - 0) contains Reference Port Number
move  b, r
pop
rad   __sub_cdc_RefPort
move  r, b;      Temporarily saving the Reference Port Number in RAM
pop
pop
rad   rad_stack_6b; (Stack - 1) contains number of fractional digits in the result (cdc_fpp)
move  b,r; B = The number of fractional digits , Result_fpp
```

### *div*

The instruction “div” is, like the multiplication, just a single step of a complete division. The necessary number of steps for a complete division depends on the accuracy of the result. The dividend is in accumulator a, the divisor is in accumulator r. Every division step contains following actions:

- leftshift b
- compare a and r. If a is bigger or equal to r then r is subtracted from a and One is added to b
- leftshift a

Start Conditions:  $0 < a < 2*r$ ,  $b = 0$

Again, multiple division steps are implemented in ROM library to be easily used by customers, calling div\_01 to div\_32. A call of function e.g. div\_24 out of this library will do a sequence of 24 division steps. The result is found in b, the remainder in a.

With N division steps the result in b:  $(a/r)+2^{(N-1)}$ ,

a: remainder\* $2^N$ .

Example 1: a = 2, r = 6, Integer division

Figure 129:  
Example: Division 2/6

| Steps | a = 2          | b        | r = 6   |  |
|-------|----------------|----------|---------|--|
|       | 000000..000010 | 0..00000 | 0..0110 | a < r, leftshift b, a                        |
| 1     | 000000..000100 | 0..00000 | 0..0110 | a < r, leftshift b, a                        |
| 2     | 000000..001000 | 0..00000 | 0..0110 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 3     | 000000..000100 | 0..00001 | 0..0110 | a < r, leftshift b, a                        |
| 4     | 000000..001000 | 0..00010 | 0..0110 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 5     | 000000..000100 | 0..00101 | 0..0110 |  |

Quotient =  $b * 2^{(1-steps)} = 0.3125$ , Remainder =  $a * 2^{(-steps)} = 4 * 2^{-5} = 0.125$

The following two, more complex examples show a nice advantage of division over multiplication: The resolution in bit is directly given by the number of division steps. With this knowledge, assembly programs can be written very effectively. It is easy to use only the number of division steps that is necessary.

Example 2: A = 8.75, R = 7.1875, Fractional number division, A & R with 4 fractional digits each.

$8.75/7.1875 = a * 2^{expA} / r * 2^{expR} = a * 2^{-4} / r * 2^{-4}$

Figure 130:  
Example: Division 8.75/7.1875

| Steps | a = 140   | b         | r = 115   |  |
|-------|-----------|-----------|-----------|--|
|       | 1000 1100 | 0000 0000 | 0111 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 1     | 0011 0010 | 0000 0001 | 0111 0011 | a < r, leftshift b, a                        |
| 2     | 0110 0100 | 0000 0010 | 0111 0011 | a < r, leftshift b, a                        |
| 3     | 1100 1000 | 0000 0100 | 0111 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 4     | 1010 1010 | 0000 1001 | 0111 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 5     | 0110 1110 | 0001 0011 | 0111 0011 | a < r, leftshift b, a                        |
| 6     | 1101 1100 | 0010 0110 | 0111 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 7     | 1101 0010 | 0100 1101 | 0111 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 8     | 1011 1110 | 1001 1011 | 0111 0011 |  |



$$\text{Quotient} = b * 2^{(1+\text{expA}-\text{expR}-\text{steps})} = 155 * 2^{(1-4+4-8)} = 1.2109$$

$$\text{Remainder} = a * 2^{(-\text{steps}-\text{expR})} = 190 * 2^{-12} = 0.0463$$

Example 3: A = 20, R = 1.2, Fractional number division, R < A.

A and R are shifted to left to display the fractional digits of R. Further, R has to be shifted to the left till it is bigger than A/2.

$$20/1.2 = a * 2^{\text{expA}} / r * 2^{\text{expR}} = a * 2^{-4} / r * 2^{-8}$$

**Figure 131:**  
**Example 3: Division 20/1.2**

| Steps | a = 320        | b              | r = 307        |  |
|-------|----------------|----------------|----------------|--|
|       | 0001 0100 0000 | 0000 0000 0000 | 0001 0011 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 1     | 0000 0001 1010 | 0000 0000 0001 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 2     | 0000 0011 0100 | 0000 0000 0010 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 3     | 0000 0110 1000 | 0000 0000 0100 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 4     | 0000 1101 0000 | 0000 0000 1000 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 5     | 0001 1010 0000 | 0000 0001 0000 | 0001 0011 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 6     | 0000 1101 1010 | 0000 0010 0001 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 7     | 0001 1011 0100 | 0000 0100 0010 | 0001 0011 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 8     | 0001 0000 0010 | 0000 1000 0101 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 9     | 0010 0000 0100 | 0001 0000 1010 | 0001 0011 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 10    | 0001 1010 0010 | 0010 0001 0101 | 0001 0011 0011 | leftshift b, a >= r: a-=r, b+=1, leftshift a |
| 11    | 0000 1101 1110 | 0100 0010 1011 | 0001 0011 0011 | a < r, leftshift b, a                        |
| 12    | 0001 1011 1100 | 1000 0101 0110 | 0001 0011 0011 |  |

$$\text{Quotient} = b * 2^{(1+\text{expA}-\text{expR}-\text{steps})} = 2134 * 2^{(1-4+8-12)} = 16.6719$$

The remainder is, as always, smaller than the denominator divided by 2steps e.g. in the present case, remainder < 1.2 / 2<sup>12</sup> = 0,0003

$$\text{Steps} = 1 + \text{expA} - \text{expB} - \text{expRes}$$

### ROM Routines

The following routines are implemented as ROM code:

**Figure 132:**  
ROM Routines

| ROM Routine                 | Address | ROM Routine  | Address |
|-----------------------------|---------|--|---------|
| _ROM_Version__              | 1024    | _ROM_median__                                      | 1804    |
| _ROM_tdc_dispatch__         | 1029    | _ROM_rdc__   | 1936    |
| _ROM_cdc_cycle__            | 1047    | _ROM_rdc_inverse__                                 | 1962    |
| _ROM_rdc_cycle__            | 1068    | _ROM_cdc__   | 2021    |
| _ROM_cdc_initialize__       | 1086    |  |         |
| _ROM_rdc_initialize__       | 1111    |  |         |
| shiftR_B_32 to _00          | 1140    |  |         |
| shiftL_B_32 to _00          | 1173    |  |         |
| shiftR_A_32 to _00          | 1206    | _ROM_memory_rd_a_32b__ to<br>_ROM_memory_wr_08b__  | 2470    |
| shiftL_A_32 to _00          | 1239    | _ROM_memory_store__                                | 2650    |
| mult_32 to _00              | 1272    | _ROM_memory_recall__                               | 2673    |
| div_33 to _00               | 1305    | _ROM_2PT_Calibration                               | 2696    |
| _ROM_div_variable__         | 1339    | _ROM_polynomial_3rd_degree                         | 2825    |
| _ROM_mult_variable__        | 1396    | __sub_polynomial_load_coefs                        | 2874    |
| _ROM_shift_a_variable__     | 1452    | _ROM_polynomial_4th_degree                         | 2923    |
| _ROM_shift_b_variable__     | 1478    | _ROM_pulse__                                       | 2969    |
| _ROM_dma__                  | 1504    | _ROM_pulse_loaded_cal_vals                         | 2990    |
| _ROM_In__                   | 1520    | _ROM_NVblock_copy_32b_to<br>_ROM_NVblock_copy_08b_ | 3138    |
| _ROM_log10__                | 1547    | _ROM_capacitance_polynomial                        | 3236    |
| _ROM_Id__                   | 1574    | _ROM_capacitance_polynomial_4d                     | 3437    |
| _ROM_signed24_to_signed32__ | 1648    |  |         |

## Handover parameters and RAM addresses for ROM routines

**Figure 133:**  
Parameters and RAM for Key ROM Routines

| ROM Routine                      | Parameter   | RAM Address |
|----------------------------------|---|-------------|
| _ROM_cdc__                       | __sub_cdc_C0_Ratio_temp to<br>__sub_cdc_C5_Ratio_temp | 58..<br>63  |
| _ROM_rdc__<br>_ROM_rdc_inverse__ | __sub_rdc_R0_Ratio_temp to<br>__sub_rdc_R3_Ratio_temp | 64..<br>67  |
| _ROM_mult_variable__             | __sub_standard_multiplier__                           | 82          |
| _ROM_div_variable__              | __sub_standard_divisor__                              | 82          |

In the following we give a detailed description of the ROM routines.

Hint: it is recommended to always include the following files:

```
#device PCap04v1
#include <pcap_standard.h>
#include <PCap04_ROM_addresses_standard.h>
```

to declare addresses to ROM routines.

***\_\_ROM\_Version\_\_***

|                                  |   |
|----------------------------------|---|
| Function:                        | This routine gives back the 8-bit version number of the ROM in the chip |
| Input parameters:                | -   |
| Output/Return value:             | A-Accu : ROM version  |
| Prerequisites                    | None  |
| Dependency on other header files | None  |
| Function call                    | jsb __ROM_Version__ ; address 0x400                                     |
| Temporary memory usage           | -   |
| Example:                         | <pre>jsb __ROM_Version__ rad RES5 move r, a</pre>                       |

***\_\_ROM\_tdc\_dispatch\_\_***

|                                  |   |
|----------------------------------|---|
| Function:                        | TDC library, to be called at the very beginning. Calls subroutines __ROM_cdc_cycle__ or __ROM_rdc-cycle__ and __tdc_awake__, depending on the trigger |
| Input parameters:                | -   |
| Output/Return value:             | -   |
| Prerequisites                    | None  |
| Dependency on other header files | None  |
| Function call                    | jsb __ROM_tdc_dispatch__  |
| Temporary memory usage           | -   |
| Example:                         | <pre>org 0 __SOP__: ; Start of program jsb __ROM_tdc_dispatch__</pre>   |

\_ROM\_cdc\_cycle\_

|                                  |  |
|----------------------------------|--|
| Function:                        | <ol style="list-style-type: none"> <li>1. Calculates the Start-Stop difference in the TDC values of the capacitance measurement ports and accumulates this in the measurement value RAM register (depending on the C_ADD_PTR) respective to the port being measured.</li> <li>2. If the calculated discharge time is too large, then the result is replaced with 0xFFFFFFFF thus indicating overflow.</li> <li>3. The DSP stops if there are more measurements to be done. In case of a last measurement, the DSP returns to where the ROM routine was called from continues to execute the post processing firmware, if any.</li> </ol> |
| Input parameters:                | -  |
| Output/Return value:             | The CDC discharge time measurement value is available in the respective RAM (88 - 95)  |
| Prerequisites                    | This function has to be called after a measurement cycle at every capacitive port.   |
| Dependency on other header files | None   |
| Function call                    | <pre>push &lt;MSB(RAM_startaddress)&gt; push &lt;LSB(RAM_startaddress)&gt; jsb _ROM_cdc_cycle__</pre>  |
| Temporary memory usage           | DPTR0 is overwritten in this routine.  |
| Example:                         | <pre>push 1 ; bit6 of starting address push M0 ; bit 5..0 of starting address jsb _ROM_cdc_cycle (not recommended, use _ROM_tdc_dispatch__ instead)</pre>  |

ROM\_Rdc\_cycle\_\_

|                                  |   |
|----------------------------------|---|
| Function:                        | <ol style="list-style-type: none"> <li>1. Calculates the Start-Stop difference in the TDC values of the temperature measurement ports and accumulates this in the measurement value RAM register (depending on the R_ADD_PTR) respective to the port being measured.</li> <li>2. If the calculated discharge time is too large, then the result is replaced with 0xFFFFFFFF thus indicating overflow.</li> <li>3. If the end of an RDC measurement triggered the DSP, then this ROM routine call will stop the DSP after doing the above tasks. Else, the DSP returns to where the ROM routine was called and continues to execute the firmware.</li> </ol> |
| Input parameters:                | -   |
| Output/Return value:             | The RDC discharge time measurement value is available in the respective RAM (84 – 87)   |
| Prerequisites                    | This function has to be called after a measurement cycle at every resistive port.   |
| Dependency on other header files | None  |
| Function call                    | <pre>push &lt;MSB(RAM_startaddress)&gt; push &lt;LSB(RAM_startaddress)&gt; jsb _ROM_rdc_cycle__</pre>   |
| Temporary memory usage           | DPTR0 is overwritten in this routine.   |
| Example:                         | <pre>push 1 ; bit6 of starting address push TM0; bit 5..0 of starting address jsb _ROM_rdc_cycle (not recommended, use _ROM_tdc_dispatch__ instead)</pre>   |

ROM\_cdc\_initialize

|                                  |  |
|----------------------------------|--|
| Function:                        | This ROM routine clears all the measurement value RAM registers, Addresses 88-95 to 0.   |
| Input parameters:                | The starting address of the RAM registers (88) which are to be cleared, is to be pushed into the RAM address stack before calling this subroutine. |
| Output/Return value:             | -  |
| Prerequisites                    | -  |
| Dependency on other header files | None   |
| Function call                    | push <MSB(RAM_startaddress)><br>push <LSB(RAM_startaddress)><br>jsb ROM_cdc_initialize   |
| Temporary memory usage           | The addresses DPTR0 and _at_DPTR0 are overwritten in this ROM routine.   |
| Example:                         | push 1 ; bit6 of starting address<br>push M0; bit 5..0 of starting address<br>jsb ROM_cdc_initialize   |

ROM\_rdc\_initialize

|                                  |  |
|----------------------------------|--|
| Function:                        | This ROM routine clears all the measurement value RAM registers, Addresses 84-87 to 0.   |
| Input parameters:                | The starting address of the RAM registers (84) which are to be cleared, is to be pushed into the RAM address stack before calling this subroutine. |
| Output/Return value:             | -  |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | push <MSB(RAM_startaddress)><br>push <LSB(RAM_startaddress)><br>jsb ROM_rdc_initialize   |
| Temporary memory usage           | The addresses DPTR0 and _at_DPTR0 are overwritten in this ROM routine.   |
| Example:                         | push 1 ; bit6 of starting address<br>push TM0; bit 5..0 of starting address<br>jsb ROM_rdc_initialize  |

*shiftL\_A\_xx; shiftR\_A\_xx; shiftL\_B\_xx; shiftR\_B\_xx*

|                                  |  |
|----------------------------------|--|
| Function:                        | Function to call a fixed number xx of shift steps of the A-Accu or B-Accu to the left or right   |
| Input parameters:                | A-Accu or B-Accu : Value to shift xx times   |
| Output/Return value:             | shiftL_A_01 A : $A * 2^{(1)}$<br>shiftL_A_02 A : $A * 2^{(2)}$<br>shiftL_A_03 A : $A * 2^{(3)}$<br>shiftL_A_31 A : $A * 2^{(31)}$  |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | jsb shiftL_A_01     jsb shiftR_A_01<br>.....<br>jsb shiftL_A_32     jsb shiftR_A_32<br><br>jsb shiftL_B_01     jsb shiftR_B_01<br>.....<br>jsb shiftL_B_32     jsb shiftR_B_32 |
| Temporary memory usage           |  |
| Example:                         | load b, dp_const_m<br>jsb shiftL_B_08  |

*mult\_xx*

|                                  |  |
|----------------------------------|--|
| Function:                        | Function to call a variable no of multiplication steps   |
| Input parameters:                | B-Accu : Multiplier 1<br>R-Accu : Multiplier 2   |
| Output/Return value:             | A-Accu : Result of multiplication  |
| Prerequisites                    | A-Accu : 0<br>B >= 0<br>R >= 0   |
| Dependency on other header files | None   |
| Function call                    | jsb mult_01<br>..<br>jsb mult_32   |
| Temporary memory usage           |  |
| Example:                         | sign b ; Store sign of multiplier b; take absolute value from b<br>clear a<br>rad __sub_standard_multiplier__<br>jsb mult_15 ; a2 * theta in A-Akku<br>sign a ; Restoring sign of the multiplier to the result |



*div\_xx*

|                                  |   |
|----------------------------------|---|
| Function:                        | div_00 to div_33: Function to call a fixed no of division steps   |
| Input parameters:                | A-Accu : Dividend<br>r : Divisor  |
| Output/Return value:             | B-Accu : Dividend / divisor   |
| Prerequisites                    | B-Accu : 0<br>$0 < a' < 2 * r'$ respectively result b needs to be $0 < b < 2$   |
| Dependency on other header files | None  |
| Function call                    | jsb div_xx  |
| Temporary memory usage           |   |
| Example:                         | <pre> rad M_internal_ref move b, r shiftL b shiftL b ; b : b + 4, to make sure dividend &gt; 2*a rad 0 move r, b rad M0 move a, r ; a = M0 clear b rad 0 ; r = M_internal_ref jsb div_29 ; b = a/r = M0/M_internal_ref </pre> |

ROM\_div\_variable\_\_

|                                  |  |
|----------------------------------|--|
| Function:                        | Function to call a variable no of division steps   |
| Input parameters:                | B-Accu : No of division steps<br>A-Accu : Dividend<br>arguments - 0 : Divisor (RAM-address 82)   |
| Output/Return value:             | B-Accu : Dividend / divisor  |
| Prerequisites                    | $0 < a < 2^r$ respectively result b needs to be $0 < b < 2$  |
| Dependency on other header files | None   |
| Function call                    | jsb_ROM_div_variable__   |
| Temporary memory usage           |  |
| Example:                         | <pre> load a, 3 ; divisor = 3 rad_arguments__; hand over via ramaddress 82 move r, a load a, 4 ; dividend = 4 load b, 23; 23 division steps jsb_ROM_div_variable__; rad RES00 move r, b  ; B : 0x555555__; B : 4/3 = 1.333333 (fpp22)                     </pre> |

\_\_ROM\_mult\_variable\_\_

|                                  |   |
|----------------------------------|---|
| Function:                        | Function to call a variable no of multiplication steps  |
| Input parameters:                | B-Accu : Multiplier 1<br>A-Accu : Number of multiplication steps<br>arguments - 0 : Multiplier 2 (RAM-address 82)   |
| Output/Return value:             | A-Accu : Result of multiplication   |
| Prerequisites                    | Multiplier 1 > 0<br>Multiplier 2 > 0  |
| Dependency on other header files | None  |
| Function call                    | jsb __ROM_mult_variable__   |
| Temporary memory usage           |   |
| Example:                         | <pre> load b, 3 ; multiplier 1 = 3 rad __arguments__ ; hand over via ramaddress 82 move r, b load b, 4 ; multiplier 2 = 4 load a, 32; 32 multiplication steps jsb __ROM_mult_variable__ ; rad RES00 move r, b  ; B : 12__ ; B : 4*3 = 12 (fpp22)                     </pre> |

ROM\_shift\_a\_variable\_\_

|                                  |  |
|----------------------------------|--|
| Function:                        | Function to call a variable no of shift A-Accu steps   |
| Input parameters:                | A-Accu : Value to shift<br>B-Accu : Number of shift steps<br>B > 0 : Left shift<br>B < 0 : Right shift   |
| Output/Return value:             | $A : A * 2 ^ ( B )$  |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | jsb_ROM_shift_a_variable__   |
| Temporary memory usage           |  |
| Example:                         | <pre>rad MyVar                ; a : MyVar; move a, r load b, -4                ; set up 4fold right shift jsb_ROM_shift_a_variable__ rad RES00 move r, a</pre> |

ROM\_shift\_b\_variable\_\_

|                                  |  |
|----------------------------------|--|
| Function:                        | Function to call a variable no of shift B-Accu steps   |
| Input parameters:                | B-Accu : Value to shift<br>A-Accu : Number of shift steps<br>A > 0 : Left shift<br>A < 0 : Right shift   |
| Output/Return value:             | $B : B * 2 ^ ( A )$  |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | jsb_ROM_shift_b_variable__   |
| Temporary memory usage           |  |
| Example:                         | <pre>rad MyVar                ; b : MyVar; move b, r load a, 13                ; set up 13fold left shift jsb_ROM_shift_b_variable__ rad RES00 move r, b</pre> |

\_ROM\_dma\_

|                          |   |
|--------------------------|---|
| Function:                | „Direct Memory Access“ – This routine copies sequential RAM-content from one address-space to another. The number of RAM values to be copied can be specified.  |
| Input parameters:        | B-Accu :Number of values to be copied<br>DPTR1 :Source RAM block address<br>DPTR0 :Destination RAM block address  |
| Output/Return value:     | The contents, i.e. the specified number of values are copied from the source RAM block to the destination RAM block.  |
| Prerequisites            | -   |
| Function call            | jsb __ROM_dma__   |
| Temporary memory usage   | -   |
| RAM permanently changed? | Yes, the destination RAM block  |
| Example                  | <pre> ; Copying the Cratio results to the persistent bank ; in RAM (cells __sub_cdc_CO_Ratio_temp to __sub_cdc_CO_Ratio_temp+5 into ; cels CO_Ratio_RAM to CO_Ratio_RAM + 5) load b, 6 load a, __sub_cdc_CO_Ratio_temp rad DPTR1 move r, a load a, CO_Ratio_RAM rad DPTR0 move r, a jsb _ROM_dma__ </pre> |

*\_ROM\_In\_, \_ROM\_log10\_, \_ROM\_Id\_*

|                          |  |
|--------------------------|--|
| Function:                | Calculation of logarithm<br>log10 (logarithm to base 10)<br>ln (logarithm naturalis)<br>ld (logarithm dualis, to base 2)<br>AccuA = ld( AccuA )<br>for all logarithms, first ld(x) is determined. For log10 and ln afterwards the return value is divided by ld(e(1)) or ld(10)<br>formula for ld(x) of<br><a href="http://de.wikipedia.org/wiki/Logarithmus#Nat.C3.BCrlicher_Logarithmus">http://de.wikipedia.org/wiki/Logarithmus#Nat.C3.BCrlicher_Logarithmus</a> |
| Input parameters:        | A-Accu : Containing parameter for logarithm dualis with 10fpp  |
| Output/Return value:     | A-Accu : Signed 16bit Value  |
| Prerequisites            |  |
| Function call            | jsb __ROM_Id__<br>jsb __ROM_In__<br>jsb __ROM_log10__  |
| Temporary memory usage   | __temporary_variables__ - 3 to __temporary_variables__   |
| RAM permanently changed? |  |
| Example                  | load a, 0xA01; A: 2.500977 (=2561*2 <sup>-10</sup> )<br>jsb __ROM_Id__<br>rad RES00<br>move r, a; A: 1.322492 (0x1528E*2 <sup>-16</sup> )  |

*\_ROM\_signed24\_to\_signed32\_*

|                                  |  |
|----------------------------------|--|
| Function:                        | Type cast from 24bit signed to 32bit signed. This function is to cast a 24bit signed value e to a 32 bit signed value. |
| Input parameters:                | Accu B : Signed 24 bit value   |
| Output/Return value:             | Accu B : Signed 32bit value  |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | <u>_ROM_signed24_to_signed32_</u>  |
| Temporary memory usage           | 1  |
| Example:                         | jsb <u>_ROM_signed24_to_signed32_</u>  |

\_\_ROM\_median\_\_

|                                  |  |
|----------------------------------|--|
| Function:                        | Median-filter: this is a quasi-median-filter. The depth of filter is defined by arguments via stack. Each new Value (X) will be compared with the current median value. Is the new value smaller or equal to the median value the last value at the list will be replaced by X. Otherwise the first value at the list will be replaced by X. Afterwards the complete list is sorted. The value at the very middle of the list is returned as new median. |
| Input parameters:                | B-Accu : Filter input (and output)<br>Stack - 2 : Start address of filter memory<br>Stack - 1 : Middle section of filter memory<br>Stack - 0 : Last address of filter memory<br>The filter order is defined by last address - start address. The necessary memory section has to be reserved for and must not be used otherwise  |
| Output/Return value:             | B-Accu : Filter output (and input)   |
| Prerequisites                    | None   |
| Dependency on other header files | None   |
| Function call                    | jsb __ROM_median__   |
| Temporary memory usage           | __temporary_variables__ - 4 to __temporary_variables__<br>FILTER_START to FILTER_STOP  |
| Example:                         | <pre> CONST FILTER_ORDER 5 CONST FILTER_START 12 CONST FILTER_MIDDLE 12 + (FILTER_ORDER/2) CONST FILTER_STOP 12 + FILTER_ORER rad Filter_input_value move b, r push FILTER_START push FILTER_MIDDLE push FILTER_STOP jsb __ROM_median__                     </pre>   |

\_\_ROM\_cdc\_\_

|                        |  |
|------------------------|--|
| Function:              | This routine contains the subroutine to determine the capacitor ratios (or inverse ratios); depending on the measurement scheme and compensation modes   |
| Input parameters:      | <p>Stack 1 : CDC_FPP number of fractional digits in the result.<br/>           Stack 0 : C_REF_PORT_NUMBER reference port number (0 to 6 for grounded, 0 to 2 for floating).<br/>           A-Accu : __sub_cdc_gain_corr__ factor for gain correction, applied on Mi, 8fpp 0 0 : gain corr = 1 else 1+ (A-Accu)<br/><br/>           FLAG_CDC_INV (Bit 5) of FLAGREG: 0 : inverse capacitance ration, 1 : capacitance rations</p> |
| Output/Return value:   | The addresses __sub_cdc_C0_Ratio_temp to __sub_cdc_C5_Ratio_temp are updated with relevant capacitance ratios (or inverse ones)  |
| Prerequisites          |  |
| Function call          | jsb __ROM_cdc__  |
| Temporary memory usage | Addresses 58 to 72   |
| Example:               | <pre> load2exp a, FLAG_CDC_INV not    a          ; Clearing FLAG_CDC_INV (Bit 5) of FLAGREG rad    FLAGREG and    r, a  load   a, 0x40 ; default gain_corr 1.25  rad    27         ; Stack - 1 ---&gt; Number of fpp in the result rad    0; Stack - 0 -&gt; Ref. port number  jsb __ROM_cdc__ ; Calling ROM routine for Ratio calculation rad __sub_cdc_C1_Ratio_temp move a, r; save Ratio from CDC           </pre>           |



\_\_ROM\_rdc\_\_ ; \_\_ROM\_rdc\_inverse\_\_

|                                  |   |
|----------------------------------|---|
| Function:                        | Subroutine to calculate the resistance ratios or inverse ratios for the temperature measurement ports<br>TM0 -> Internal Reference Resistance Port;<br>TM1 -> External Resistance Port, PT0<br>TM2 -> External Resistance Port, PT1<br>TM3 -> Internal Sensor Resistance Port   |
| Input parameters:                | Stack 0 : FPP_RRATIO fixed point position of result<br>A-Accu : REF_PORT_NUMBER reference, 0 = internal 1=external  |
| Output/Return value:             | <u>__sub_rdc_R0_Ratio_temp</u> TM0/Ref or Ref/TM0<br><u>__sub_rdc_R1_Ratio_temp</u> TM1/Ref or Ref/TM1<br><u>__sub_rdc_R2_Ratio_temp</u> TM2/Ref or Ref/TM2<br><u>__sub_rdc_R3_Ratio_temp</u> TM3/Ref or Ref/TM3  |
| Prerequisites                    | None  |
| Dependency on other header files | #include <memory.h>   |
| Function call                    | jsb <u>__ROM_rdc__</u><br>jsb <u>__ROM_rdc_inverse__</u>  |
| Temporary memory usage           | <u>__temporary_variables__</u> - 8 to <u>__temporary_variables__</u>  |
| Example:                         | load a, 0 ; Reference Port Number = 0<br>rad 25 ; Number of fpp in the result<br>; A-Akku: Reference Port Number (0 ... 3)<br>; Stack - 0: Value of fpp of result = <u>__rdc_fpp__</u><br>jsb <u>__ROM_rdc__</u> ; Determine RDC ratios<br>;jsb <u>__ROM_rdc_inverse__</u> ; To determine inverse-RDC ratios<br>rad <u>__sub_rdc_R1_Ratio_temp</u> ; save ratio from RDC<br>move a, r |

***\_ROM\_memory\_ (read/write volatile memory)***

|                                  |  |
|----------------------------------|--|
| Function:                        | Functions for memory NVRAM access<br>read functions: * signed/unsigned, 8/16/24/32bit RAM into Accu A or B<br>write functions: * 8/16/24/32 bit (write data from RAM cell mem_data)<br>The address in RAM cell mem_add is automatically incremented with read/write operations.  |
| Input parameters:                | -  |
| Output/Return value:             | Accu-A or Accu-B contains the specified value from the NVRAM   |
| Prerequisites                    | None   |
| Dependency on other header files | #include <memory.h>  |
| Function call                    | jsb _ROM_memory_rd_a_32b__<br>jsb _ROM_memory_rd_a_u24b__ ; jsb _ROM_memory_rd_a_s24b__<br>jsb _ROM_memory_rd_a_u16b__ ; jsb _ROM_memory_rd_a_s16b__<br>jsb _ROM_memory_rd_a_u08b__ ; jsb _ROM_memory_rd_a_s08b__<br>jsb _ROM_memory_rd_b_32b__<br>jsb _ROM_memory_rd_b_u24b__ ; ...<br><br>_ROM_memory_wr_32b__ ; _ROM_memory_wr_24b__<br>_ROM_memory_wr_16b__ ; _ROM_memory_wr_08b__ |
| Temporary memory usage           |  |

|                        |  |
|------------------------|--|
| <p><b>Example:</b></p> | <pre> Single read: load a, &lt;read_mem_add&gt; ; mem_add : &lt;read_mem_add&gt; rad mem_add move r, a jsb _ROM_memory_rd_a_u24b__  Single write: load a, MEM_WE ; enable memory writing                     ;(disable write protection)  rad mem_ctrl move r, a  load a, &lt;write_address&gt; ; mem_add : &lt;write_mem_add&gt; rad mem_add move r, a load a, &lt;data to store&gt; ; mem_data : &lt;data to store&gt; rad mem_data move r, a jsb _ROM_memory_wr_32b__  ; optional: set write protection load a, MEM_WR_PROTECT rad mem_ctrl move r, a  Autoincrement: load a, 900 ; address (decimal) 900 rad mem_add move r, a load a, 0x012345; data : 0x012345 rad mem_data move r, a jsb _ROM_memory_wr_24b__                     ; add 900: 0x45                     ; add 901: 0x23                     ; add 902: 0x01                     ; mem_add after this operation : 903 </pre> |
|------------------------|--|

***\_ROM\_MEMORY\_RECALL/STORE***

|                                  |  |
|----------------------------------|--|
| Function:                        | <p>Functions for nonvolatile memory access<br/>           All nonvolatile accesses are secured by RAM cell mem_ctrl. If a store or a recall should be activated this has to be enabled by dedicated codes to mem_ctrl to prevent such actions by accident.<br/>           Store: mem_ctrl : MEM_STORE (0x2d00)<br/>           Recall: mem_ctrl : MEM_RECALL (0x5900)</p> |
| Input parameters:                |  |
| Output/Return value:             |  |
| Prerequisites                    | None   |
| Dependency on other header files | <memory.h>   |
| Function call                    | <pre>jsb_ROM_memory_store__ jsb_ROM_memory_recall__</pre>  |
| Temporary memory usage           |  |
| Example                          | <pre>Store: load a, MEM_STORE rad mem_ctrl move r, a jsb_ROM_memory_store__  Recall: load a, MEM_RECALL rad mem_ctrl move r, a jsb_ROM_memory_recall__</pre>   |

***\_ROM\_2pt\_calibration***

|                               |  |
|-------------------------------|--|
| <p>Function:</p>              | <p>This function performs a two point calibration with the given set of input values. The mathematical formula of the function is given below (calculation of xi):</p> $xi = \frac{(xi\_at\_ccp1 - xi\_at\_ccp2)}{(ci\_at\_ccp1 - ci\_at\_ccp2)} \cdot (ci - ci\_at\_ccp1) + xi\_at\_ccp1$ <p>The input values, i.e., xi_at_ccp1, xi_at_ccp2, ci_at_ccp1, ci_at_ccp2 are defined in the calibration data</p>   |
| <p>Input parameters:</p>      | <p>A-Accu :  FPP_ci  -  FPP_ccp <br/>                 FPP_ci : No. of fractional digits in the input capacitance ratio<br/>                 FPP_ccp : No. of fractional digits in the calibration value of capacitance ratio<br/>                 DPTR0 : Start address of the RAM containing the constants in 4 consecutive addresses in the following order:<br/>                 xi_at_ccp1, xi_at_ccp2, ci_at_ccp1, ci_at_ccp2<br/>                 Stack-0 : RAM address of Input Capacitance Ratio ci</p> <p>NOTE: It is a must that ALL the 4 calibration values have the same fpp !!!</p>                          |
| <p>Output/Return value:</p>   | <p>A-Accu :xi with FPP_ccp fractional digits</p>   |
| <p>Prerequisites</p>          | <p>All the four calibration values must have the same fpp.</p>   |
| <p>Function call</p>          | <p><i><b>_ROM_2pt_calibration__</b></i></p>  |
| <p>Temporary memory usage</p> | <p><i><b>__temporary_variables__</b></i> - 8 to <i><b>__temporary_variables__</b></i></p>  |
| <p>Example:</p>               | <pre> load    a, FPP_difference; A Accu : ( FPP_ci  -  FPP_ccp ) rad     xi_at_ccp1; Start address containing the list of ; calibration values  move    b, r rad     DPTR0          ; DPTR0 now contains starting address ; of the calibration values move    r, b rad     C1_ratio ; Inputs : ; A Accu : (FPP_ci - FPP_ccp) ; DPTR0 : Start address of the Calibration values in RAM ; Stack-0 : RAM address of Input Capacitance Ratio ci ; Output : ; A Accu : xi with FPP_ccp fractional digits jsb     _ROM_2PT_Calibration rad     2pt_result; Storing the returned value in RAM move    r, a                 </pre> |

*\_ROM\_polynomial\_3rd\_degree*

|                                  |   |
|----------------------------------|---|
| Function:                        | <p>This function calculates the third degree polynomial for a given input value, with known coefficients. The mathematical formula of the function is given below:<br/> <math display="block">an = ((cc3n / xi + cc2n)/xi + cc1n)/xi + cc0n</math>                     where cc3n, cc2n, cc1n and cc0n are the coefficients of the third degree polynomial. xi is the given input value which can be a capacitance or resistance ratio for example.<br/>                     This can be used for capacitance and resistance polynomials of this form.</p>  |
| Input parameters:                | <p>DPTR0: start address of the memory containing the following :</p> <ul style="list-style-type: none"> <li>cc3n: 3rd degree coefficient</li> <li>cn_div3n: division steps for cc3n</li> <li>cc2n: 2nd degree coefficient</li> <li>cn_div2n: division steps for cc2n</li> <li>cc1n: 1st degree coefficient</li> <li>cn_div1n: division steps for cc1n</li> <li>cc0n: constant coefficient</li> </ul> <p>A-Accu: Input Capacitance or Resistance Ratio (or Inverse Ratio)</p>  |
| Output/Return value:             | A-Accu: Result of the polynomial  |
| Prerequisites                    | None  |
| Dependency on other header files | None  |
| Function call                    | jsb _ROM_polynomial_3rd_degree  |
| Temporary memory usage           | __temporary_variables__   |
| Example:                         | <p>The following example calculated temperature using the temperature polynomial</p> <pre> rad    Ratio_temp ; Resistance ratio move   a, r          ; A-Akku contains R_ratio  load   b, cc3n_address ; Start address containing the list         ; of calibration values rad    DPTR0 ; DPTR0 now contains starting address move   r, b          ; of the coefficient and steps list  ; Input : DPTR0: start address of the list memory ; A-Akku: Input Capacitance or Resistance Ratio ; (or Inverse Ratio) jsb    _ROM_polynomial_3rd_degree; A = theta rad    theta move   r, a                     </pre> |

*\_ROM\_polynomial\_4th\_degree*

|                                  |   |
|----------------------------------|---|
| Function:                        | <p>This function calculates the third degree polynomial for a given input value, with known coefficients. The mathematical formula of the function is given below:<br/> <math display="block">an = (((cc4n / xi + cc3n)/xi + cc2n)/xi + cc1n)/xi + cc0n</math>                     where cc4n, cc3n, cc2n, cc1n and cc0n are the coefficients of the third degree polynomial. xi is the given input value which can be a capacitance or resistance ratio for example.<br/>                     This can be used for capacitance and resistance polynomials of this form.</p>                                    |
| Input parameters:                | <p>DPTR0: start address of the memory containing the following :</p> <ul style="list-style-type: none"> <li>cc4n: 4th degree coefficient</li> <li>cn_div4n: division steps for cc4n</li> <li>cc3n: 3rd degree coefficient</li> <li>cn_div3n: division steps for cc3n</li> <li>cc2n: 2nd degree coefficient</li> <li>cn_div2n: division steps for cc2n</li> <li>cc1n: 1st degree coefficient</li> <li>cn_div1n: division steps for cc1n</li> <li>cc0n: constant coefficient</li> </ul> <p>A-Accu: Input Capacitance or Resistance Ratio (or Inverse Ratio)</p>   |
| Output/Return value:             | A-Accu: Result of the polynomial  |
| Prerequisites                    | None  |
| Dependency on other header files | None  |
| Function call                    | jsb _ROM_polynomial_4th_degree  |
| Temporary memory usage           | __temporary_variables__-3 to __temporary_variables__  |
| Example:                         | <p>The following example calculated temperature using the temperature polynomial</p> <pre> rad    Ratio_temp ; Resistance ratio move   a, r          ; A-Akku contains R_ratio  load   b, cc4n_address ; Start address containing the list         ; of calibration values rad    DPTR0 ; DPTR0 now contains starting address move   r, b          ; of the coefficient and steps list  ; Input : DPTR0: start address of the list memory ; A-Akku: Input Capacitance or Resistance Ratio ; (or Inverse Ratio) jsb    _ROM_polynomial_4th_degree; A = theta rad    theta move   r, a                     </pre> |

\_\_ROM\_pulse\_\_

|   |   |
|---|---|
| <p>Function:</p>                        | <p>This function determines the pulse output, given two input co-ordinates (result_1, pulse_out_1) and (result_2, pulse_out_2) and the current result_n. The input coordinates are copied from the NVRAM, given the address and must fulfill the pre-requisite.</p> <p>The result_n, result_1 and result_2 can be the result from the capacitance measurement (like Humidity, Pressure etc) or the temperature measurement (theta). The "result_n" is converted into "Pulse_out" which can then be assigned to PULSE0 or PULSE1 output.</p> <p>The mathematical formula of the function is given below:</p> $\text{Pulse\_out} = \frac{(\text{pulse\_out\_1} - \text{pulse\_out\_2})}{(\text{result\_1} - \text{result\_2})} * (\text{result\_n} - \text{result\_1}) + (\text{pulse\_out\_1})$ <p>Additionally Pulse_out is limited to the range between pulse_out_min and pulse_out_max.</p> <p>The constants pulse_out_1, pulse_out_2, result_1, result_2, pulse_out_min and pulse_out_max are defined in the calibration memory and have to be copied to the RAM before calling this function.</p> <p>!!! <b>NOTE</b> : result_n and result_1 must have the same format.</p> |
| <p>Input parameters:</p>                | <p>A-Accu: Value of result_n<br/>         B-Accu: 10-bit NVRAM Start Address containing the list of calibration values</p>  |
| <p>Output/Return value:</p>             | <p>A-Accu: Pulse_out value as integer</p>   |
| <p>Prerequisites</p>                    | <p>result_1: 4 bytes (Same fpp as result_2 and result_n)<br/>         result_2: 4 bytes (Same fpp as result_1 and result_n)<br/>         pulse_out_1: 2 bytes (Integer)<br/>         pulse_out_2: 2 bytes (Integer)<br/>         pulse_out_max: 2 bytes (Integer)<br/>         pulse_out_min: 2 bytes (Integer)</p>   |
| <p>Dependency on other header files</p> | <p>-</p>  |
| <p>Function call</p>                    | <p>jsb __ROM_pulse__</p>  |
| <p>Temporary memory usage</p>           | <p>__temporary_variables__ - 8 to __temporary_variables__</p>   |
| <p>Example:</p>                         | <pre>rad    Z_result ; Value to be given as pulse output move   a, r  load   b, result1_NVaddress ; Starting address in NVRAM       ; containing constants  ; Input : B-Accu: 10 bit NVRAM Address ;        A-Accu : Value of result_n jsb    __ROM_pulse__; Output in A-Accu is an integer rad    PULSE0 ; Pulse output on PULSE0 move   r, a</pre>  |



*\_ROM\_pulse\_loaded\_cal\_vals*

|                                  |   |
|----------------------------------|---|
| Function:                        | This ROM routine has the same functionality as the <code>_ROM_pulse__</code> routine. Only difference is that the constant values have to be copied from the calibration NVRAM memory to the RAM by the firmware, before this routine is called.  |
| Input parameters:                | <p>A-Akku: Value of result_n<br/>                 DPTR0: Start address of the RAM containing the constant values in 6 consecutive addresses in the following order:<br/>                 result_1 (Same fpp as result_2 and result_n) result_2 Same fpp as result_1 and result_n pulse_out_1 (Integer)<br/>                 pulse_out_2 (Integer)<br/>                 pulse_out_max (Integer)<br/>                 pulse_out_min (Integer)</p> |
| Output/Return value:             | A-Akku: Pulse_out value as integer  |
| Prerequisites                    | The coordinate values must be copied from the NVRAM to a RAM space by the firmware.   |
| Dependency on other header files | -   |
| Function call                    | <code>jsb _ROM_pulse_loaded_cal_vals</code>   |
| Temporary memory usage           | <code>__temporary_variables__ - 8 to __temporary_variables__</code>   |
| Example:                         | <pre> rad    Z_result ; Value to be given as pulse output move   a, r  load  b, result1_RAMaddress ; Starting address in RAM       ; containing constants  rad  DPTR0 move r, b ; Input : DPTR0: Starting RAM Address ; A-Accu : Value of result_n jsb  _ROM_pulse_loaded_cal_vals       ; Output in A-Accu is an integer rad  PULSE0 ; Pulse output on PULSE0 move  r, a                     </pre>  |

\_\_ROM\_NVblock\_copy\_\_

|                                  |   |
|----------------------------------|---|
| Function:                        | Copy a block of data from NVRAM to RAM  |
| Input parameters:                | <p>DPTR0: Start address of the RAM<br/>           B-Accu : Starting address of the NVRAM<br/>           A-Accu: Count of the number of values to be copied<br/>           SIGNED_VALUE_NV in FLAGREG must be set (for reading signed values) or cleared (for reading unsigned values) - relevant only for 24/16/08 bit values</p>   |
| Output/Return value:             | RAM contains a copy of the specified number of values from the NVRAM  |
| Prerequisites                    | None  |
| Dependency on other header files | None  |
| Function call                    | <pre>jsb _ROM_NVblock_copy_32b_; jsb _ROM_NVblock_copy_24b_; jsb _ROM_NVblock_copy_16b_; jsb _ROM_NVblock_copy_08b_;</pre>  |
| Temporary memory usage           | <u>__temporary_variables__</u>  |
| Example:                         | <p>This example copies 4 values, 32 bits each from the NVRAM to the RAM address starting at xi_at_ccp1:</p> <pre>load    a, RAM_address         ; DPTR0 &lt;-- starting RAM address rad DPTR0 move    r, a load2exp a, 2 ; Count = 4 load    b, NVRAM_address ; Starting NVRAM address  ; Subroutine to copy values from NVRAM -&gt; RAM, ;Returns current NVRAM address in B-Akku jsb     _ROM_NVblock_copy_32b_</pre> |

***\_ROM\_capacitance\_polynomial***

|                               |   |
|-------------------------------|---|
| <p>Function:</p>              | <p>This function calculates the value of the capacitance polynomial,<br/> <math>Z\_result = (((a2 * theta) + a1) * theta) + a0</math><br/>                 given the capacitance ratio or inverse (Cratio) and temperature (theta).<br/>                 a2, a1 and a0 are 3rd degree polynomials of the Cratio value</p> <p><math>a_n = ((cc3n / xi + cc2n)/xi + cc1n)/xi + cc0n</math><br/>                 where cc3n .... cc0n are the coefficients of the third degree polynomial in the NVRAM</p>   |
| <p>Input parameters:</p>      | <p>DPTR0: Address of the capacitance ratio or inverse(Cratio)<br/>                 DPTR1: Address of the temperature (theta)<br/>                 B-Accu Starting address of coefficient values in NVRAM<br/>                 Arg_6: Z_min<br/>                 Arg_7: Z_max<br/>                 FLAGREG, Bit 7 (LIN_3BYTE_COEFF) :<br/>                 1 -&gt; 3 bytes in each of the coefficients<br/>                 0 -&gt; 4 bytes in each of the coefficients</p>  |
| <p>Output/Return value:</p>   | <p>A-Accu: Z_result = result of the polynomial</p>  |
| <p>Prerequisites</p>          | <p>-</p>  |
| <p>Function call</p>          | <p><i><b>_ROM_capacitance_polynomial__</b></i></p>  |
| <p>Temporary memory usage</p> | <p><i><b>__temporary_variables__</b></i> - 8 to <i><b>__temporary_variables__</b></i><br/>                 Arg_0 to Arg_7</p>   |
| <p>Example:</p>               | <pre> ; Copying Z_min and Z_max to Arg_6 and Arg_7 load  a, Arg_6 ; DPTR0 &lt;-- starting argument ; memory address (RAM) rad   DPTR0 move  r, a load2exp a, 1 ; Count = 2 load  b, NV_Cal_vals; Starting address of Calibration ; values in NVRAM jsb   _ROM_NVblock_copy_32b_; Subroutine to copy Z_min ; and Z_max values from NVRAM -&gt; argument RAM  ;-----  load  a, C1_ratio  rad   DPTR0 ; DPTR0: Address of the capacitance ; ratio or inverse(Cratio) move  r, a load  a, theta ; DPTR1: Address of the temperature rad   DPTR1 move  r, a load  b, NV_QUADRATIC_COEFFS; Address of the coefficients ;in NVRAM in B-Accu jsb   _ROM_capacitance_polynomial rad   Z_result move  r, a                 </pre> |

***\_ROM\_capacitance\_polynomial\_4d***

|                        |  |
|------------------------|--|
| Function:              | This function calculates the value of the capacitance polynomial,<br>$Z\_result = (((a2 * theta) + a1) * theta) + a0$<br>given the capacitance ratio or inverse (Cratio) and temperature (theta).<br>a2, a1 and a0 are 4th degree polynomials of the Cratio value<br>$a_n = (((cc4n / xi + cc3n)/xi + cc2n)/xi + cc1n)/xi + cc0n$<br>where cc4n .... cc0n are the coefficients of the fourth degree polynomial in the NVRAM  |
| Input parameters:      | DPTR0: Address of the capacitance ratio or inverse(Cratio)<br>DPTR1: Address of the temperature (theta)<br>B-Accu Starting address of coefficient values in NVRAM<br>Arg_6: Z_min<br>Arg_7: Z_max  |
| Output/Return value:   | A-Accu: Z_result = result of the polynomial  |
| Prerequisites          | -  |
| Function call          | <code>_ROM_capacitance_polynomial__</code>   |
| Temporary memory usage | <code>__temporary_variables__ - 8 to __temporary_variables__</code><br>Arg_0 to Arg_7  |
| Example:               | <pre> ; Copying Z_min and Z_max to Arg_6 and Arg_7 load  a, Arg_6 ; DPTR0 &lt;-- starting argument ; memory address (RAM) rad   DPTR0 move  r, a load2exp a, 1 ; Count = 2 load  b, NV_Cal_vals; Starting address of Calibration ; values in NVRAM jsb   _ROM_NVblock_copy_32b_; Subroutine to copy Z_min ; and Z_max values from NVRAM -&gt; argument RAM  ;-----  load  a, C1_ratio  rad   DPTR0 ; DPTR0: Address of the capacitance ; ratio or inverse(Cratio) move  r, a load  a, theta ; DPTR1: Address of the temperature rad   DPTR1 move  r, a load  b, NV_QUADRATIC_COEFFS; Address of the coefficients ;in NVRAM in B-Accu jsb   _ROM_capacitance_polynomial_4d rad   Z_result move  r, a         </pre> |

### Assembly Programs

The PCap04 assembler is a multi-pass assembler that translates assembly language files into HEX files as they will be downloaded into the device. For convenience, the assembler can include header files. The user can write his own header files but also integrate the library files as they are provided by **ams**. The assembly program is made of many statements which contain instructions and directives. In the former section we explained the instructions in detail. In the following sections we describe the directives and some sample code.

Each line of the assembly program can contain only one directive or instruction statement. Statements must be contained in exactly one line.

#### Symbols

A symbol is a name that represents a value. Symbols are composed of up to 31 characters from the following list:

A - Z, a - z, 0 - 9, \_

Symbols are not allowed to start with numbers. The assembler is case sensitive, so care has to be taken for this.

#### Numbers

Numbers can be specified in hexadecimal or decimal. Decimal have no additional specifier. Hexadecimals are specified by leading "0x".

#### Expressions and Operators

An expression is a combination of symbols, numbers and operators. Expressions are evaluated at assembly time and can be used to calculate values that otherwise would be difficult to be determined.

The following operators are available with the given precedence:

| Level | Operator | Description                          |
|-------|----------|--------------------------------------|
| 1     | ()       | Brackets, specify order of execution |
| 2     | * /      | Multiplication, Division             |
| 3     | + —      | Addition, Subtraction                |

Example:

```
CONST value 1
equal ((value + 3)/2)
```

### Directives

The assembler directives define the way the assembly language instructions are processed. They also provide the possibility to define constants, to reserve memory space and to control the placement of the code. Directives do not produce executable code.

The following table provides an overview of the assembler directives.

**Figure 134:**  
Overview of Assembler Directives

| Directive | Description  | Example   |
|-----------|--|---|
| CONST     | Constant definition, CONST [name] [value]<br>value might be a number, a constant, a sum of both  | CONST Slope 42<br>CONST Slope constant + 1                                    |
| LABEL:    | Label for target address of jump instructions. Labels end with a colon. All rules that apply to symbol names also apply to labels.   | jsb LABEL1<br>LABEL1:<br>...  |
| ;         | Comment, lines of text that might be implemented to explain the code. It begins with a semicolon character. The semicolon and all subsequent characters in this line will be ignored by the assembler. A comment can appear on a line itself or follow an instruction.   | ; this is a comment   |
| org       | Sets a new origin in program memory for subsequent statements.   | org 0x23  |
| equal     | Insert three bytes of user defined data in program memory, starting at the address as defined by org.  | equal 0x332211<br>; write 0x11 to address 0x23,<br>; 0x22 to address 0x24 ... |
| #device   | Directive definition, #device [dev_name] might be the name of the used device.<br><br>Defines which Library is used for the assembler and defines the subdirectory \lib\[dev_name] for the included header or library file.  | #device PCap03-Z  |
| #include  | Include the header or library file named in the brackets < > or quotation marks " ". The code will be added at the line of the include command.<br><br>Names in brackets refer to the <b>ams</b> library with the defined subdirectory \lib\[dev_name]. Without using #device directive it refers to the fixed subdirectory \lib.<br><br>In quotation marks the might be just the file name in case it is in the same folder as the program, but also the complete path. | #include <rdc.h><br>#include "rdc.h"  |

| Directive  | Description   | Example   |
|--|---|---|
| <code>#ifdef</code><br><code>#elseif</code><br><code>#endif</code> | Directive to implement code or not, dependig on the value of the symbol following the <code>#ifdef</code> directive. Use e.g. to include header files only once into a program. | <code>#ifdef __standard_h__</code><br><code>#else</code><br><code>#define __standard_h__</code> |
| <code>#define</code>   | Defines a symbol that will be interpreted as true when being analysed by the <code>#ifdef</code> directive  | <code>...</code><br><code>#endif</code>   |

## Sample Code

In the following we show some sample code for programming loops in the various kinds, for the use of the load instruction and the rotate instruction.

### ***“for” Loop***

| Assembler  | C-Equivalent                        | Comment  |
|--|-------------------------------------|--|
| load a, max<br>not a<br>inc a<br>rad index<br>move r, a<br>do:<br>;{..<br>rad index<br>inc r<br>jCarC do | for(index=-max; index < 0; index++) | max : number of repetitions<br>2nd complement for max (~max+1)<br><br>store (-max) to index<br><br>loop body<br><br>loop increment<br>repeat while index < 0 |

### ***“while” Loop***

| Assembler  | C-Equivalent                    | Comment   |
|--|---------------------------------|---|
| do:<br>rad expression<br>move a, r<br>jEQ done<br>;{..<br>clear a<br>jEQ do<br>done; | while ( expression )<br>{..<br> | activate Status Flags for „expression“. Jump if expression == 0<br>loop body<br>unconditional jump without writing to program counter stack |

### ***“do - while” Loop***

| Assembler  | C-Equivalent                      | Comment   |
|--|-----------------------------------|---|
| do:<br>;{..<br>rad expression<br>move a, r<br>jNE do | do<br>{..<br>while ( expression ) | loop body<br><br>activate Status Flags<br>jump if expression != 0 |



**“do - while” with 2 pointers**

| Assembler  | C-Equivalent   | Comment  |
|--|--|--|
| load a, MW7<br>rad loopLimit<br>move r, a<br>load a, MW0<br>rad DPTR0<br>move r, a<br>load a, RES0<br>rad DPTR1<br>move r, a<br>do:<br>rad _at_DPTR0<br>move a, r<br>rad _at_DPTR1<br>move r, a<br>rad loopLimit<br>move a, r<br>rad DPTR1<br>inc r<br>rad DPTR0<br>inc r sub a, r<br>jCarS do | <pre> loopLimit = *MW7  ptrSource = *MW0;  ptrSink = *Res0;  do { *ptrSink++ = *ptrSource++ }                     </pre> <p>while ( ptrSource &lt;= MW7)</p> | load max-address for ptrSource<br><br>load ptrSource with source address<br><br>load ptrSink with sink address<br><br>loop body<br>load value from source<br><br>write value to sink<br><br>write max-address to a<br><br>increment sink address<br><br>increment source address<br>limitLoop – ptrSource<br>repeat loop if ptrSource <= max-address |

**Rotate Right A to B**

To rotate a value right from Akku A to Akku B, Akku B and R must be set to zero. Afterwards with each mult command a single „rotate right from A to B“ is done. This function could be used e.g. to shift a 8-bit value to the highest byte in the register.

| Assembler  | C-Equivalent  | Comment |
|--|---|---------|
| load a, 0xa3<br>clear b<br>move r, b<br>mult ;(8x)<br>mult<br>..<br>mult | <pre> A = &lt;U8bC&gt; b = a &lt;&lt; 40                     </pre> |         |

**Libraries**

The PICOCAP assembler offers the possibility to implement library files. With these libraries the firmware can be written in a modular manner. Common library files are for definitions of variable and constant names.

When the DSP has to be programmed by the user for a specific application or when the firmware ought to be modified, these library functions can be simply integrated into the application program without any major tailoring. They save programming effort for known, repeatedly used, important functions. Some library files are interdependent on other file(s) from the library.

The library functions are called header files (they have \*.h extension) in the assembler software and have to be included in the main \*.asm program. The path for the library files should be \lib\[dev\_name] in the folder where the assembler is.

The following are the header files that we supply together with the assembler as part of the evaluation kit.

| Device Related  |
|---|
| pcap_standard.h<br>PCap04_ROM_addresses_standard.h<br>pcap_config.h |

The input parameters, output parameters, effect on RAM contents etc. for each of these library functions are explained in the tables below.

**Note(s):** In the standard firmware and in all the library files, the notation “ufdN” is used as a comment. This shows if the parameter is signed or unsigned and the number of fractional digits in the number, N. For e.g. ufd21 indicates that the parameter is an unsigned fixed point number with 21 fractional digits. If the u at the beginning is missing, it is a signed number.

**pcap\_standard.h**

|                         |  |
|-------------------------|--|
| Function:               | <p>This is a standard library for PCap04 firmware projects. It contains the major address mappings and constant names for the PCap04.<br/> <b>Note:</b> This file should be always included. It contains no commands, so no program space is wasted</p>  |
| Definitions (examples): | <pre> ... ;- Temp. Variables and Arguments Def. -----*/ CONST __arguments__ 82 CONST __number_of_arguments__ 10 CONST __temporary_variables___arguments__ - __number_of_arguments__  CONST Arg_0    __arguments__ - 8 CONST Arg_1    __arguments__ - 7 ... ;- RAM-Addresses OUT (&amp;IN) -----*/ CONST FLAGREG96 CONST RES00    97 CONST RES01    98 CONST RES02    99 ...                 </pre> |

**PCap04\_ROM\_addresses\_standard.h**

|                         |  |
|-------------------------|--|
| Function:               | <p>This file declares the jump labels for ROM routines</p>   |
| Definitions (examples): | <pre> ... CONST _ROM_dma__0x5d1; 1489 CONST _ROM_In__0x5e1 ; 1505 CONST _ROM_log10__0x5fc; 1532 ... CONST _ROM_cdc__0x7d4; 2004 CONST __sub_cdc_C0_Ratio_temp0x3A; 58 CONST __sub_cdc_C1_Ratio_temp0x3B; 59 ...                 </pre> |

***pcap\_config.h***

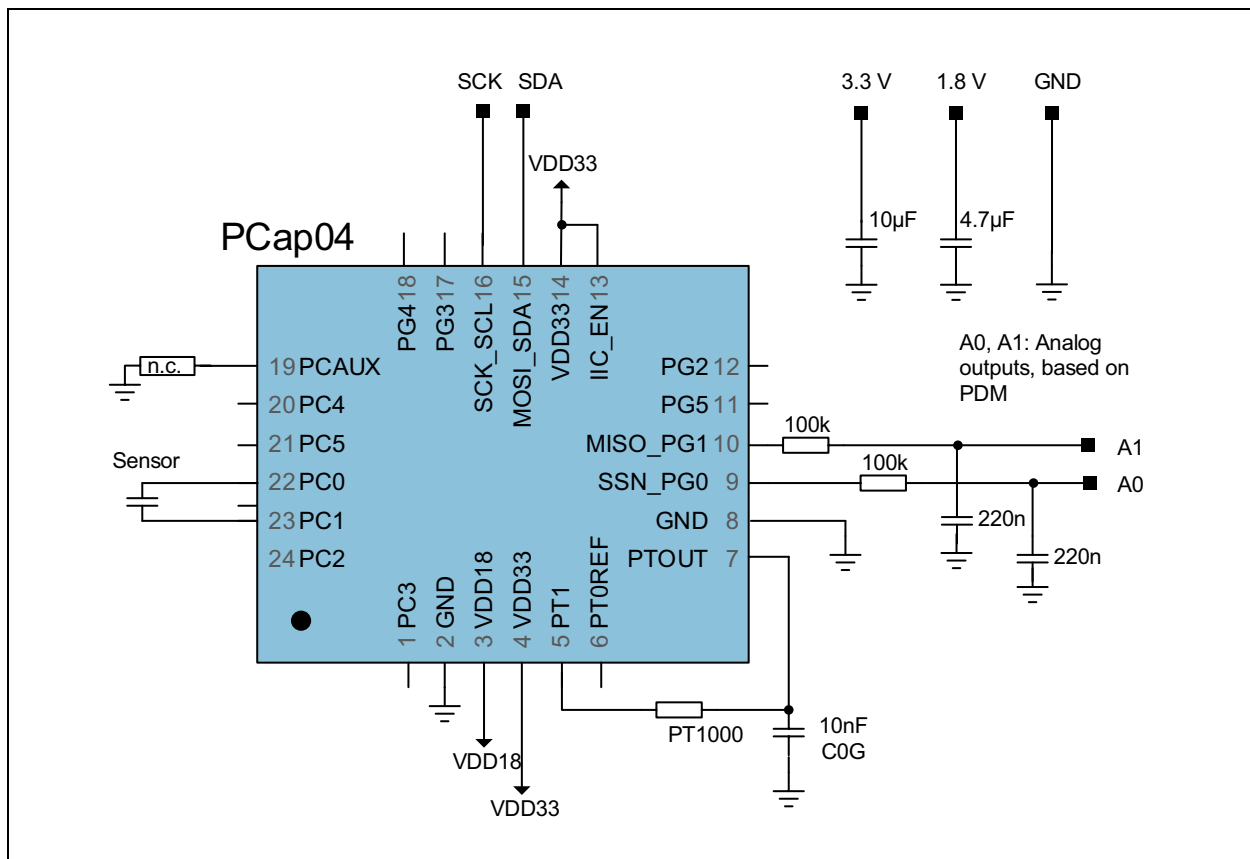
|                         |   |
|-------------------------|---|
| Function:               | Configuration register addresses and names for pcap04. It is for convenience, contains no commands and so no program space is wasted.   |
| Definitions (examples): | <pre> CONST CFG_ADD_OFFSET 960 ; Start address of config in ; NVRAM ; ----- Register 13 CONST CFG_ADD_C_TRIG_SEL 13 + CFG_ADD_OFFSET ... ; ----- Register 42 CONST CFG_ADD_EXTERNAL_FLAGS 42+ CFG_ADD_OFFSET CONST CFG_BM_C_MEDIAN_EN0x01 ; Bit Mask for C_MEDIAN_EN ; Enable median filter for CDC values ... </pre> |

## Application Information

### Schematic

Pcap04 needs only a few external components for operation. Of importance is a sufficient buffering of the supply voltage. We recommend 10µF for VDD33 and 4.7µF for VDD18. A simple RC network may be used for integration of the PDM outputs to generate analog output signal.

**Figure 135:**  
**Typical Schematics with I<sup>2</sup>C Interface and PDM Analog Outputs**

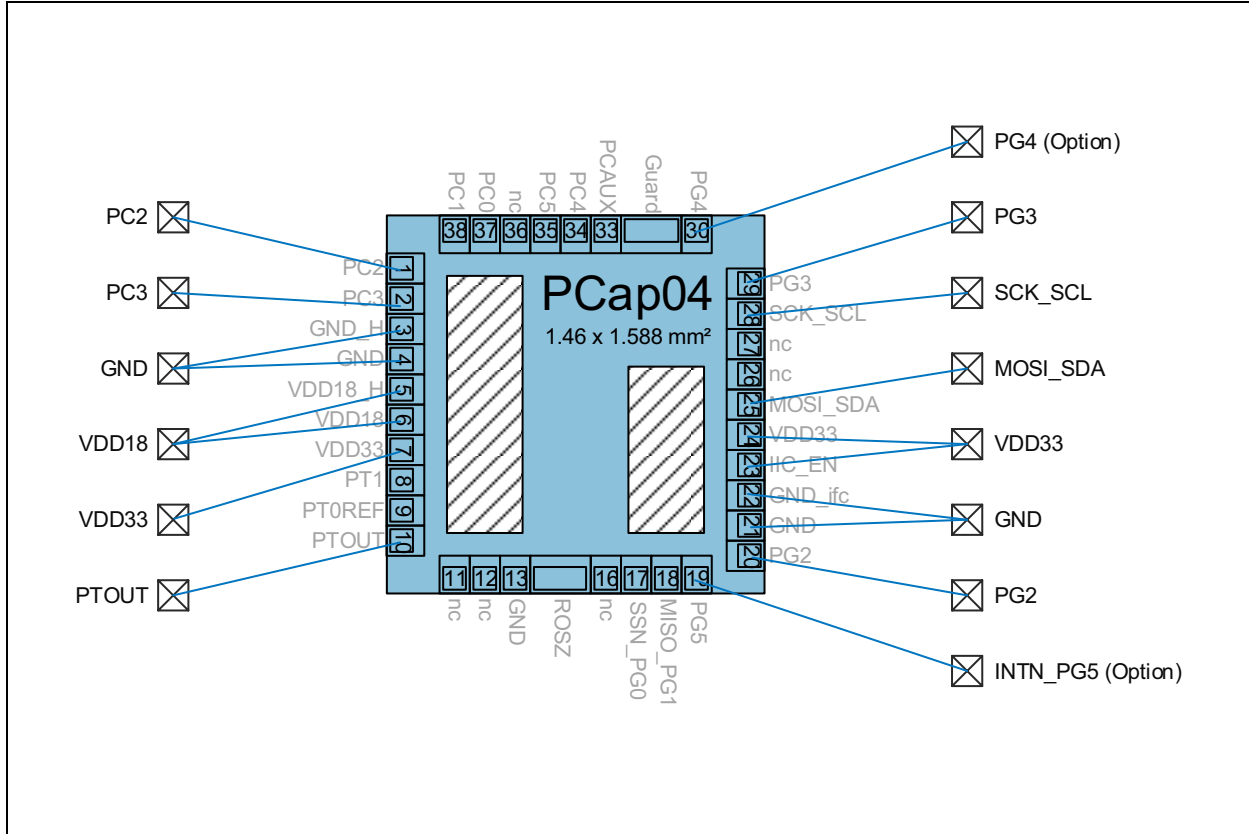


### Minimized Bonding

PCap04 is designed that for compact one-sensor applications the die may be bonded on two sides only.

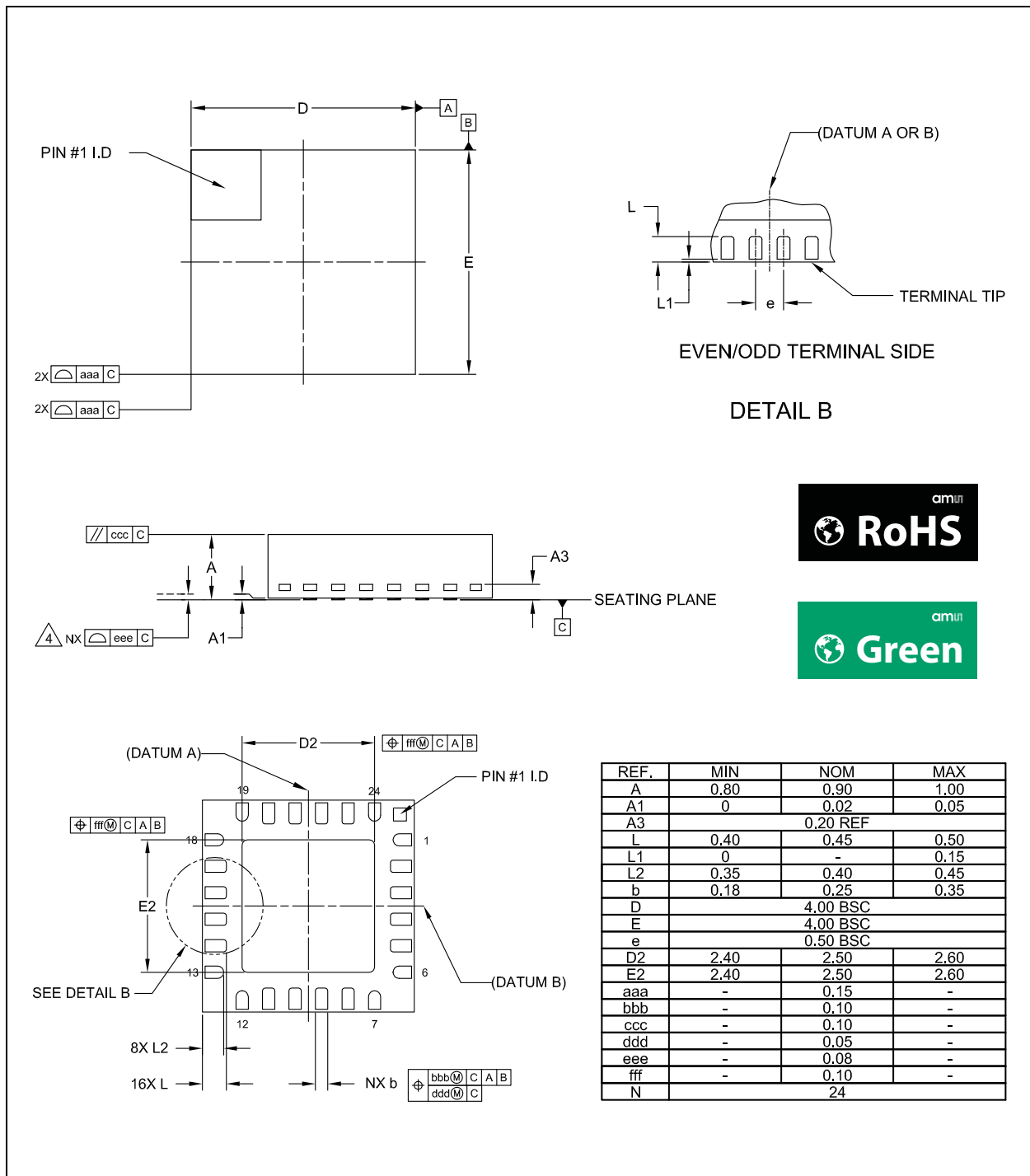
Minimum: GND: #3, 4, 21,22, VDD33: #7, 24, VDD18: #5, 6 have to be connected.

**Figure 136:**  
**2-Side Bonding**



## Package Drawings & Markings

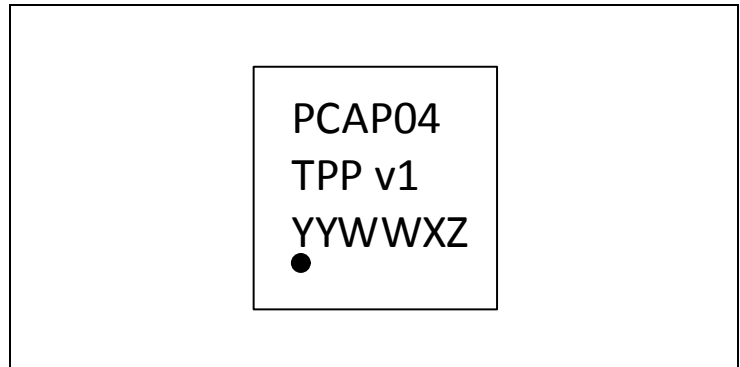
Figure 137:  
Package Drawing (QFN24)



**Note(s):**

1. Dimensioning and tolerancing conform to ASME Y14.5M-1994
2. All dimensions are in millimeters (angles are in degrees).
3. Dimension b applies to metallized terminal and is measured between 0.25mm and 0.30mm from terminal tip. Dimension L1 represents terminal full back from package edge up to 0.15mm is acceptable.
4. Coplanarity applies to the exposed heat slug as well as the terminal.
5. Radius on terminal is optional.
6. N id the total number of terminals.

**Figure 138:**  
Package Marking



**Figure 139:**  
Package Code

| PCAP04 | T  | PP                    | v1                  | YY   | WW   | X                               | Z                                |
|--------|--|-----------------------|---------------------|------|------|---------------------------------|----------------------------------|
| Part   | Temperature<br>A : -40°C to 125°C<br>B : -40°C to 85°C | Package<br>QF : QFN24 | Silicon<br>revision | Year | Week | Assembly<br>Plant<br>Identifier | Assembly<br>Traceability<br>Code |



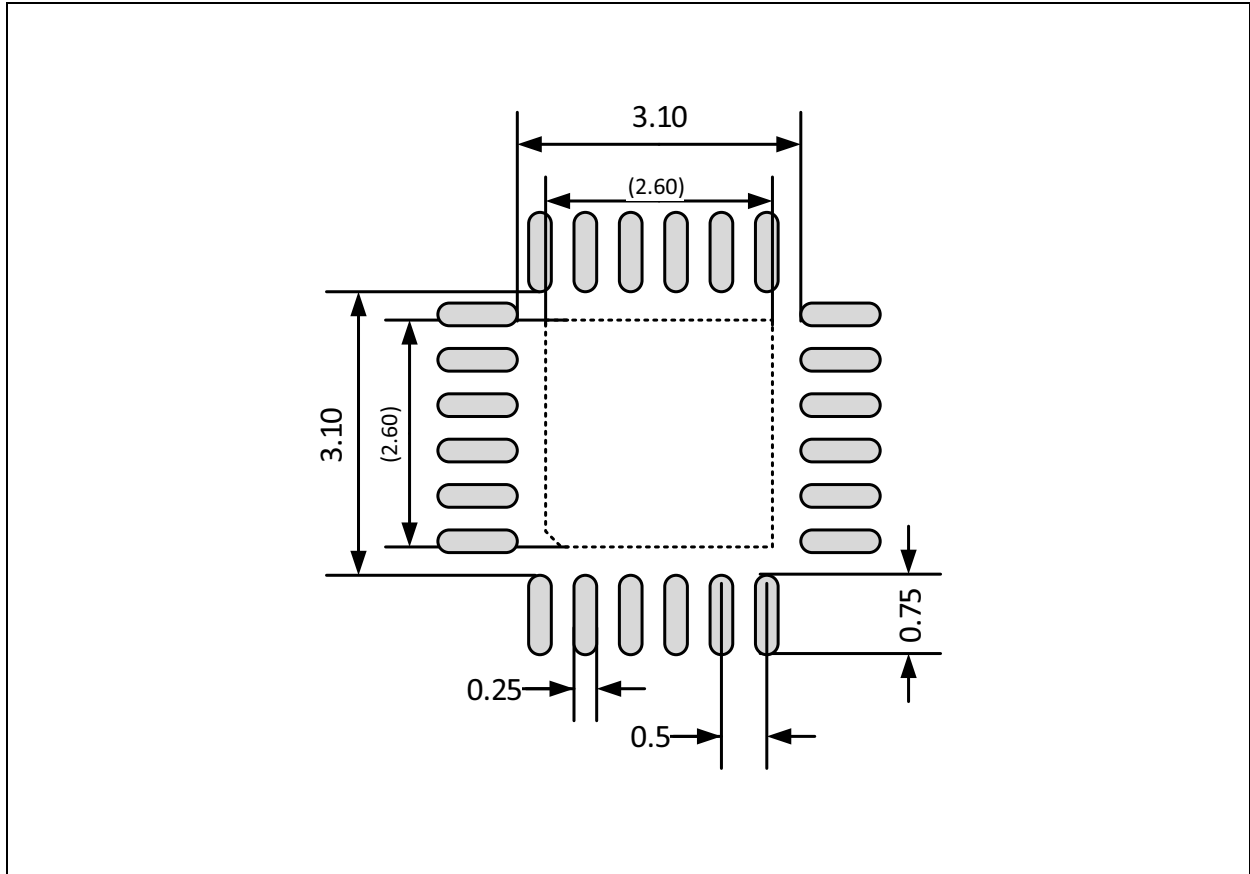
## PCB Pad Layout

**Caution:** The center pad is internally connected to GND. No wires other than GND are allowed underneath.

It is recommended to not solder the center pad. Too much solder paste could reduce solder quality.

Suitable socket: e.g. Plastronics 32QN50S15050D

**Figure 140:**  
Landing Pattern



**Note(s):**

1. All dimensions are in mm.

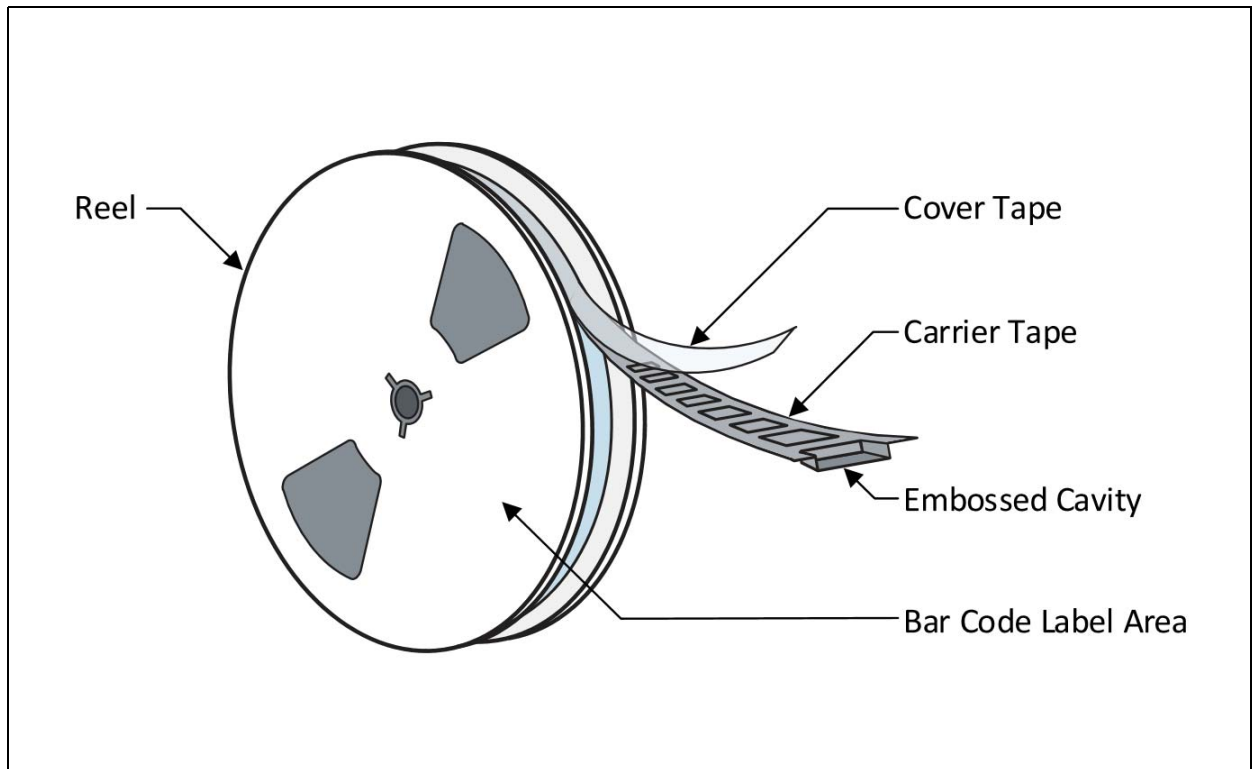
### Tape & Reel Information

Figure 141:  
Package Overview (only QFN)

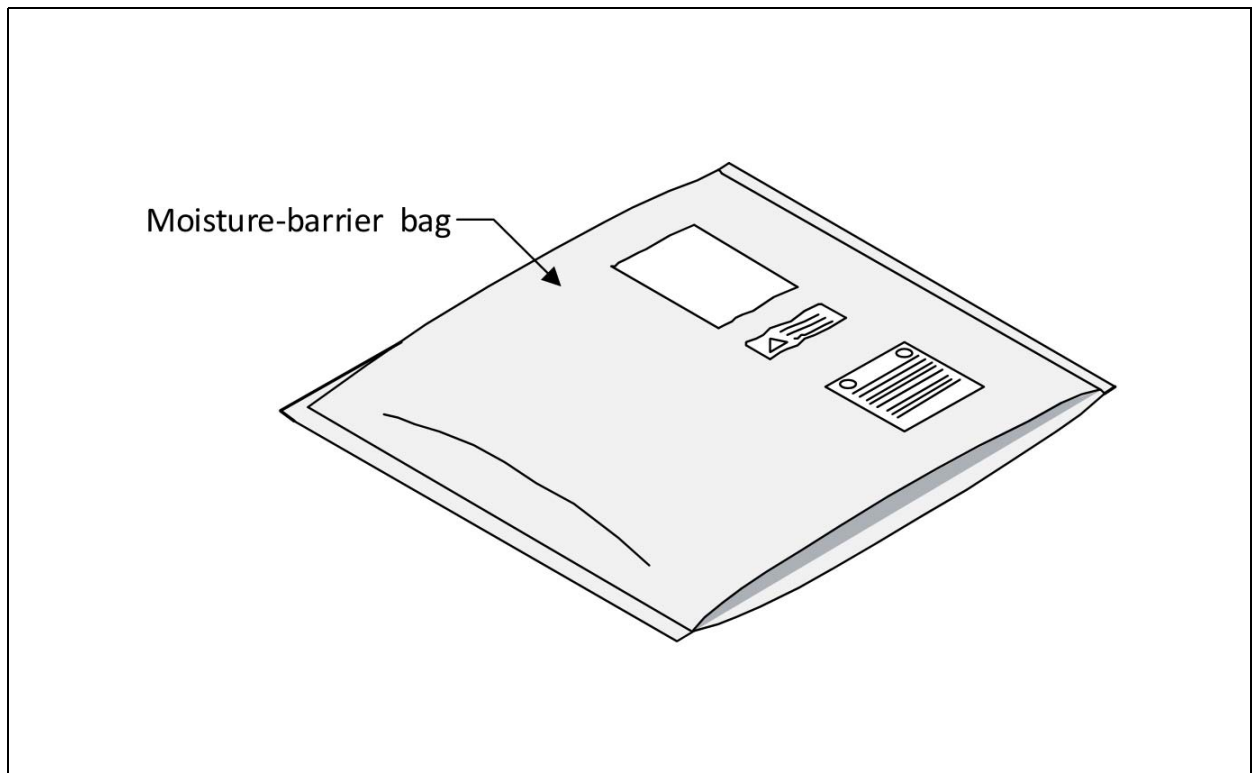
| Package | Device | Devices per Reel |          |
|---------|--------|------------------|----------|
|         |        | 7" Reel          | 13" Reel |
| QFN24   | PCap04 | 1000             | 6000     |

The tape-and-reel configuration is used for transport and storage from the manufacturer (ams AG) to the customer, and for use in the customer manufacturing plant. The configuration is designed for feeding components to automatic-placement machines for surface mounting on board assemblies. The complete configuration consists of a carrier tape with sequential individual cavities that hold individual components, and a cover tape that seals the carrier tape to retain the components in the cavities. Single reels are packed into dry-pack and inserted into intermediate boxes before shipping.

Figure 142:  
Reel

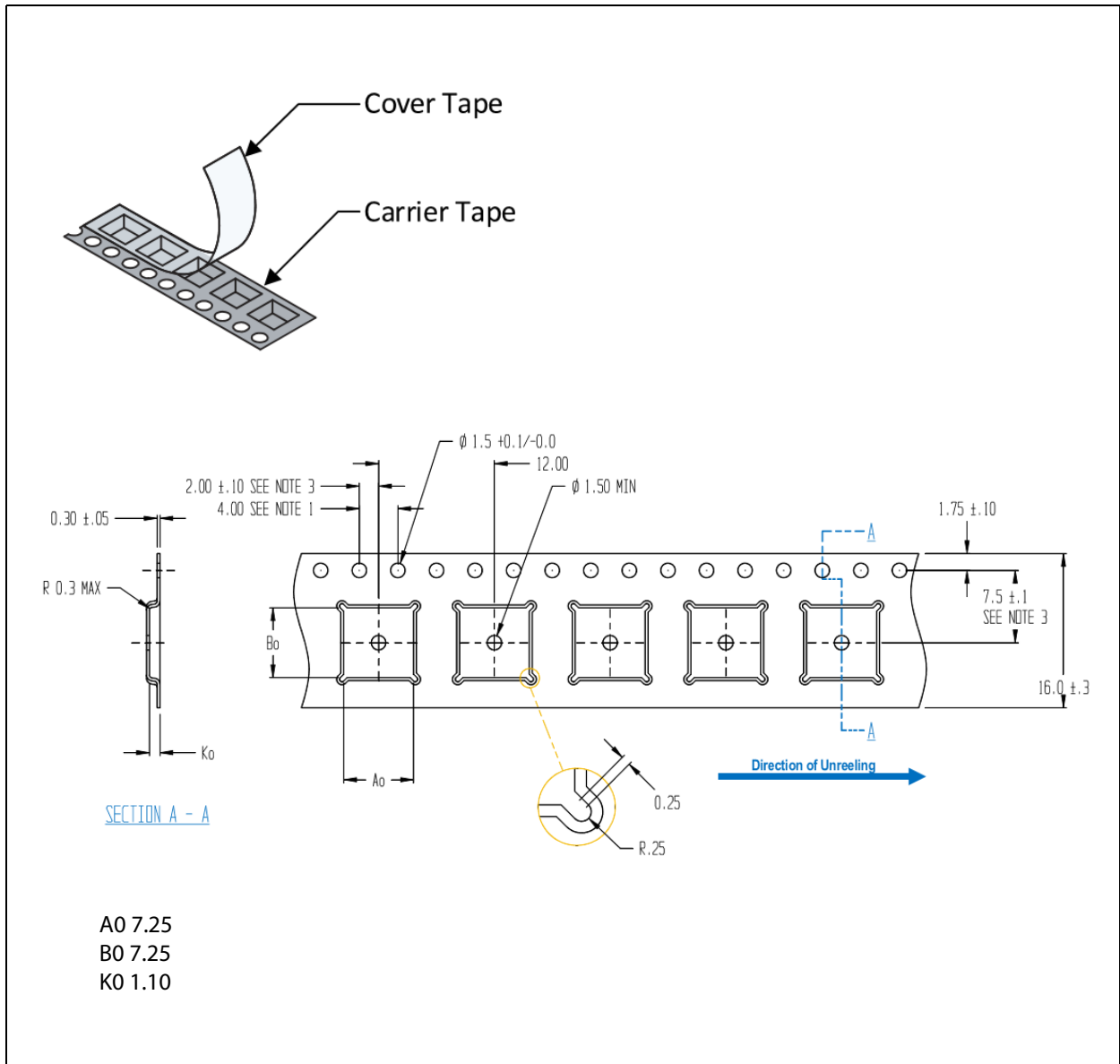


**Figure 143:**  
**Single Reel in Dry Bag**



Carrier tape is widely used for presenting devices to pick-and-place machines for automatic placement onto printed circuit boards.

Figure 144:  
Carrier Tape QFN24 (32)



**Note(s):**

1. Sprocket hole pitch cumulative tolerance  $\pm 0.2$
2. Camber in Compliance with EIA 481
3. Pocket position relative to sprocket hole measured as true position of pocket, not pocket hole
4. All dimensions in mm

## **Soldering & Storage Information**

### IPC/JEDEC J-STD-020

The reflow peak soldering temperature (body temperature) is specified according to IPC/JEDEC J-STD-020 "Moisture/Reflow Sensitivity Classification for Non-hermetic Solid State Surface Mount Devices." The lead finish for Pb-free leaded packages is "Matte Tin" (100% Sn)

## Ordering & Contact Information

Figure 145:  
Ordering Information

| Ordering Code  | Package | Marking                     | Delivery Form                  | Delivery Quantity    |
|----------------|---------|-----------------------------|--------------------------------|----------------------|
| PCap04-AQFT-24 | QFN24   | PCAP04<br>AQF V1<br>YYWWXZZ | 13" Tape & Reel<br>in dry pack | 6000 pcs/reel        |
| PCap04-AQFM-24 |         |                             | 7" Tape & Reel<br>in dry pack  | 1000 pcs/reel        |
| PCap04-BQFT-24 | QFN24   | PCAP04<br>BQF V1<br>YYWWXZZ | 13" Tape & Reel<br>in dry pack | 6000 pcs/reel        |
| PCap04-BQFM-24 |         |                             | 7" Tape & Reel<br>in dry pack  | 1000 pcs/reel        |
| PCap04-ASWB    | n.a.    | n.a.                        | Sorted wafer in box            | 1 wafer = ~10000 die |
| PCap04-ASDF    | n.a.    | n.a.                        | Sorted wafer on blue foil      | 1 wafer = ~10000 die |
| PCap04-BSWB    | n.a.    | n.a.                        | Sorted wafer in box            | 1 wafer = ~10000 die |

Buy our products or get free samples online at:

[www.ams.com/ICdirect](http://www.ams.com/ICdirect)

Technical Support is available at:

[www.ams.com/Technical-Support](http://www.ams.com/Technical-Support)

Provide feedback about this document at:

[www.ams.com/Document-Feedback](http://www.ams.com/Document-Feedback)

For further information and requests, e-mail us at:

[ams\\_sales@ams.com](mailto:ams_sales@ams.com)

For sales offices, distributors and representatives, please visit:

[www.ams.com/contact](http://www.ams.com/contact)

### Headquarters

ams AG  
Tobelbader Strasse 30  
8141 Premstaetten  
Austria, Europe

Tel: +43 (0) 3136 500 0

Website: [www.ams.com](http://www.ams.com)

## RoHS Compliant & ams Green Statement

**RoHS:** The term RoHS compliant means that ams AG products fully comply with current RoHS directives. Our semiconductor products do not contain any chemicals for all 6 substance categories, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes.

**ams Green (RoHS compliant and no Sb/Br):** ams Green defines that in addition to RoHS compliance, our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

**Important Information:** The information provided in this statement represents ams AG knowledge and belief as of the date that it is provided. ams AG bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ams AG has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ams AG and ams AG suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

## Copyrights & Disclaimer

Copyright ams AG, Tobelbader Strasse 30, 8141 Premstaetten, Austria-Europe. Trademarks Registered. All rights reserved. The material herein may not be reproduced, adapted, merged, translated, stored, or used without the prior written consent of the copyright owner.

Devices sold by ams AG are covered by the warranty and patent indemnification provisions appearing in its General Terms of Trade. ams AG makes no warranty, express, statutory, implied, or by description regarding the information set forth herein. ams AG reserves the right to change specifications and prices at any time and without notice. Therefore, prior to designing this product into a system, it is necessary to check with ams AG for current information. This product is intended for use in commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or life-sustaining equipment are specifically not recommended without additional processing by ams AG for each application. This product is provided by ams AG "AS IS" and any express or implied warranties, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose are disclaimed.

ams AG shall not be liable to recipient or any third party for any damages, including but not limited to personal injury, property damage, loss of profits, loss of use, interruption of business or indirect, special, incidental or consequential damages, of any kind, in connection with or arising out of the furnishing, performance or use of the technical data herein. No obligation or liability to recipient or any third party shall arise or flow out of ams AG rendering of technical or other services.



## Document Status

| Document Status          | Product Status  | Definition   |
|--------------------------|-----------------|--|
| Product Preview          | Pre-Development | Information in this datasheet is based on product ideas in the planning phase of development. All specifications are design goals without any warranty and are subject to change without notice  |
| Preliminary Datasheet    | Pre-Production  | Information in this datasheet is based on products in the design, validation or qualification phase of development. The performance and parameters shown in this document are preliminary without any warranty and are subject to change without notice            |
| Datasheet                | Production      | Information in this datasheet is based on products in ramp-up to full production or full production which conform to specifications in accordance with the terms of ams AG standard warranty as given in the General Terms of Trade                                |
| Datasheet (discontinued) | Discontinued    | Information in this datasheet is based on products which conform to specifications in accordance with the terms of ams AG standard warranty as given in the General Terms of Trade, but these products have been superseded and should not be used for new designs |

**Revision Information**

| Changes from 1-02 (2017-Nov-08) to current revision 1-03 (2018-Apr-04) | Page |
|--|------|
| Updated Figure 1   | 1    |
| Updated Figure 8   | 10   |
| Updated notes under Figure 10  | 13   |
| Updated text under Figure 77   | 53   |
| Updated Figure 78  | 54   |
| Updated text under Figure 80   | 56   |
| Updated text under Figure 89   | 62   |
| Updated text under Trigger   | 66   |
| Updated text under NVRAM Access  | 94   |

**Note(s):**

1. Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
2. Correction of typographical errors is not explicitly mentioned.

**Content Guide**

|           |   |
|-----------|---|
| <b>1</b>  | <b>General Description</b>                        |
| 1         | Key Benefits & Features                           |
| 2         | Applications MEMS Sensors                         |
| 2         | Block Diagram                                     |
| <b>3</b>  | <b>Pin Assignments</b>                            |
| 4         | Pin Description                                   |
| 6         | Pad Coordinates                                   |
| <b>8</b>  | <b>Absolute Maximum Ratings</b>                   |
| <b>10</b> | <b>Electrical Characteristics</b>                 |
| 13        | CDC Characteristics                               |
| 14        | RDC Characteristics                               |
| <b>16</b> | <b>Timing Characteristics</b>                     |
| <b>17</b> | <b>Detailed Description</b>                       |
| <b>19</b> | <b>Register Description</b>                       |
| 19        | Configuration Registers                           |
| 19        | Register Overview                                 |
| 24        | Detailed Configuration Register Description       |
| 24        | Configuration Register 0 (Address 0x0)            |
| 24        | Configuration Register 1 (Address 0x1)            |
| 25        | Configuration Register 2 (Address 0x2)            |
| 25        | Configuration Register 3 (Address 0x3)            |
| 26        | Configuration Register 4 (Address 0x4)            |
| 26        | Configuration Register 5 (Address 0x5)            |
| 27        | Configuration Register 6 (Address 0x6)            |
| 27        | Configuration Register 7:8 (Address 0x7, 0x8)     |
| 27        | Configuration Register 11:9 (Address 0x9;0xA;0xB) |
| 28        | Configuration Register 12 (Address 0xC)           |
| 28        | Configuration Register 13 (Address 0xD)           |
| 28        | Configuration Register 14 (Address 0xE)           |
| 29        | Configuration Register 15 (Address 0xF)           |
| 29        | Configuration Register 16 (Address 0x10)          |
| 29        | Configuration Register 17 (Address 0x11)          |
| 30        | Configuration Register 18 (Address 0x12)          |
| 31        | Configuration Register 19 (Address 0x13)          |
| 31        | Configuration Register 20 (Address 0x14)          |
| 32        | Configuration Register 21 (Address 0x15)          |
| 32        | Configuration Register 22 (Address 0x16)          |
| 33        | Configuration Register 23 (Address 0x17)          |
| 33        | Configuration Register 24 (Address 0x18)          |
| 34        | Configuration Register 25 (Address 0x19)          |
| 34        | Configuration Register 26 (Address 0x1A)          |
| 34        | Configuration Register 27 (Address 0x1B)          |
| 35        | Configuration Register 28 (Address 0x1C)          |
| 35        | Configuration Register 29 (Address 0x1D)          |
| 35        | Configuration Register 30 (Address 0x1E)          |
| 36        | Configuration Register 31 (Address 0x1F)          |
| 37        | Configuration Register 32 (Address 0x20)          |
| 37        | Configuration Register 33 (Address 0x21)          |
| 38        | Configuration Register 34 (Address 0x22)          |

|           |   |
|-----------|---|
| 38        | Configuration Register 35 (Address 0x23)                                |
| 38        | Configuration Register 36 (Address 0x24)                                |
| 39        | Configuration Register 37 (Address 0x25)                                |
| 39        | Configuration Register 38 (Address 0x26)                                |
| 39        | Configuration Register 39 (Address 0x27)                                |
| 39        | Configuration Register 40 (Address 0x28)                                |
| 40        | Configuration Register 41 (Address 0x29)                                |
| 40        | Configuration Register 42 (Address 0x30)                                |
| 40        | Configuration Register 47 (Address 0x2F)                                |
| 41        | Configuration Register 48 (Address 0x30)                                |
| 41        | Configuration Register 49 (Address 0x31)                                |
| 41        | Configuration Register 50 (Address 0x32)                                |
| 42        | Configuration Register 51 to 53: ams internal Registers, 0x00 mandatory |
| 42        | Configuration Register 54 (Address 0x36)                                |
| 42        | Configuration Register 55 to 61: ams internal Registers, 0x00 mandatory |
| 42        | Configuration Register 62 (Address 0x3e)                                |
| 42        | Configuration Register 63 (Address 0x3f)                                |
| <b>43</b> | <b>Read Registers</b>   |
| 45        | Result Registers  |
| 45        | Status Registers  |
| <b>47</b> | <b>Principles of Operation</b>  |
| <b>47</b> | <b>Converter Frontend</b>   |
| 47        | Capacitance-to-Digital Converter (CDC)                                  |
| 47        | <i>Measuring Principle</i>  |
| 47        | <i>Connecting Sensors</i>   |
| 48        | <i>Discharge Resistors</i>  |
| 49        | <i>Cycle</i>  |
| 51        | <i>Sequence</i>   |
| 52        | <i>Conversion</i>   |
| 55        | CDC Compensation Options  |
| 55        | <i>Internal Compensation</i>  |
| 55        | <i>External Compensation</i>  |
| 56        | <i>DC Balance</i>   |
| 57        | <i>Gain Correction</i>  |
| 57        | CDC Important Parameters  |
| 57        | <i>Cycle Clock</i>  |
| 58        | <i>Cycle Time</i>   |
| 59        | <i>Sequence</i>   |
| 60        | <i>Conversion</i>   |
| 61        | Guarding  |
| 64        | RDC Resistance-to-Digital Converter                                     |
| 64        | <i>Measuring Principle</i>  |
| 64        | <i>Connecting Sensors</i>   |
| 65        | <i>Cycle &amp; Conversion</i>   |
| 66        | <i>Trigger</i>  |
| 67        | RDC Important Parameters  |
| 67        | <i>Cycle Clock</i>  |
| 68        | <i>Sequence</i>   |
| 68        | <i>Conversion</i>   |
| 68        | <i>RDC Results, Ratios</i>  |
| <b>69</b> | <b>Interfaces (Serial &amp; PDM/PWM)</b>                                |

|           |  |
|-----------|--|
| 69        | Serial Interfaces (SIF)  |
| 70        | <i>Opcodes</i>   |
| 71        | I <sup>2</sup> C Compatible Interface  |
| 72        | <i>I<sup>2</sup>C Timing</i>   |
| 72        | <i>I<sup>2</sup>C Write</i>  |
| 73        | <i>I<sup>2</sup>C Read</i>   |
| 73        | SPI Interface  |
| 74        | <i>SPI Timing</i>  |
| 76        | GPIO and PDM/PWM   |
| 78        | <i>Debouncing Filter</i>   |
| 78        | <i>PDM and PWM</i>   |
| <b>82</b> | <b>Oscillators</b>   |
| <b>83</b> | <b>DSP &amp; Memory</b>  |
| 84        | DSP & Environment  |
| 85        | RAM Structure  |
| 87        | <i>Registers 0 to 95, User RAM</i>   |
| 88        | <i>Register 96, Flags &amp; Internal Control Signals</i>                             |
| 88        | <i>DSP Read Register 97</i>  |
| 89        | <i>DSP Read Registers 98 to 103</i>  |
| 89        | <i>DSP Read/Write Registers 105 to 108, Data Pointer</i>                             |
| 90        | <i>DSP Read Register 126, PORTINFO (PORTERR&lt;7...0&gt;, PORTMASK&lt;7...0&gt;)</i> |
| 90        | <i>DSP Write Registers 97 to 104, RES00...RES07</i>                                  |
| 90        | <i>DSP Write Registers 109, 110, PIO_REF...PI1_REF</i>                               |
| 90        | <i>DSP Write Register 111, TIMER0</i>  |
| 91        | <i>DSP Read/Write Registers 112 to 120, MEM_CTRL, MEM_ADD, MEM_DATA</i>              |
| 92        | <i>DSP Read Registers 121 to 124, TDC_START, TDC_STOP, C_ADD_PTR, R_ADD_PTR</i>      |
| 93        | NVRAM and ROM  |
| 93        | <i>NVRAM Structure</i>   |
| 94        | <i>NVRAM Access</i>  |
| 94        | <i>ROM Structure</i>   |
| 94        | <i>DSP Inputs &amp; Outputs</i>  |
| 99        | <i>ALU Flags</i>   |
| 100       | <i>DSPOUT – GPIO Assignment</i>  |
| 104       | DSP Configuration  |
| 105       | <i>DSP Start</i>   |
| 105       | <i>Watchdog</i>  |
| 106       | Instruction Set  |
| 107       | <i>Instructions</i>  |
| 125       | Instruction Details  |
| 125       | <i>Pointer</i>   |
| 126       | <i>Transfer Constants with Push and Pop</i>  |
| 126       | <i>mult</i>  |
| 127       | <i>Handover of Constants by Push &amp; Pop</i>                                       |
| 127       | <i>div</i>   |
| 130       | ROM Routines   |
| 132       | <i>_ROM_Version__</i>  |
| 132       | <i>_ROM_tdc_dispatch__</i>   |
| 133       | <i>_ROM_cdc_cycle__</i>  |
| 134       | <i>_ROM_Rdc_cycle__</i>  |
| 135       | <i>_ROM_cdc_initialize__</i>   |
| 135       | <i>_ROM_rdc_initialize__</i>   |

|     |   |
|-----|---|
| 136 | <i>shiftL_A_xx; shiftR_A_xx; shiftL_B_xx; shiftR_B_xx</i> |
| 136 | <i>mult_xx</i>  |
| 137 | <i>div_xx</i>   |
| 138 | <i>_ROM_div_variable__</i>                                |
| 139 | <i>_ROM_mult_variable__</i>                               |
| 140 | <i>_ROM_shift_a_variable__</i>                            |
| 140 | <i>_ROM_shift_b_variable__</i>                            |
| 141 | <i>_ROM_dma__</i>   |
| 142 | <i>_ROM_ln__, _ROM_log10__, _ROM_ld__</i>                 |
| 142 | <i>_ROM_signed24_to_signed32__</i>                        |
| 143 | <i>_ROM_median__</i>                                      |
| 144 | <i>_ROM_cdc__</i>   |
| 145 | <i>_ROM_rdc__ ; _ROM_rdc_inverse__</i>                    |
| 146 | <i>_ROM_memory_ (read/write volatile memory)</i>          |
| 148 | <i>_ROM_MEMORY_RECALL/STORE</i>                           |
| 149 | <i>_ROM_2pt_calibration</i>                               |
| 150 | <i>_ROM_polynomial_3rd_degree</i>                         |
| 151 | <i>_ROM_polynomial_4th_degree</i>                         |
| 152 | <i>_ROM_pulse__</i>                                       |
| 153 | <i>_ROM_pulse_loaded_cal_vals</i>                         |
| 154 | <i>_ROM_NVblock_copy__</i>                                |
| 155 | <i>_ROM_capacitance_polynomial</i>                        |
| 156 | <i>_ROM_capacitance_polynomial_4d</i>                     |
| 157 | Assembly Programs   |
| 158 | Directives  |
| 160 | Sample Code   |
| 160 | “for” Loop  |
| 160 | “while” Loop  |
| 160 | “do - while” Loop   |
| 161 | “do - while” with 2 pointers                              |
| 161 | Rotate Right A to B                                       |
| 162 | Libraries   |
| 163 | pcap_standard.h   |
| 163 | PCap04_ROM_addresses_standard.h                           |
| 164 | pcap_config.h   |

## **165 Application Information**

|     |                   |
|-----|-------------------|
| 165 | Schematic         |
| 166 | Minimized Bonding |

## **167 Package Drawings & Markings**

### **169 PCB Pad Layout**

### **170 Tape & Reel Information**

### **173 Soldering & Storage Information**

### **174 Ordering & Contact Information**

### **175 RoHS Compliant & ams Green Statement**

### **176 Copyrights & Disclaimer**

### **177 Document Status**

### **178 Revision Information**

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

## ScioSense:

[PCAP04-BQFT-24](#) [PCAP04-ASWB](#) [PCAP04-BQFM-24](#) [PCAP04-ASDF](#) [PCAP04-AQFT-24](#) [PCAP04-ASDW-100](#)  
[PCAP04-BSWB-290](#) [PCAP04-BSWB-735](#) [PCAP04-BQFM-24-V2](#)