

PS09-DLC-EVA

DLC PCB based on PS09

For Digital Load Cell (DLC) development

7th June 2011

Document-No.: DB_PS09-DLC-EVA_en V0.1



Published by acam-messelectronic gmbh

© acam-messelectronic gmbh 2011

Disclaimer / Notes

„Preliminary“ product information describes a product which is not in full production so that full information about the product is not available yet. Therefore, acam messelectronic GmbH („acam“) reserves the right to modify this product without notice. The information provided by this data sheet is believed to be accurate and reliable. However, no responsibility is assumed by acam for its use, nor for any infringements of patents or other rights of third parties that may result from its use. The information is subject to change without notice and is provided „as is“ without warranty of any kind (expressed or implied). Picostrain is a registered trademark of acam. All other brand and product names in this document are trademarks or service marks of their respective owners.

Support

For a complete listing of Direct Sales, Distributor and Sales Representative contacts, visit the acam web site at: <http://www.acam.de/company/distributors>.

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

Table of Contents

	Page
1 Introduction	1-1
2 Connecting Strain gauges	2-2
	2.1 Connecting a Half Bridge
	2.2 Connecting a Full Bridge Sensor
	2.3 Connecting a Wheatstone
	2.4 Test Connection without Sensor
	2.5 Capacitor Selection & Assembly
3 The I2C Interface	3-1
4 EEPROM : Program development	4-1
5 UART	5-1
	5.1 Transmission using the UART
	5.2 Reception using the UART
	5.3 Testing a UART Program
6 Miscellaneous	6-1
7 Appendix	7-2
	7.1 Sample Program 1
	7.2 Sample Program 2
	7.3 Schematic Diagrams
	7.4 Layout PS09-DLC-EVA

acam ®, **PICO**STRAIN ® are registered trademarks of acam-messelectronic gmbh

1 Introduction

The PS09 DLC evaluation board is a hardware that can be connected to a load cell to realize a digital load cell (DLC) system, with the electronics on board being quasi integrated with the load cell itself. This DLC system can be used to evaluate the various capabilities of the newest chip in the PICOSTRAIN series namely PS09. Please refer to the detailed data sheet of the PS09 chip to know more about the salient features of the chip itself.

This document is a starter's guide to using the PS09 DLC evaluation board. It outlines the important features of the hardware that would help to use this board in coordination with a load cell. This board is thought of as a universally usable hardware for DLCs. It is possible to evaluate most of the features of the PS09 chip using this board.

The PS09 DLC EVA board is a 4 layer¹ PCB with components mounted on both sides. The board has a 4 MHz oscillator and the external comparator option. The PS09 chip supports three digital interfaces - SPI, I2C and UART. The chip can be programmed using SPI and I2C only. However, only I2C programming interface to the PS09 chip is supported on the DLC EVA board. SPI interface is not supported by this board. After programming, the UART interface is also supported. The DLC EVA board connects to the PICOPROG V2.0 programmer available from acam. The board is powered with a 5 V power supply voltage from the PICOPROG programmer; this is regulated on the EVA board to 3.3 V and supplied to the PS09 chip.

Figure 1-1 Adapter board with jumpers



1

The DLC EVA board is a complete evaluation system that allows evaluation and programming, with connection of various types of sensors. For realizing the PCB on a compact size, 4 layers were necessary. But in later DLC applications, 2 layer boards ought to be sufficient for most cases.

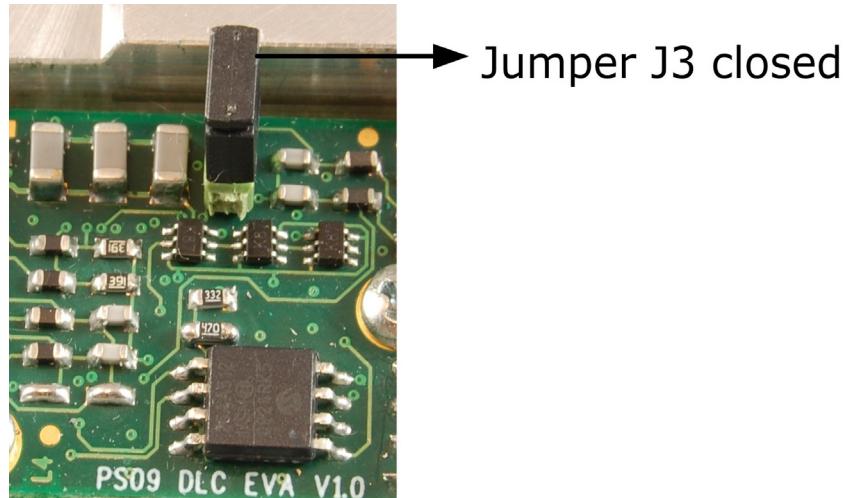
The communication from the PS09 evaluation software installed on the PC to the DLC EVA board can be established through the PICOPROG V 2.0 programmer. The programmer connects from the USB port of the PC to an adapter board. The adapter board is used to connect the programmer to the PS09 DLC EVA board on one side; on the other side it connects the PS09 DLC board to a UART interface. Depending on whether the DLC board must be supplied with 5 V from the PICOPROG programmer or from the UART connector, the jumper on the adapter board has to be set to either '*Picoprog*' or '*UART*'. Figure 1-1 is a picture of the adapter board showing the labeled jumper and the connectors.

When the PS09 chip has to be configured through the evaluation software or if the on-board external EEPROM has to be programmed from the assembler software, the jumper is set to '*Picoprog*'. Then the DLC EVA board is directly connected to the Picoprog and the PS09 is accessible by the software through the I2C Interface. When a program has been programmed in the OTP or external EEPROM of the PS09 the PICOPROG may be disconnected. If the program in OTP/EEPROM performs communication through the UART of PS09, then the jumper has to be changed to '*UART*', so that the DLC Board is supplied with the 5 V voltage from the UART connector and the UART signals can be connected to an interface accordingly.

1.1 Front end & Stand-alone modes

The PS09 can be operated in front end mode through the I2C interface, i.e. through the PICOPROG programmer. In this case, the PS09 DLC board is accessible through the assembler and evaluation

Figure 1-2 Front end mode - Jumper J3

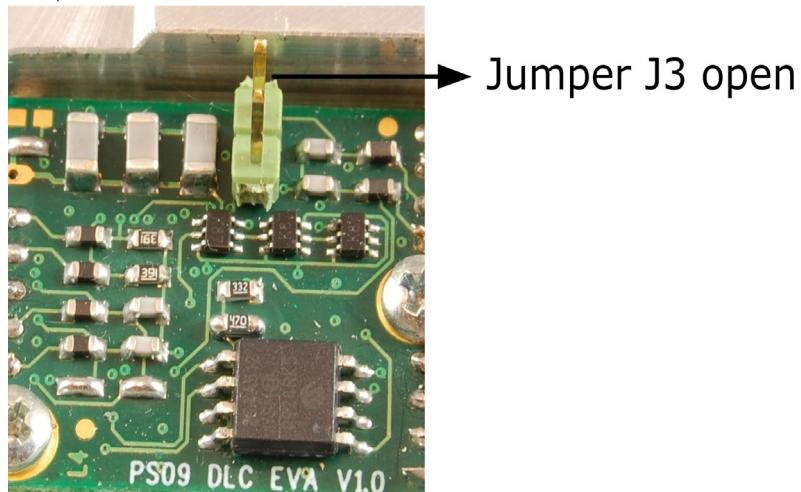


software for read / write and programming of the EEPROM or OTP . This is called front end mode where the communication can be performed with the chip through the I2C interface.

For this the **jumper J3** on the DLC board has to be **closed** (This jumper sets the MODE pin of PS09 to VCC). Figure1-2 shows the jumper setting for front end mode.

However once the PS09 is programmed appropriately such that a relevant measurement configuration is performed, then the chip can operate in standalone mode. In such a case, no I2C interface communication is supported anymore. The chip recognizes the stand-alone mode when the MODE pin is floating. This is achieved by **opening** the jumper **J3**. The following figure shows the jumper setting for stand-alone mode.

Figure 1-3 Stand-alone mode - Jumper J3



2 Connecting Strain gauges..... 2-2

2.1 Connecting a Half Bridge Sensor	2-2
2.2 Connecting a Full Bridge Sensor	2-3
2.3 Connecting a Wheatstone Full Bridge.....	2-4
2.4 Test Connection without Sensor	2-5
2.5 Capacitor Selection & Assembly	2-5

2 Connecting Strain gauges

It is possible to connect a strain gauge sensor to the PS09 DLC EVA board in different ways. The following chapter explains how to connect load cells with a single half bridge, a conventional full bridge or a Wheatstone full bridge to the board.

The PS09 DLC board allows to connect up to a maximum of two half bridges and perform measurements across the individual half bridges.

2.1 Connecting a Half Bridge Sensor

By virtue of the PICOSTRAIN measuring principle, it is sufficient to connect a half bridge with ONLY two resistors for a DLC system of high quality. There are two ways to connect a pure half bridge to the PS09 chip

- The conventional half bridge connection
- The alternative half bridge connection

These two connections are shown in the following figures.

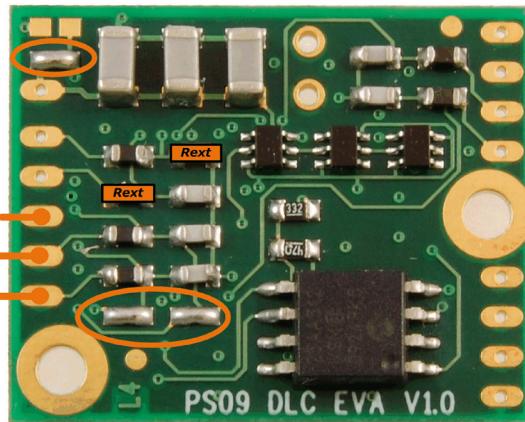
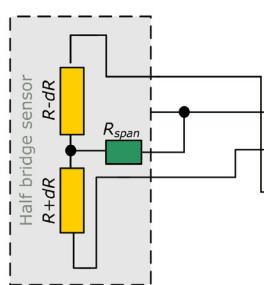
In the conventional half bridge connection external resistors (R_{ext}) are connected to the unused ports. The value of the external resistors must be equal to the value of the strain gage resistance in the half bridge i.e.

$$R_{ext} = R_{sg}$$

For e.g., in the Figure 2-1 : for 350 Ohm load cells, with $R_{span}=45$ Ohm, $R_{ext} = 350+45= 395$ Ohm.

Figure 2-1

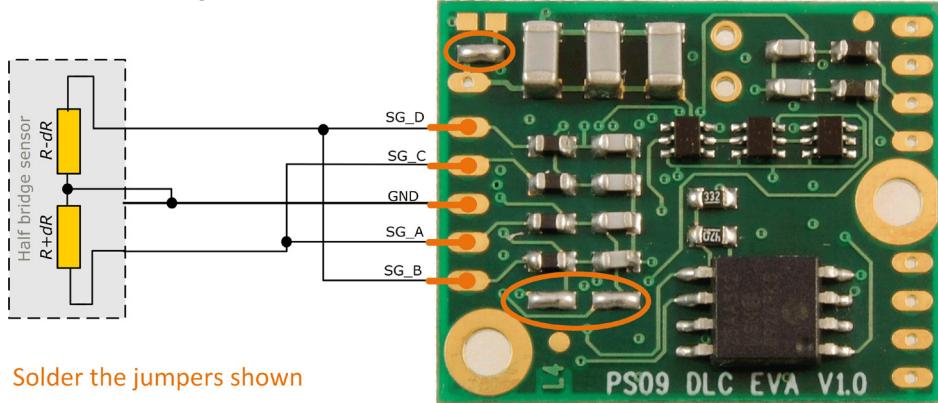
Classical Half Bridge



Solder R_{ext} across the solder pads shown in the picture
Solder the jumpers shown

Figure 2-2

Alternate Half Bridge



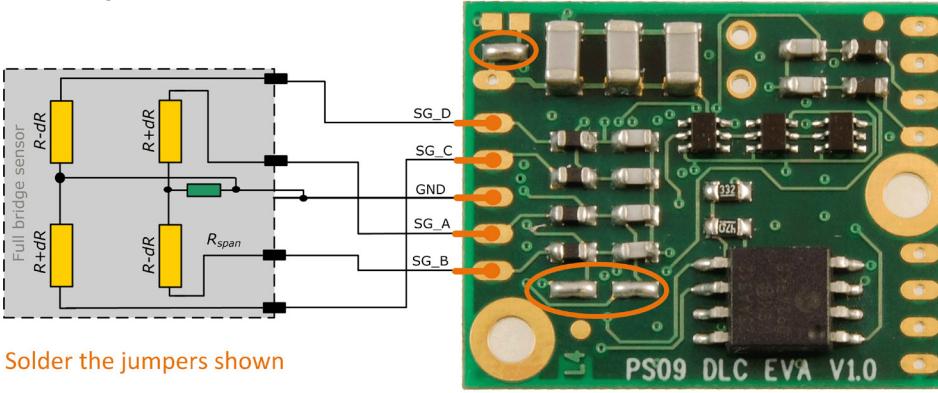
In the alternative half bridge connection the unused ports are connected parallel to the other two ports as shown in the figure above.

2.2 Connecting a Full Bridge Sensor

For PICOSTRAIN a full bridge is ideally separated into two half bridges. This wiring can increase the resolution compared to Wheatstone bridges by 0.6 bit. The connection of the Picostrain full bridge is shown in the following figure.

Figure 2-3

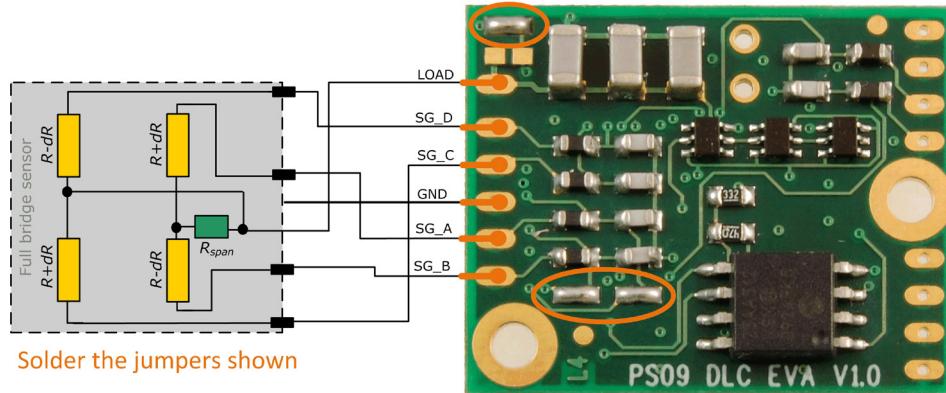
Full Bridge



For certain applications that demand higher resolution, PSRR etc. it would be necessary to use the PS081 compatible mode of PS09. For details on this mode and its associated advantages, please

refer to Section 3.3.5 of PS09 Data Sheet. The connection of the PS081 compatible Mode is as follows:

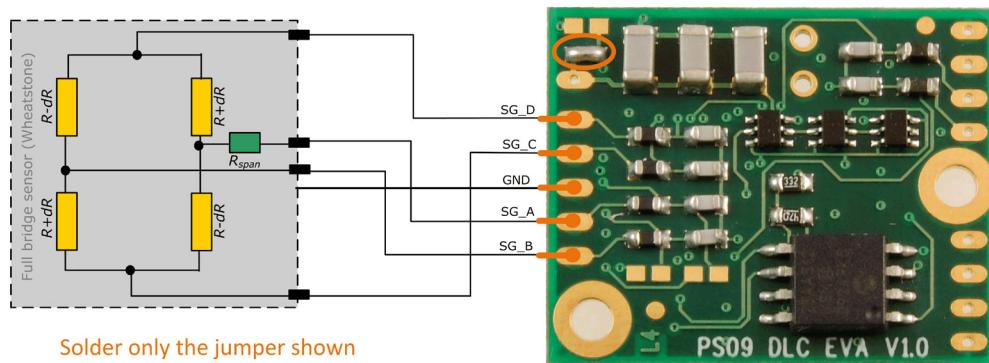
Figure 2-4 PS081 Compatible Mode - Full bridge



2.3 Connecting a Wheatstone Full Bridge

Existing load cells in Wheatstone configuration can be connected to the module without any modification. In general this will end in 0.6 bit less resolution compared to PICOSTRAIN wiring. It might be reasonable to use Wheatstone bridges in case of cables to the sensor longer than 0.5 m. The following figure shows the connection of the Wheatstone bridge.

Figure 2-5 Wheatstone bridge



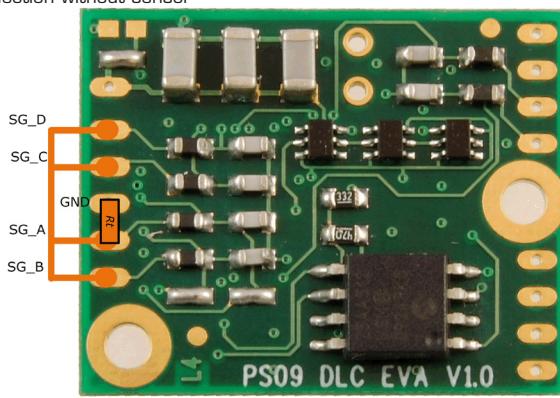
Note: In Wheatstone configuration, only Single Conversion Measurement mode is recommended.

The noise observed when reading out the measurement results in Wheastone connection, configured in continuous mode, using I2C (as on the DLC board) is high. Hence continuous mode is not recommended in this configuration.

2.4 Test Connection without Sensor

For test purposes, instead of connecting a load cell, it is possible to simply connect a SMD Resistor (for e.g. 1kOhm). This may be helpful in early connection tests or for program development, where actual sensors are not required initially. For e.g. during program development accurate results of measurement are not necessary, but a measurement just needs to be performed. For this purpose, just connect a simple SMD resistor between any one strain gage port and ground, and connect the other three ports to that port. This is illustrated in the figure below

Figure 2-6 Test connection without sensor



Connect an SMD Resistor $R_t = 1 \text{ k}\Omega$

2.5 Capacitor Selection & Assembly

The PICOSTRAIN measurement principle is based on measuring the discharging time of a capacitor. For this reason the correct size and material of the capacitors is significant to achieve best measuring results. In general we recommend a discharging time in the range of

$$T \sim 0.7 \cdot R \cdot C = 80 \text{ to } 120 \mu\text{s}$$

This is valid for half bridge and full bridge connections.

We recommend COG or CFCAP (Multilayer ceramic from Taiyo-Yuden) material capacitors. X7R capacitors can be used, too, but will show some minor loss in temperature stability.

The recommended values are:

$R_{sg} = 350 \text{ Ohm}$: $C_{load} = 300 \text{ nF to } 400 \text{ nF}$

$R_{sg} = 1000 \text{ Ohm}$: $C_{load} = 100 \text{ nF to } 150 \text{ nF}$

For Wheatstone connection, the discharging time is further reduced by a factor of 0.7; the following formula can be used to calculate the discharging capacitance.

$$T \sim 0.7 * 0.75 * R * C = 60 \text{ to } 110 \mu\text{s}$$

The recommend values for Wheatstone mode are:

R_{sg} = 350 Ohm : Cload = 300 nF to 400 nF

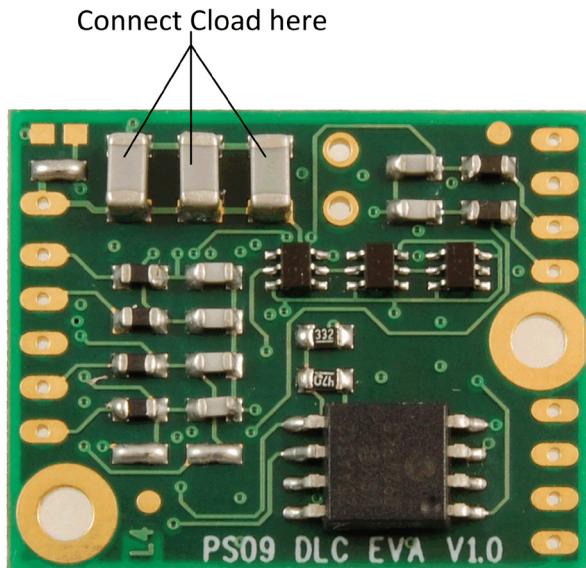
R_{sg} = 1000 Ohm : Cload = 100 nF to 200 nF

The PS09 DLC EVA board is pre-assembled with Cload = 3 x 100 nF = 300 nF.

The capacitor Cload must be chosen appropriately depending on the bridge configuration and soldered across the points shown in the figure below. The sum of the three shown capacitors must be equal to value of Cload.

Figure 2-7

Capacitor assembly



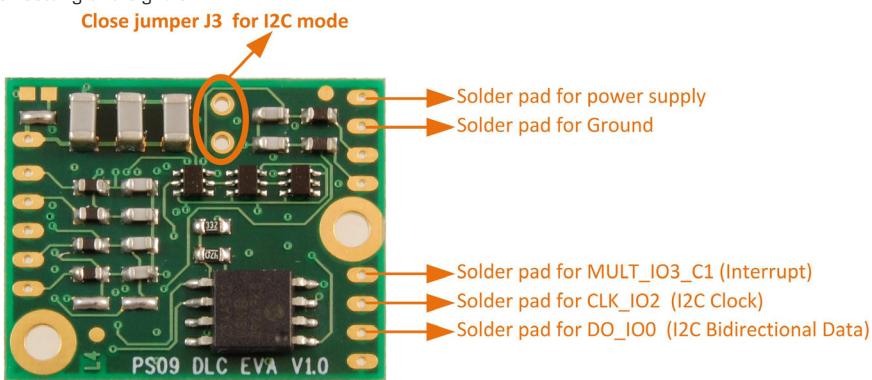
3 The I2C Interface

As already mentioned, the PS09 chip on the DLC EVA board is accessible through the I2C interface. The PS09 chip can be used in the pure front end converter mode. In this case, the chip can be addressed as an I2C slave by an external microcontroller. This mode is an option when the PS09 chip is used only as a converter to perform the measurement and the measurement values are read out and processed further by an external microcontroller.

Please refer to the **Section 4.5.1** of the PS09 data sheet for more details on the pure front end mode and other modes with the PS09. The section describes the various modes for SPI interface only (not supported on the DLC EVA board). The same conceptually however applies to the I2C interface also. The following are features that are specific to the I2C interface:

- The MODE pin of the PS09 chip ought to be set to VCC for I2C communication, i.e. the Jumper J3 on the DLC EVA board must be closed.
- Connect the I2C signals from the external microcontroller (I2C Clock and Bidirectional data line) to the respective pins of the PS09. The solder pads are shown in the Figure 3-1.
- Please refer to **Section 4.6** of the PS09 data sheet to know how to communicate with the PS09 through the I2C interface and the list of op-codes. The op codes are the same as the SPI interface, only an additional header (containing Start bit and Device address) and footer (Stop bit), have to be sent additionally in I2C.

Figure 3-1 I2C Mode - jumper setting and signals



When using the PS09 as a pure front end converter, the following is the program flow that has to be followed:

- 1. Set the MODE pin of the PS09 to VCC, by closing jumper J3 as shown in Figure 3-1.

- 2. Send a 'Power on Reset' (0xFO) op code to PS09 through the I2C interface. Wait for 10 us, then send a 'Watchdog off' (0x9E) op-code. Refer Section 4.6.2 of PS09 data sheet for I2C communication.
- 3. Configure the RAM of the PS09:

Configure the PS09 chip by writing the 16 configuration registers in the RAM Address 48-63 and UART registers in 80-91 (if needed), through the I2C interface.

RAM Write op-code (0x00) + address + 3 bytes (Config. data)

Please refer to Chapter 5 of the PS09 data sheet for a description of the Configuration register contents. Refer to Section 4.6.2.1 of PS09 data sheet to know how to perform a RAM write through the I2C interface.

NOTE:

- a) All otp_xxx_xxx bits to 0 (Configreg_1, bits 0-2).
 - b) Configure MULT_IO3_C1 pin as interrupt output from PS09 in order to generate interrupt after every measurement completion. Connect this signal from the DLC board to an input of the microcontroller and use it to poll for interrupt. See marked in the figure above, the solder pad of the MULT_IO3_C1 signal on the DLC board.
- 4. Optional:
Control read of RAM configuration using I2C interface for RAM Read (0x40) of RAM addresses 48-63. Refer to Section 4.6.2.1 of PS09 data sheet to know how to perform a RAM read through the I2C interface.
 - 5. Start measurement:
Send Init Reset opcode (0xCO) through I2C interface, followed by op code Start New Cycle (0xCC).
 - 6. Poll for Interrupt:
New Measurement value in PS09 is indicated by the Interrupt line (MULT_IO3_C1) going from 0 to 1. Read HBO result at RAM Address 244.

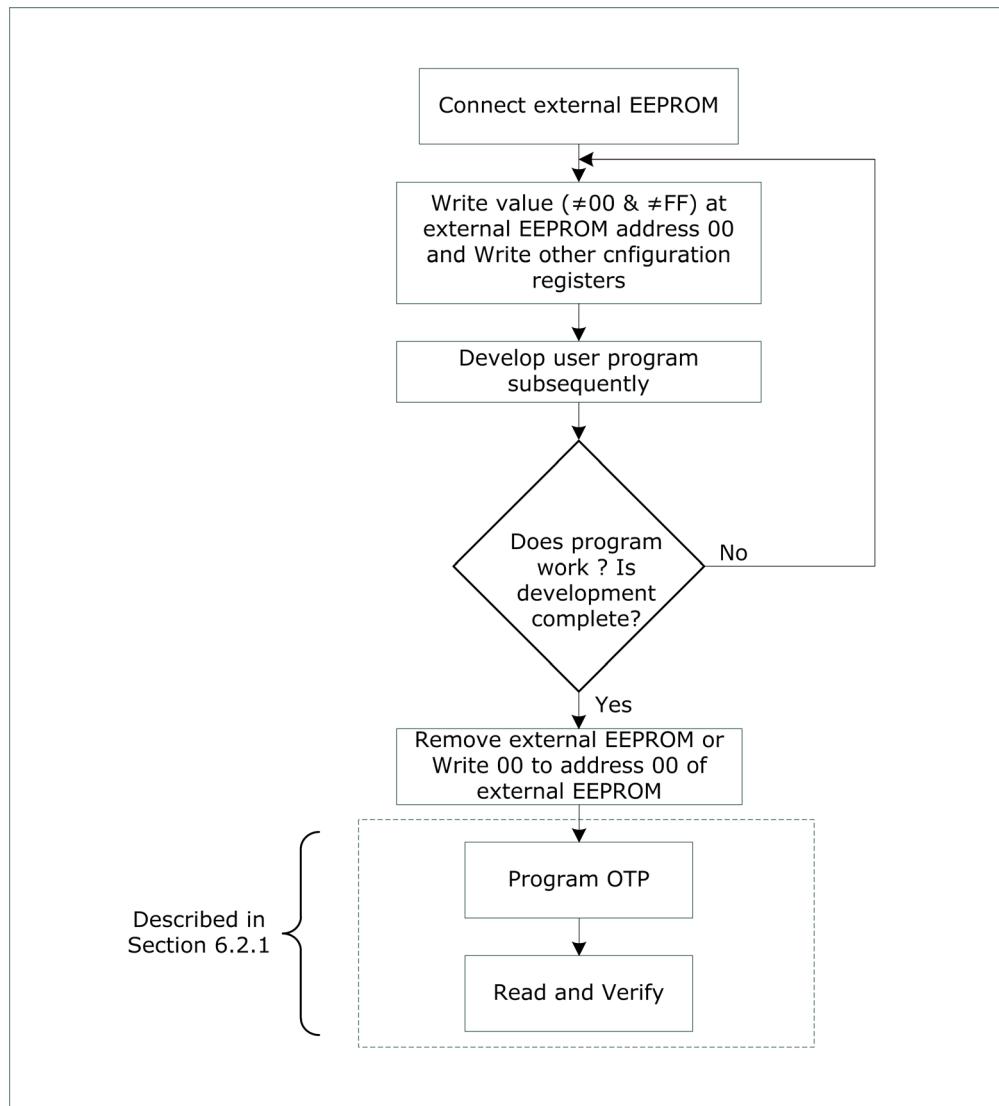
NOTE: In pure front end mode, as the first step, it has to be ensured that the external EEPROM is erased and the OTP is unprogrammed.

4 EEPROM : Program development

The PS09 is equipped with an on-chip one-time programmable memory, the OTP which is 8 kB in size. Since it is only one-time programmable, the chip supports an external EEPROM of maximum 8 kB and thus enables application program development.

Please refer to Section 6.2.3 of the PS09 data sheet for information on how to use the external EEPROM during program development. For details on how and when to program the OTP with the user code, please refer to Section 6.2.1 of the PS09 data sheet. The following flowchart shows the program development sequence using external EEPROM.

Figure 4-1 User program development using external EEPROM



Once the program is complete and final, the on-chip OTP can be programmed with the same program. Please note the execution time of a piece of code from the external EEPROM is 10-15 times longer than when the same code is executed from the OTP. This fact has to be kept in mind when programming delay routines using incr/decr opcodes.

The PS09 DLC EVA board is pre-assembled with a 64 kB external EEPROM from Microchip (25AA512). Please note that only 8 kB of this external EEPROM is addressable by the PS09.

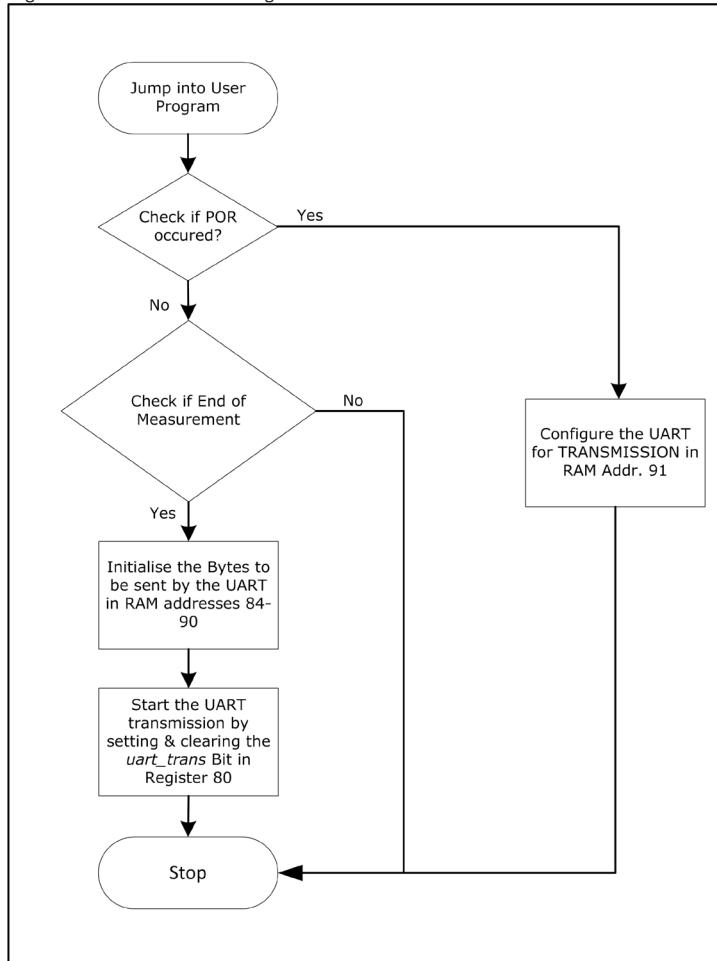
5 UART

The PS09 is equipped with a Universal Asynchronous Receiver Transmitter, UART that supports various configurable baud rates from 300 baud to 115200 baud. The UART has a 16 byte transmit buffer and can thus send a maximum of 16 bytes in one go. The UART can always receive byte-wise only. The UART can be configured to interrupt the DSP on every byte reception. For more information on the various features of the UART and how to configure those using registers in the PS09, please refer to Section 4.7 of the PS09 data sheet.

5.1 Transmission using the UART

The UART can be enabled by the auto configuration, but it can be configured only by the user program. Hence the UART has to be configured on power on reset by the user program. It is then ready for use. The following is a brief flowchart that shows how to use the UART in the PS09 chip for transmission.

Figure 5-1 Flowchart - Program for transmission using UART



Please refer to the appendix for a simple example with assembler program code that demonstrates how to send 16 arbitrary bytes from the UART at the end of every measurement. More sophisticated examples, for e.g. to send measurement values from the PS09 through the UART, are available as part of the PS09 Assembler under 'Search Examples'.

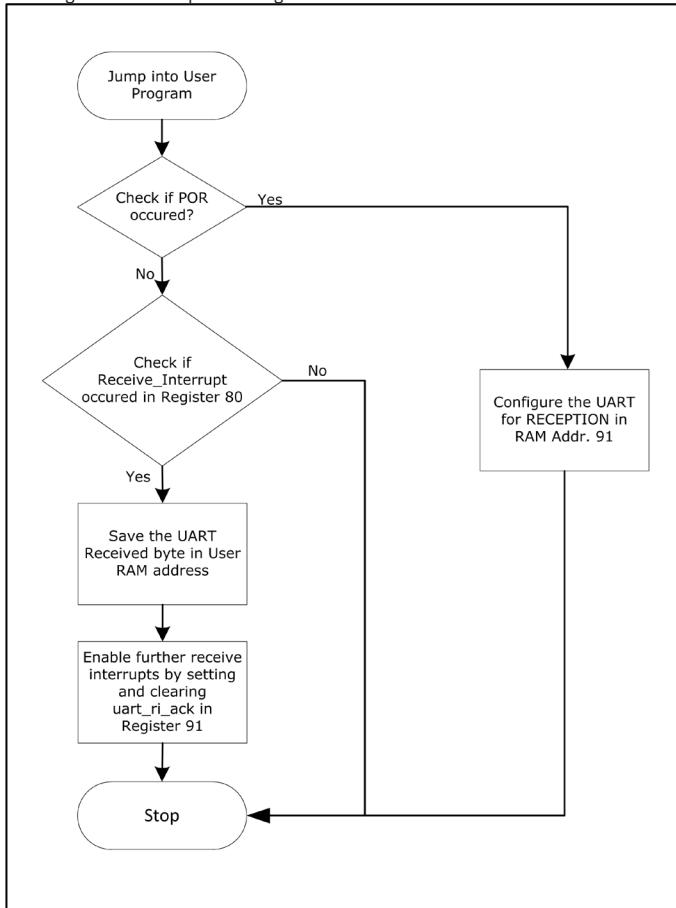
5.2 Reception using the UART

The PS09 UART can receive byte-wise only; the PS09 can be configured to interrupt the DSP on every byte reception. The time interval between successive incoming bytes ought to be long enough for the receive interrupt servicing program to store the previously received byte. Otherwise, the previously received byte is overwritten and data is lost. Hence care should be taken to read out the bytes in a timely manner so as not to lose any data.

The UART can be enabled by the auto configuration, but it can be configured only by the user program. Hence the UART has to be configured for reception on power on reset by the user program.

Figure 5-2

Flowchart - Program for reception using UART



It is then ready for reception. The Figure 4-2 is a brief flowchart that shows how to use the UART in the PS09 chip for reception.

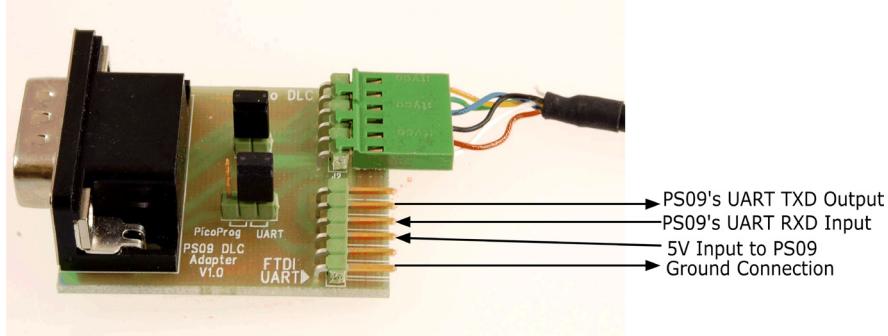
5.3 Testing a UART Program

The signals of the UART, the Transmit (TXD) and Receive (RXD) lines are available at the adapter board and can be connected through the appropriate interface cable. One possible interface to develop and test the UART programs is the TTL-232R-3V3² cable, a USB TTL serial cable from FTDI. This cable connects to the USB port of the PC and emulates a COM port at the PC. At the other end the cable can connect to a serial interface like UART. Hence this cable can be directly inserted into the connector on the adapter board labeled 'UART FTDI'. This connection with the PC enables to observe the UART communication on a HyperTerminal application on the PC.

Note: When this TTL-232R-3V3² FTDI cable is connected, the power supply of 5 V for the DLC Board comes from the cable.

However as already stated, the UART signals from the PS09 can be connected to an external serial

Figure 5-3 Adapter board with labeled UART signals



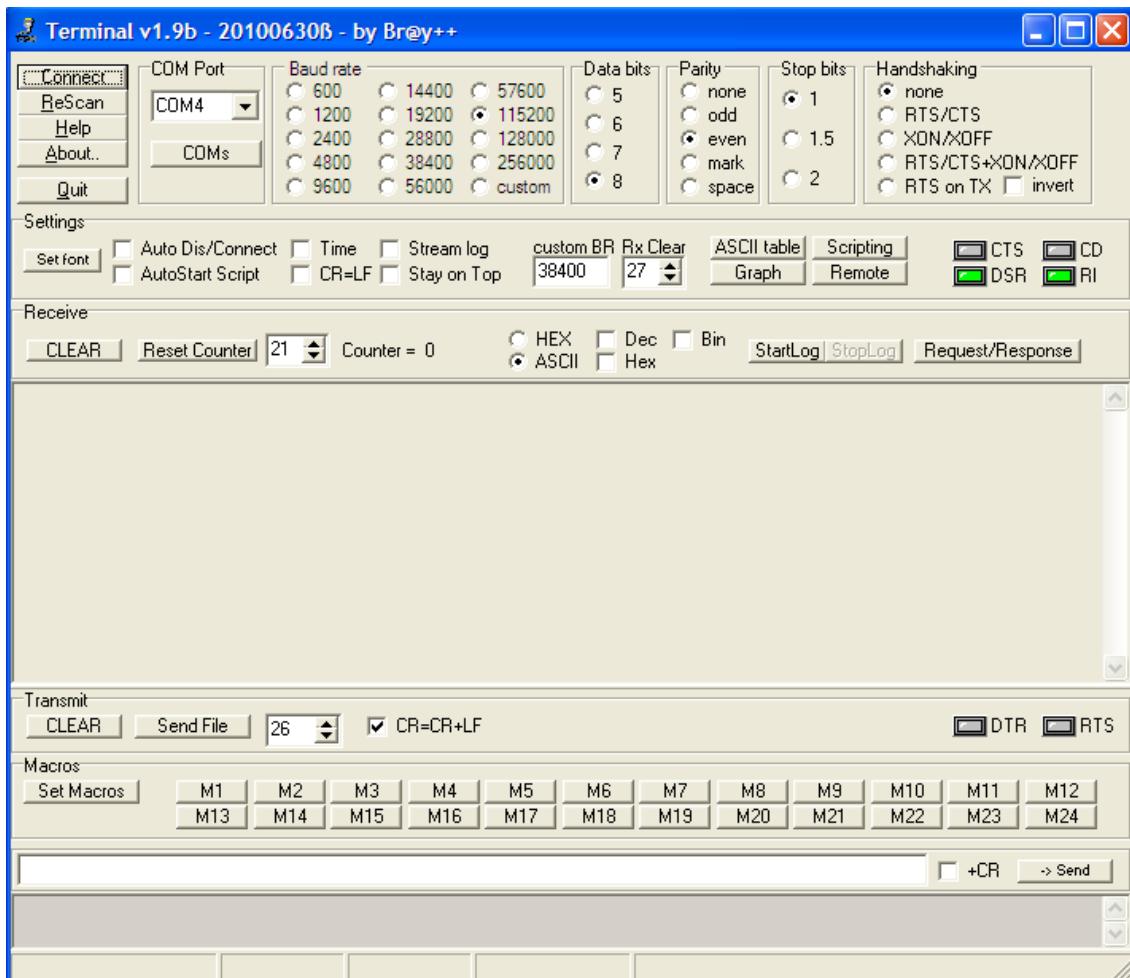
interface in any other standard manner. Please note that the 5V supply for the DLC board must also be provided for externally. The Figure 5-3 shows the UART signals labeled on the connector on the adapter board.

The result of the program, i.e. the bytes sent by the UART can be observed on a HyperTerminal program on the PC. In the folder RS232-Terminal-Program on the CD, there is a freeware HyperTerminal program – *Terminal v1.9b.exe* for installation. The following is a screenshot of the same.

² The TTL-232R-3V3 cable from FTDI available from Farnell, www.farnell.com (Order No.1329311).

Figure 5-4

Screenshot of Terminal v1.9b software



5.3.1 How to use the Terminal v1.9b program?

When the Sample Program 1 (from Appendix) has been downloaded into the external EEPROM of the PS09 DLC EVA board, the PS09 is ready to send the UART bytes. For the UART bytes from the PS09 to appear on the screen of the Terminal program, the settings on the top of *Terminal* window have to be adjusted accordingly as follows.

- The serial communication (*COM*) port to which the PS09 UART lines are connected has to be selected.
- The baud rate that has been programmed in the PS09 UART has to be selected. The Program 1 configures a baud rate of *115200 baud*; hence this has to be selected in the Terminal window.
- The Data bit is selected as *8*

- The Parity is selected as *Even*
- Handshaking is set as *None*.

Then press the *Connect* button at the top left corner of the Terminal Window. This now establishes a connection between the *Terminal* software and the PS09 UART interface. Details on how to run the program and a screenshot of the result of the UART program observed using the *Terminal* software is also available with the program in the Appendix.

Notes:

- Depending on whether the power for the DLC comes from the PICOPROG programmer or the UART connector, the second jumper in Figure 5-3 has to be set to '*Picoprog*' or '*UART*' respectively. See Chapter 1 for more details.
- Once the UART programs are downloaded into the external EEPROM through the PICOPROG programmer, they can be run on the PS09 either in front end mode or in stand-alone mode. Basically, front end mode implies that connection to PICOPROG programmer exists. Stand-alone mode means that the programmer is disconnected. Please see Chapter 1 for details on how to set front end and stand-alone mode.

6 Miscellaneous

6.1 Literature Guide

Datasheets

Title	Document-No	Date
PS09 Single Chip Solution for Strain Gauges	DB_PS09_en V0.3	May 2011
PS09-EVA-KIT Evaluation System for PS09	DB_PS09-EVA_en V0.1	May 2011

White Papers

Title	Document-No	Date
How to Lower Gain and Offset Drift Drift of a Load Cell by using TGGain and TKOffset Factors of PS081	WP002 V1.0	October 2008
Construction Guideline for solar driven Scales	WP001 V1.0	June 2008

Application Notes

Title	Document-No	Date
Meterological Investigations of PS081 Determining Zero Drift and Gain Drift	AN018 V1.0	July 2008
Strain Gauge Wiring with PICOSTRAIN	AN012 V1.0	August 2005
Rspan by Temp Compensation Compensation of Gain error for uncompensated Load Cells	AN021 V1.0	July 2009
Design Guideline for Building a Solar Kitchen Scale	AN022 V1.1	August 2009
Design Guideline for Building a Solar Body Scale	AN023 V1.3	September 2009

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

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

6.2 Document History

07.06.2011 - First Release

7 Appendix..... 7-2

7.1 Sample Program 1.....	7-2
7.1.1 Assembler Program	7-2
7.1.2 Program result.....	7-5
7.2 Sample Program 2.....	7-6
7.2.1 Assembler Program	7-6
7.2.2 Program Result	7-8
7.3 Schematic Diagrams	7-9
7.4 Layout PS09-DLC-EVA.....	7-10

7 Appendix

7.1 Sample Program 1

Transmission using UART of PS09

7.1.1 Assembler Program

```
-----
; File: uart_send_demo.asm
-----
; The PS09 jumps into the user code at the end of every measurement
; because otp_usr_prg=1 in config.h file.
-----
; This program uses the built in UART in PS09 to send 16 bytes of
; data through the GPIO4 at the end of each measurement.
; GPIO4 acts as the transmit line (TXD) of the UART
; On every POR, the UART is configured to transmit. (cfg_uart_transmit)
; The UART configuration in Config.reg 11 and 13 are best done as
; part of the autoconfig in config.h file.
; However it is shown here in code for better understanding
; Author: VK
-----

#include "config.h"

; On Power on reset, the UART is configured for transmission

ramadr    224+ 22      ; Checking for POR (flg_rstpwrr)
skipBitC  r, 19, 2      ; If yes, configure UART for transmission
jsub      cfg_uart_transmit
goto      end

ramadr    224+22 ; Check if End of Measurement?
skipBitC  r, 16, 2

; If yes, then initialising bytes to be sent through the UART
; and Start transmission

jsub      initialise_uart_bytes
goto      send_uart_bytes      ; Start transmission

goto      end                  ; If not End of measurement , then terminate

-----
initialise_uart_bytes:

ramadr    86          ; Bytes 1, 2, 3 for the transmit buffer
move      r, 0x434241

ramadr    87          ; Bytes 4, 5, 6 for the transmit buffer
move      r, 0x464544
```

```

ramadr      88          ; Bytes 7, 8, 9 for the transmit buffer
move        r, 0x494847

ramadr      89          ; Bytes 10,11, 12 for the transmit buffer
move        r, 0x4C4B4A

ramadr      90          ; Bytes 13, 14, 15 for the transmit buffer
move        r, 0x4F4E4D

ramadr      84          ; Bytes 16 for the transmit buffer
move        r, 0x510000
jsubret
;-----
send_uart_bytes:
ramadr      91          ; Starting the transmit operation
;-----
bitset      r, 10         ; Pulse to start Transmission
nop
nop
nop
bitclr      r, 10
;-----
end:       clrwdt
stop

;----- End of main program-----


; ----- Subroutines-----
; The configs in Config.Reg 11 and 13 for the UART can be done in config.h itself
; It is shown here only for better understanding
cfg_uart_transmit:

; General UART configurations
ramadr      91
bitset      r, 19          ; uart_clk_en = 1
bitset      r, 18          ; uart_4Mhz_divider =1
bitset      r, 13          ; uart_mode = 1
and         r, 0xFFFF0F    ; Masking bits uart_baud_rate
or          r, 0x0000A0    ; uart_baud_rate =0xA----> 115200 baud

; Transmit module configs.
ramadr      48+13         ; Config. reg 13
bitclr      r, 3           ; multio4_sel=2 -> GPIO4 as UART TXD
bitclr      r, 2
bitset      r, 1
bitclr      r, 0

```

```

ramadr        48+11      ; Config. reg 11
bitclr        r, 9
bitclr        r, 8      ; io_en_4 = 00---> GPIO4 as output

ramadr        91
bitclr        r, 11     ; uart_par = 0 - Even parity

bitset         r, 20      ; uart_tx_cnt[4:0] = 16
bitclr        r, 3
bitclr        r, 2
bitclr        r, 1
bitclr        r, 0

ramadr        48+3      ; Config.reg 3
bitset         r, 20      ; uart_en = 1

jsub          delay      ; Delay for the UART core to be enabled

jsubret
;-----
delay:
move           z, 0x0000FF
dly_loop:
decr           z
gotoNE         dly_loop
jsubret

;*****File: config.h*****
;*****Author: VK*****
;-----


equal 0x230042 ; Config Register 0
equal 0x2C44C7 ; Config Register 1
equal 0x5044D5 ; Config Register 2
equal 0x8200B9 ; Config Register 3
equal 0x400000 ; Config Register 4
equal 0x400000 ; Config Register 5
equal 0x400000 ; Config Register 6
equal 0x400000 ; Config Register 7
equal 0x100000 ; Config Register 8
equal 0x000000 ; Config Register 9
equal 0x14FBA4 ; Config Register 10
equal 0x00203F ; Config Register 11

```

```

equal 0x251204 ; Config Register 12
equal 0x74E142 ; Config Register 13
equal 0x60A031 ; Config Register 14
equal 0x800501 ; Config Register 15

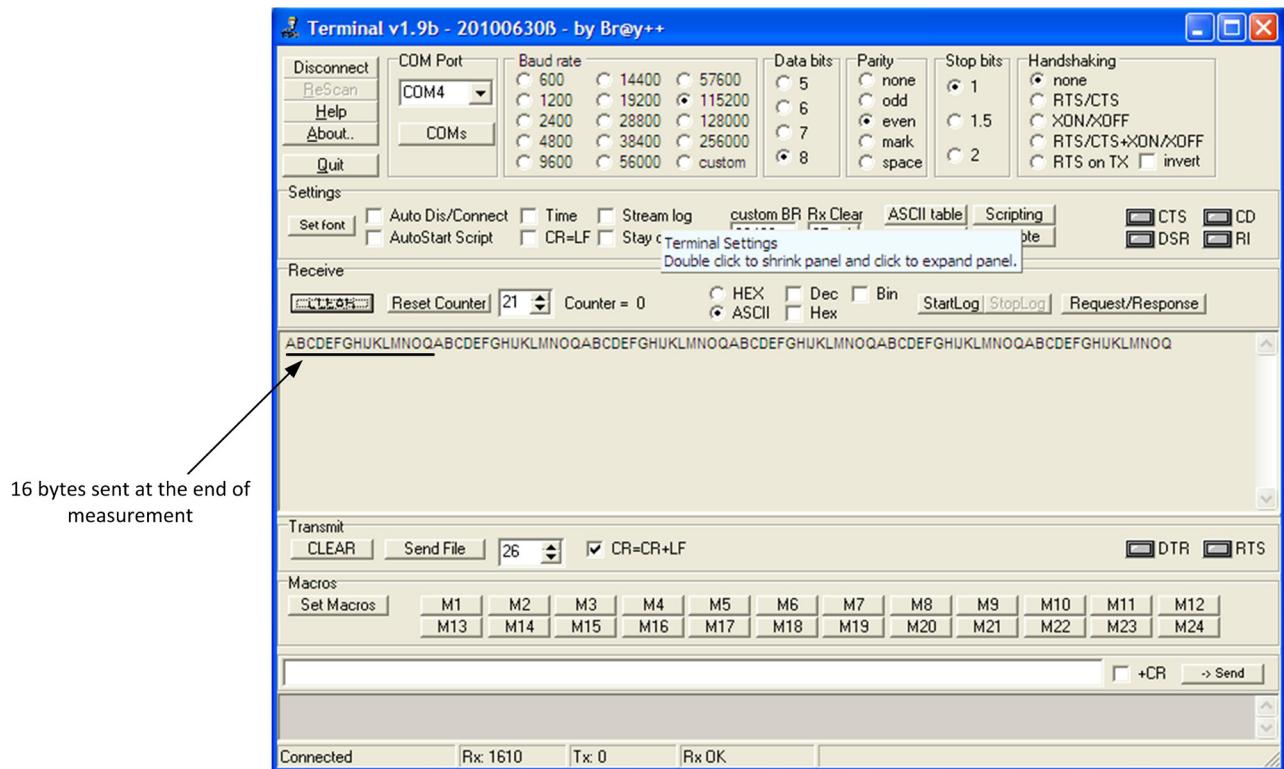
```

7.1.2 Program result

Download the program into the external EEPROM on the DLC EVA board through the PICOPROG programmer. Send a power on reset to the PS09 so the chip is configured and the program starts running (Front end Mode). The UART starts sending 16 bytes after each measurement completion. This can be received on the HyperTerminal application as shown in the following screenshot.

Please note that after programming if the PICOPROG is disconnected from adapter board, then the PS09 has to operate in Stand-alone mode (Jumper J3 open, See Chapter 1 for details). In this case, the power for the DLC has to come from the UART connector, hence set the jumper on the adapter board to 'UART'.

Figure 7-1 Result : UART transmit program



7.2 Sample Program 2

Reception using UART of PS09

7.2.1 Assembler Program

```

;-----
;-----
; File: uart_receive_demo_DLC.asm
; This program is executed when the UART in PS09 on a DLC EVA board
; has received a byte of data through the RXD Line (GPIO5)
; The PS09 is configured to interrupt the DSP when the UART receive interrupt is
generated.
; The received byte from the UART is stored in the data EEPROM at address 1
; and at RAM Address 110. You can read the received byte through the Labview Software
; and then restart the program in the EEPROM by a Power on reset through the Labview
software.
; On every POR, the UART is configured to receive.
; The UART configuration in Reg 11 is best done as part of the autoconfig in config.h file.
; Author : VK
;-----

#include "config.h"

; On Power on reset, the UART is configured for transmission

ramadr    224+ 22           ; Checking for POR (flg_rstpwrr)
skipBitC   r, 19, 1          ; If yes configure UART for receiving
jsub      cfg_uart_receive

; Checking for receive interrupt flag(scon_o<0>), if cleared, then terminate
ramadr    80
gotoBitC  r, 8, end

store_received_byte:
move      x, r               ; Moving the content to x akku
and       x, 0x0000FF         ; Masking the Lower 8 bits
ramadr    1
putepr   x      ; Storing the received data in CEEPROM at addr.1
ramadr    110
move      r, x   ; Storing the received data in RAM addr.110

; Setting the uart_ri_ack to enable further interrupts on reception
ramadr    91
bitset   r, 17
bitclr   r, 17

end:     clrwdt
stop

----- End of main program-----

```

```

; ----- Subroutines-----
; The configuration in Config.reg 11 can be done as part of the autoconfig in config.h

cfg_uart_receive:

; General UART configurations
ramadr    91
bitset    r, 19           ; uart_clk_en = 1
bitset    r, 18           ; uart_4Mhz_divider =1
bitset    r, 13           ; uart_mode = 1
and       r, 0xFFFF0F      ; Masking bits uart_baud_rate
or        r, 0x0000A0      ; uart_baud_rate =0xA ----> 115200 baud

; Receive module configs.

ramadr    91
bitset    r, 9            ; uart_rdx_sel=3 -> GPIO5 as UART RXD
bitset    r, 8
bitset    r, 15           ; uart_auto_det_stop = 1
bitclr   r, 16           ; uart_mpcomm = 0
bitset    r, 23           ; irq_uart_en = 1 - Enabling DSP
                           ; interruption on uart receive
                           ; Config. reg 11
ramadr    48+11
bitset   r, 11
bitset   r, 10           ; io_en_5_mio = 3---> GPIO5 as input
bitset   r, 21           ; io_en_digital [5] = 1, enabling input
                           ; buffer for GPIO5
ramadr    91
bitset   r, 12           ; uart_rec_en = 1 ; Enabling UART

ramadr    48+3
bitset   r, 20           ; Config.reg 3
                           ; uart_en = 1

jsub     delay           ; Delay for the UART core to be enabled

jsubret
;-----delay:
move     z, 0x0000FF

dly_loop:
decr     z
gotoNE   dly_loop
jsubret
*****
```

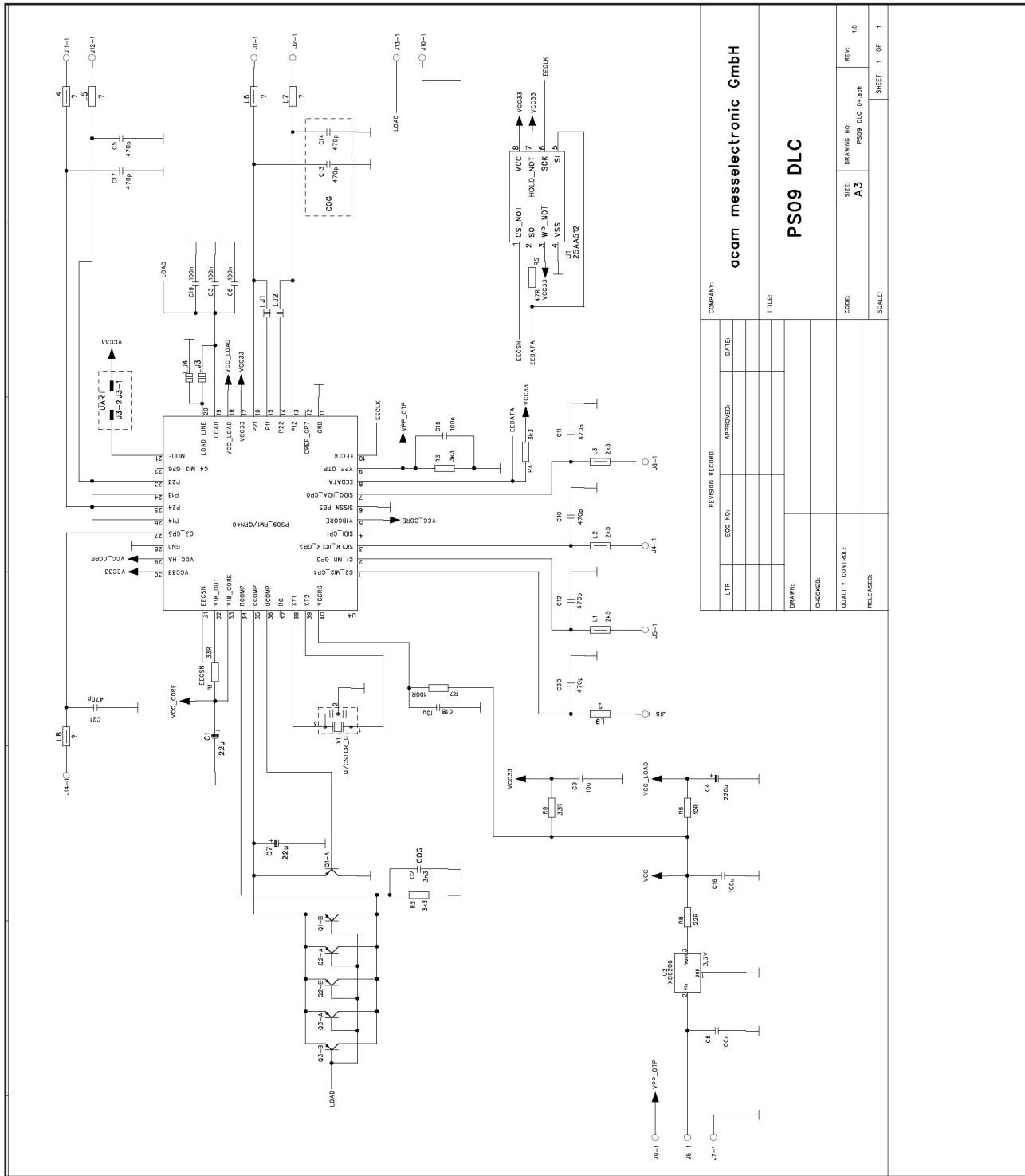
```
;-----  
;      File: config.h  
;      Author: VK  
;  
  
equal 0x230042 ; Config Register 0  
equal 0x2C44C7 ; Config Register 1  
equal 0x5044D5 ; Config Register 2  
equal 0x8200B9 ; Config Register 3  
equal 0x400000 ; Config Register 4  
equal 0x400000 ; Config Register 5  
equal 0x400000 ; Config Register 6  
equal 0x400000 ; Config Register 7  
equal 0x100000 ; Config Register 8  
equal 0x000000 ; Config Register 9  
equal 0x14FBA4 ; Config Register 10  
equal 0x00203F ; Config Register 11  
equal 0x251204 ; Config Register 12  
equal 0x74E142 ; Config Register 13  
equal 0x60A031 ; Config Register 14  
equal 0x800501 ; Config Register 15
```

7.2.2 Program Result

Download the program into the external EEPROM on the DLC EVA board through the PICOPROG programmer. Send a power on reset to the PS09 so that the chip is configured by the auto configuration and the program starts running (Front end mode). Send a UART byte through the HyperTerminal application to the PS09. The above program receives the UART byte and stores it in the Data EEPROM at address 1 and RAM Address 110. You can read the received byte through the PS09 evaluation software. To start running the program again, send a power on reset through the evaluation software.

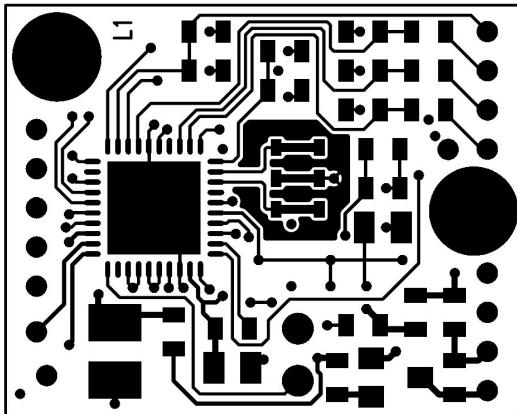
Please note that after programming if the PICOPROG is disconnected from adapter board, then the PS09 has to operate in Stand-alone mode (Jumper J3 open, See Chapter 1 for details). In this case, the power for the DLC has to come from the UART connector, hence set the jumper on the adapter board to ‘UART’.

7.3 Schematic Diagrams

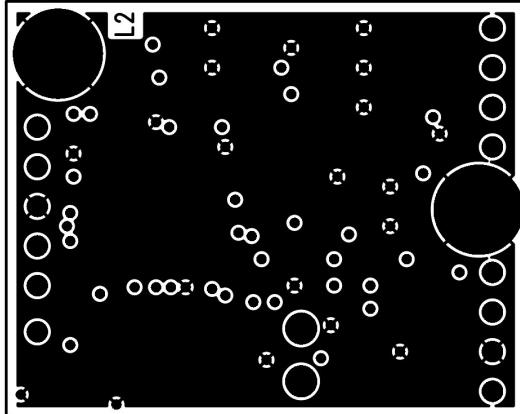


7.4 Layout PS09-DLC-EVA

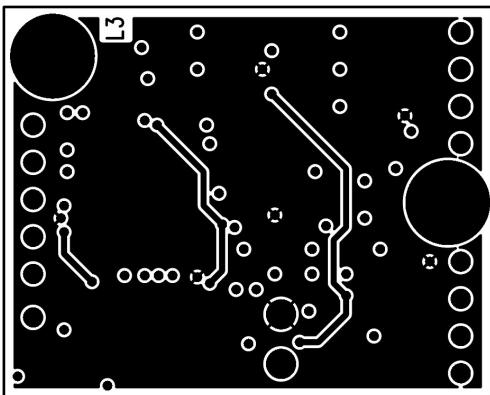
Layer 1



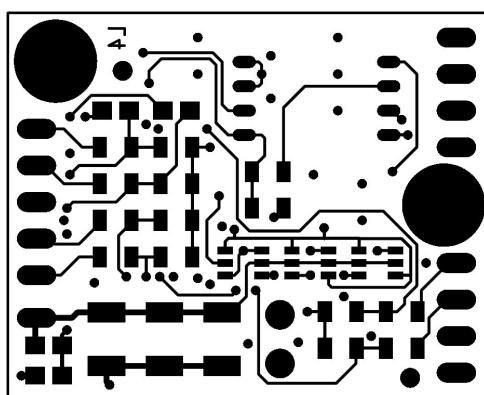
Layer 2



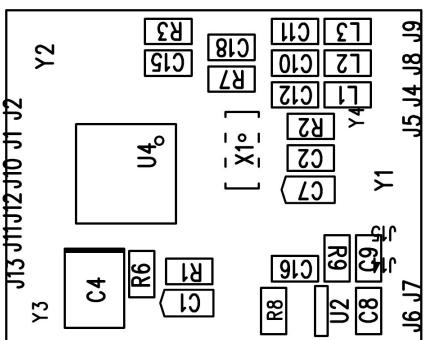
Layer 3



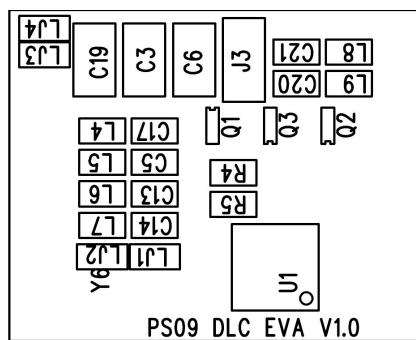
Layer 4

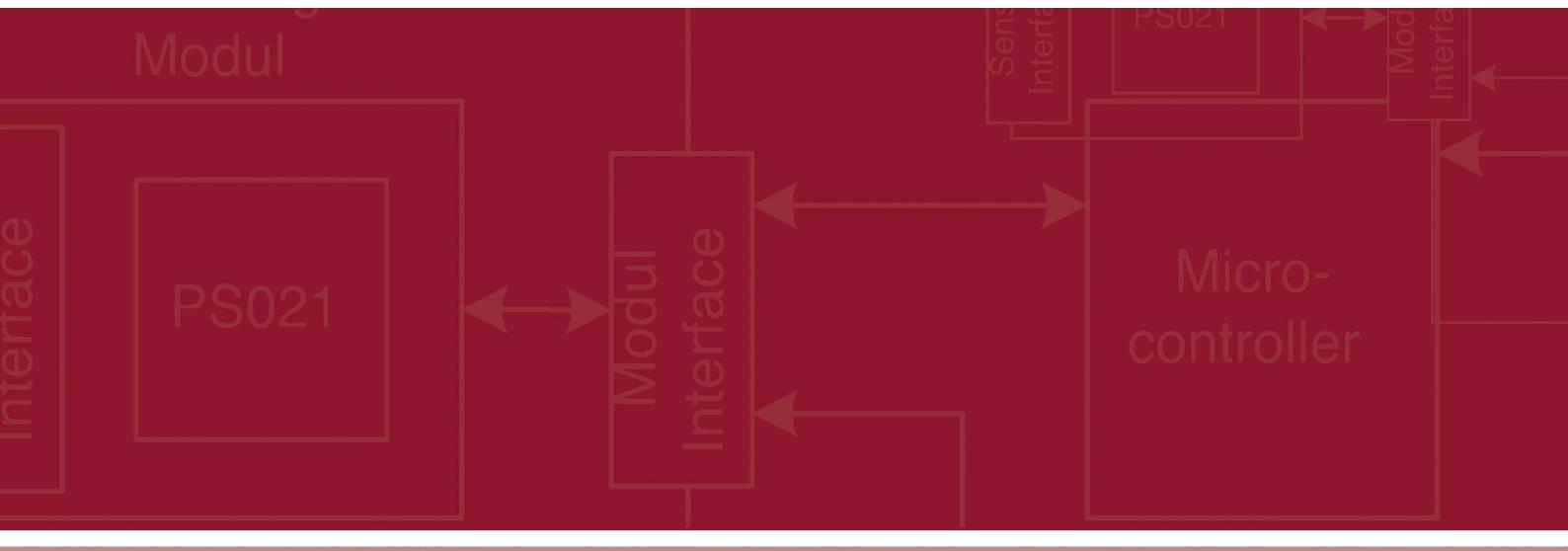


Top Layer placement



Bottom Layer placement





acam-messelectronic gmbh

Am Hasenbiel 27

76297 Stutensee-Blankenloch

Germany / Allemagne

ph. +49 7244 7419 - 0

fax +49 7244 7419 - 29

e-mail: support@acam.de

www.acam.de