Freescale Semiconductor
Application Note

AN3785
Rev 5, 7/2009

How to Implement the Freescale MPL115A Digital Barometer

by: John Young

INTRODUCTION

MPL115A is a simple barometer with digital output and high performance targeting low cost commercial applications. The device employs a MEMS PRT pressure sensor with a conditioning IC to provide accurate pressure data. The sensor output is accurate to ±1 kPa over the 50 kPa to 115 kPa pressure range. An integrated ADC provides digitized temperature and pressure sensor outputs via an I^{2}C or SPI bus. The part's operating voltage is from 2.375 V to 5.5 V. The part is available as either an I^{2}C or SPI part. The customer can order the device in either bus protocol depending on which is more convenient for their end application.

Calibration data is housed in on-board ROM. This data is used by a host microcontroller to apply a compensation algorithm to the raw ADC data from the pressure sensor and may be accessed at any time. The Calibration data is stored as a series of coefficients that are applied to the raw data to compensate for temperature and pressure variation in the raw output.

The following application note will describe the SPI and I^{2}C protocol needed to communicate to the digital pressure sensor. Following communication and extraction of the stored coefficients, raw data from the sensor's PRT can be compensated on the host microcontroller for pressure and temperature. The application note details this process and the end result of the calculation is Compensated Pressure.

Sample Boards:

Freescale has two sample eval boards available in either SPI or I^{2}C MPL115A device variations; KITMPL115A1SPI or KITMPL115A2I2C. These are simple PCB boards with sensor parts, associated pull-up resistors and decoupling capacitors soldered onboard for customer evaluation. This provides a quick sample tool to run evaluations for the small device given its LGA footprint.

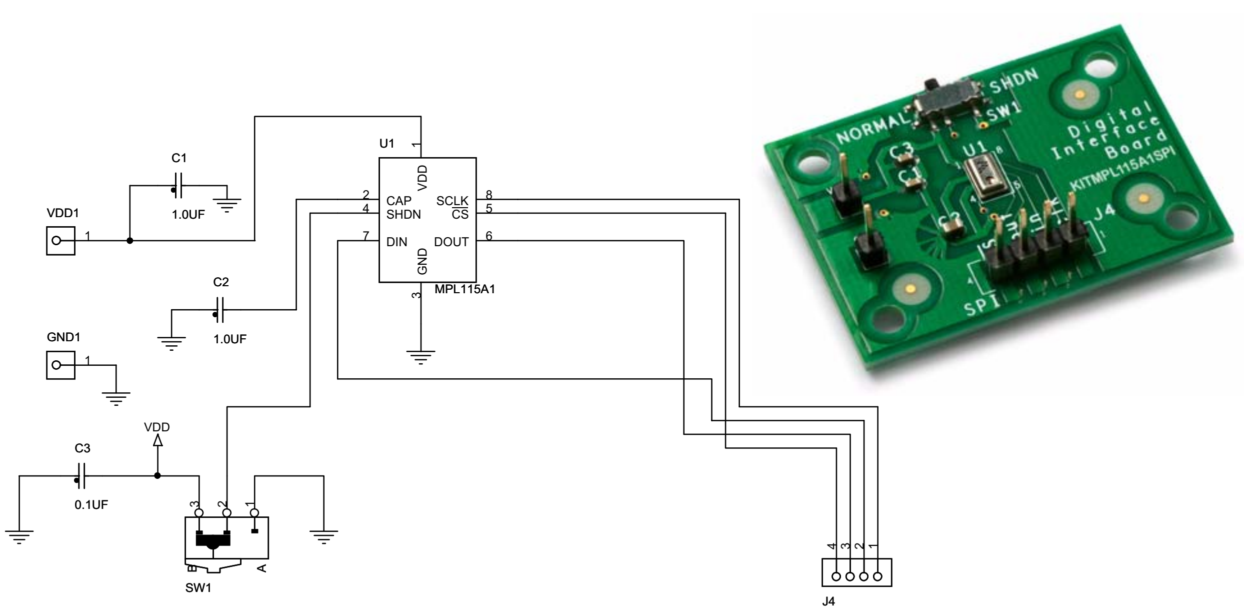


Figure 1. KITMPL115A1SPI: MPL115A1 (SPI) Interface Board

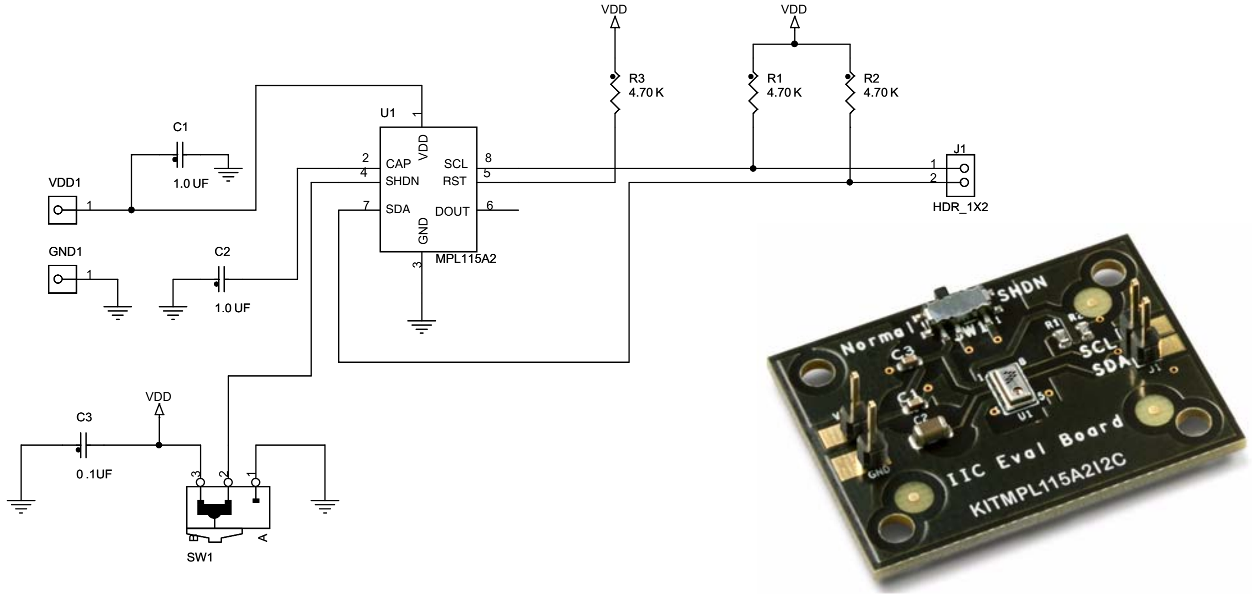


Figure 2. KITMPL115A2I2C: MPL115A2 (I^2C) Interface Board

PRODUCT FEATURES

The MPL115A is an absolute pressure sensor with digital output for low cost applications. A miniature 5 x 3 x 1.2 mm LGA package ideally suits it for portable electronics and space constrained applications. Low current consumptions of 5 μA during Active mode and 1 μA during Shutdown (Sleep) mode target battery and other low-power applications. A wide operating temperature range from -40°C to +105°C fits demanding environmental requirements.

MPL115A employs a MEMS pressure sensor with a conditioning IC to provide accurate pressure measurement from 50 to 115 kPa. An integrated ADC provides digitized temperature and pressure sensor outputs via an I^2C or SPI port. Calibration Data is stored in internal ROM. Utilizing raw sensor output, the host microcontroller executes a compensation algorithm to render *Compensated Absolute Pressure* with 1 kPa accuracy.

The MPL115A pressure sensor's small form factor, low power capability, precision, and digital output optimize it for barometric measurement applications

ABSOLUTE MAXIMUM RATINGS

Voltage (with respect to GND unless otherwise noted)
V_{DD} .... -0.3 V to +5.5 V
SCLK, $\overline{CS}$, D_{IN}, D_{OUT} .... -0.3 V to V_{DD}+0.3 V
Operating Temperature Range .... -40°C to +105°C
Storage Temperature Range.. -40°C to +125°C

ELECTRICAL CHARACTERISTICS

(V_{DD} = 2.375 V to 5.5 V, T_A = -40°C to +105°C, unless otherwise noted. Typical values are at V_{DD} = 3.3 V, T_A = +25°C.

Ref	Parameters	Symbol	Conditions	Min	Typ	Max	Units
1	Operating Supply Voltage	V_{DD}		2.375	3.3	5.5	V
2	Supply Current	I_{DD}	Shutdown (SHDN = GND)	—	—	1	μA
			Standby	—	3.5	10	μA
			Average – at one measurement per second	—	5	—	μA
Pressure Sensor							
3	Range			50	—	115	kPa
4	Resolution			—	0.15	—	kPa
5	Accuracy		-20ºC to 85ºC	—	±1	—	kPa
6	Conversion Time (*Start Pressure Convert*)	tcp	Time between start convert command and data available in the Pressure register	—	3	—	ms
Temperature Sensor							
7	Range			-40	—	105	ºC
8	Conversion Time (*Start Temperature Convert*)	tct	Time between start convert command and data available in the Temperature register	—	3	—	ms
9	Conversion Time (*Start Both Convert*)	tcb	Time between start convert command and data available in the Pressure and Temperature registers	—	3	—	ms
10	Resolution		Temperature ADC is 472 counts at 25ºC		-5.35		counts/ºC
SPI Inputs: SCLK, $\overline{CS}$, D_{IN}							
11	SCLK Clock Frequency	f_{SCLK}	[1]	—	—	8	MHz
12	Low Level Input Voltage	VIL		—	—	$0.3V_{DD}$	V
13	High Level Input Voltage	VIH		0.7 V_{DD}	—	—	V
SPI Outputs: D_{OUT}							
14	Low Level Output Voltage	VOL1	At 3 mA sink current	0	—	0.4	V
		VOL2	At 6 mA sink current	0	—	0.6	
15	High Level Output Voltage	VOH1	At 3 mA source current	V_{DD} – 0.4 V	—	—	V
I^2C I/O Stages: SCL, SDA							
16	SCL Clock Frequency	f_{SCL}		—	—	400	KHz
17	Low Level Input Voltage	VIL		—	—	$0.3V_{DD}$	V
18	High Level Input Voltage	VIH		$0.7V_{DD}$	—	—	V
I^2C Outputs: SDA							
19	Data Setup Time	t_{SU}	Setup time from command receipt to ready to transmit	100	—	—	ns
I^2C Addressing							
20	MPL115A2 uses 7-bit addressing, does not acknowledge the general call address 0000000. Slave address has been set to 0x60 or 1100000.						

NOTES:
1. Nominal maximum SPI clock frequency.

PIN CONFIGURATION

MPL115A2 (I^2C)　　**MPL115A1 (SPI)**

Figure 3. Top View

PIN DESCRIPTION (I^2C/SPI)

PIN	NAME	FUNCTION
1	VDD	**VDD Power Supply Connection.** VDD range is 2.375 V to 5.5 V.
2	CAP	**External Capacitor** Output decoupling capacitor for main internal regulator. Connect a 1 μF capacitor to ground.
3	GND	**Ground**
4	SHDN	**Shutdown** Connect to GND to disable the device. When in shut down the part draws 1 μA supply current and all communications pins (RST/$\overline{\text{CS}}$, SCL/SCLK, SDA/DOUT, NC/DIN) are high impedance. Connect to VDD for normal operation.
5	RST/$\overline{\text{CS}}$	**Reset bar (I^2C) / $\overline{\text{CS}}$ (SPI)** I^2C devices: Drive line low to disable I^2C communications. I^2C pins are high impedance and communications are ignored. All internal functions operate normally. SPI devices: SPI Chip Select line. See SPI communications section for details.
6	NC/DOUT	**NC (I^2C) / DOUT (SPI)** I^2C devices: No connection. SPI devices: Serial data output.
7	SDA/DIN	**SDA (I^2C) / DIN (SPI)** I^2C devices: Serial data I/O line. SPI devices: Serial data input.
8	SCL/SCLK	**I^2C/SPI Serial Clock Input.**

DETAILED DESCRIPTION

The system comprises a pressure sensor and a conditioning IC housed within a low profile package. The conditioning IC contains a temperature sensor together with an ADC and some ROM for coefficient storage. The host microcontroller communicates with MPL115A via the I^2C or SPI port and performs the calculations necessary to obtain calibrated pressure from the raw ADC values.

MPL115A uses standardized calculation algorithms to determine pressure using the temperature and pressure ADC output values together with stored coefficient values.

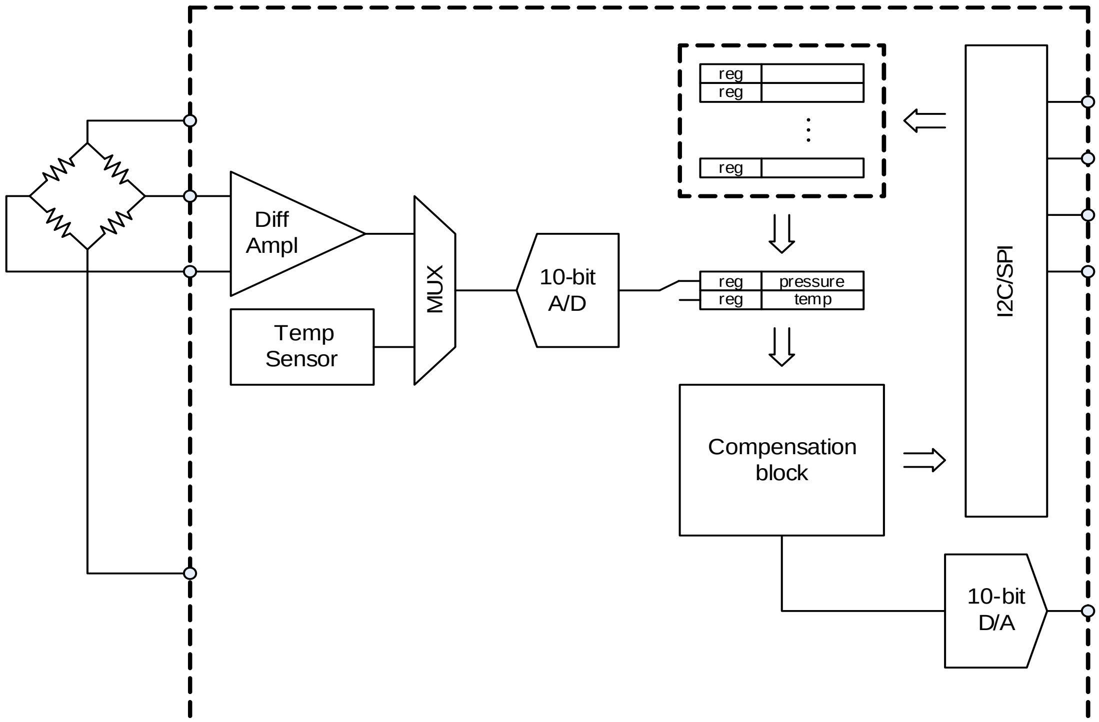


Figure 4. MPL115A Block Diagram

BASIC OVERVIEW OF FUNCTIONS/OPERATION

MPL115A will interface to a host (or system) microcontroller in the customer's application. All communications are via I^2C or SPI. A typical usage sequence is as follows:

Initial Power Up

All circuit elements are active. I^2C/SPI port pins are high impedance and associated registers are cleared. On conclusion of the startup routine the main system circuits shut down leaving the part in standby "wake on I^2C" mode or, in the case of SPI devices, in the communication mode determined by the $\overline{CS}$ line.

When using the SPI version, Chip Select ($\overline{CS}$) only controls communications and does not reset the part.

Reading Coefficient Data

The user then typically accesses the part and reads coefficient data. The main circuits within the slave device are disabled during read activity. The coefficients are typically stored in host micro local memory but can be uploaded at any time. The part always enters standby mode at the conclusion of the start-up routine or conversion activity.

For longer periods of inactivity the user may assert the SHDN input (drive input low). This removes power from all internal circuits, including any registers. The registers are then reloaded when SHDN is driven high.

Data Conversions

These are initiated by sending the *start pressure convert*, *start temperature convert* or *start both convert* request via I^2C/SPI. The main system circuits are activated (wake up) in response to a valid start conversion command. The result of the conversion is placed in the appropriate register (Pressure, Temperature or both). The typical wait time is 3 ms associated with each conversion type.

Compensated Pressure Reading

The host microcontroller manipulates the stored coefficient data with the now incoming changing pressure and temperature data. Utilizing a standard equation (detailed later) data is manipulated for compensation and the raw pressure and temperature data now becomes the current barometric/atmospheric pressure. Remember that this is an absolute pressure measurement with a vacuum as a reference.

Communications

MPL115A normally operates as either an I^2C or SPI slave device depending on the condition of the communications bit in the setup register. Special use cases: test and debug force the part into a predetermined mode.

I^2C Interface

I^2C mode MPL115A operates as an I^2C slave capable of bus speeds up to 400 kbits/sec (Fast Mode). MPL115A uses 7-bit addressing, does not acknowledge the general call address 0000000, and does not perform clock stretching. *The current address set up on the MPL115A is 0x60 or 1100000.*

The I^2C port pins SDA and SCL are compatible with the full supply voltage range. Data is written to and read from the device in bit sequence as 8-bit blocks.

Write Mode

I^2C writes to the device are conducted using normal I^2C protocol. An I^2C start condition is followed by the 7-bit slave address and *Write* bit. The slave acknowledges the write request and latches the following data byte from the master. Further bytes are ignored when in normal (user) mode. A stop or repeat start condition must be sent by the master to complete the sequence. User mode write commands consist of start convert commands for pressure, temperature or both. These commands are shown in Table 2. An example of the user mode I^2C write sequence is shown in Figure 5.

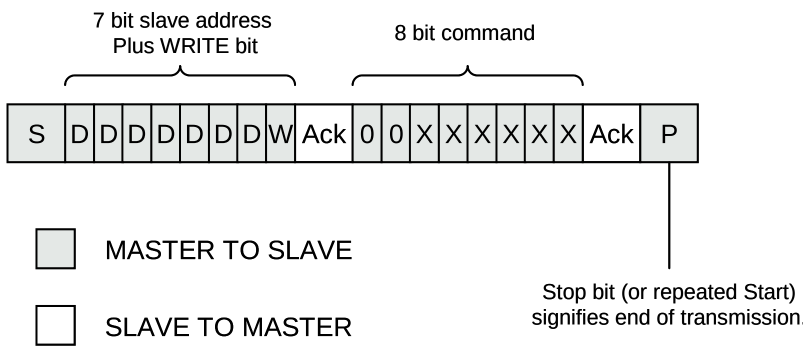


Figure 5. I²C Write Sequence – User Mode

Read Mode

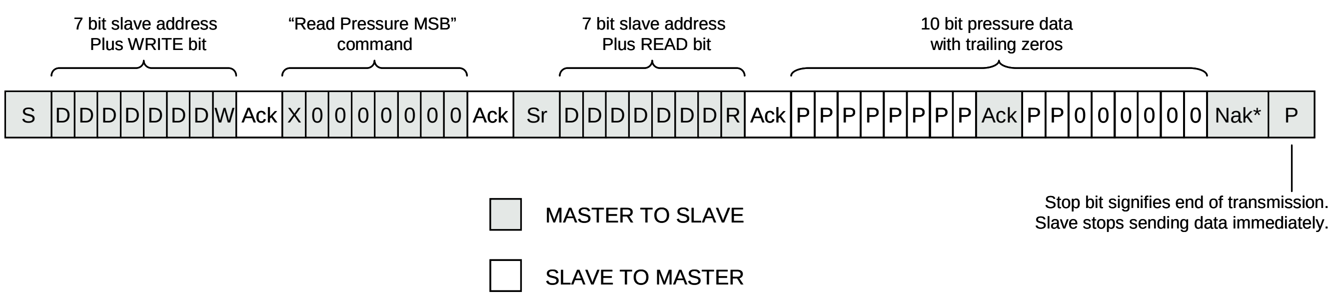
I^2C reads from the device are conducted using normal I^2C protocol.

To read from the device the master must first send a data address or command that points to the required data. To do this the master sends the 7-bit slave address + *Write* bit followed by data address byte. Transmission is initiated by an I^2C start condition and ended by either a *Repeated Start* or a *Stop* condition. The master then resends the 7-bit slave address, this time with a *Read* bit, either directly following the *Repeated Start* condition or, if the previous transmission ended with a *Stop* condition, preceded by an I^2C start condition.

The slave acknowledges each byte received and proceeds to send the requested data following receipt of the second slave address. The slave continues to send data, incrementing the address pointer at the end of each byte. The master acknowledges receipt of each byte sent by the slave and asserts a *Stop* (or *Repeated Start*) condition at the conclusion of the transmission.

Examples of I^2C reads for pressure data (2 bytes) and coefficient data (up to 12 bytes) are shown in Figure 6 and Figure 7.

10-bit pressure and temperature data is transmitted MSB first with trailing zeros. This allows the user to read only the first byte in systems that do not require the full resolution of the part and may only have limited processing capability. The device continues to send data until either the transmission is complete or the master either issues a stop condition or fails to assert an acknowledge bit i.e. the slave treats a “Nak” (not acknowledge) as a stop bit. The master thus controls the amount of data it receives and discards any data that is not needed.



* The standard termination sequence is “Nak” followed by the stop condition. An “Ack” followed by the stop condition may also be used.

Figure 6. I²C Read Pressure Sequence

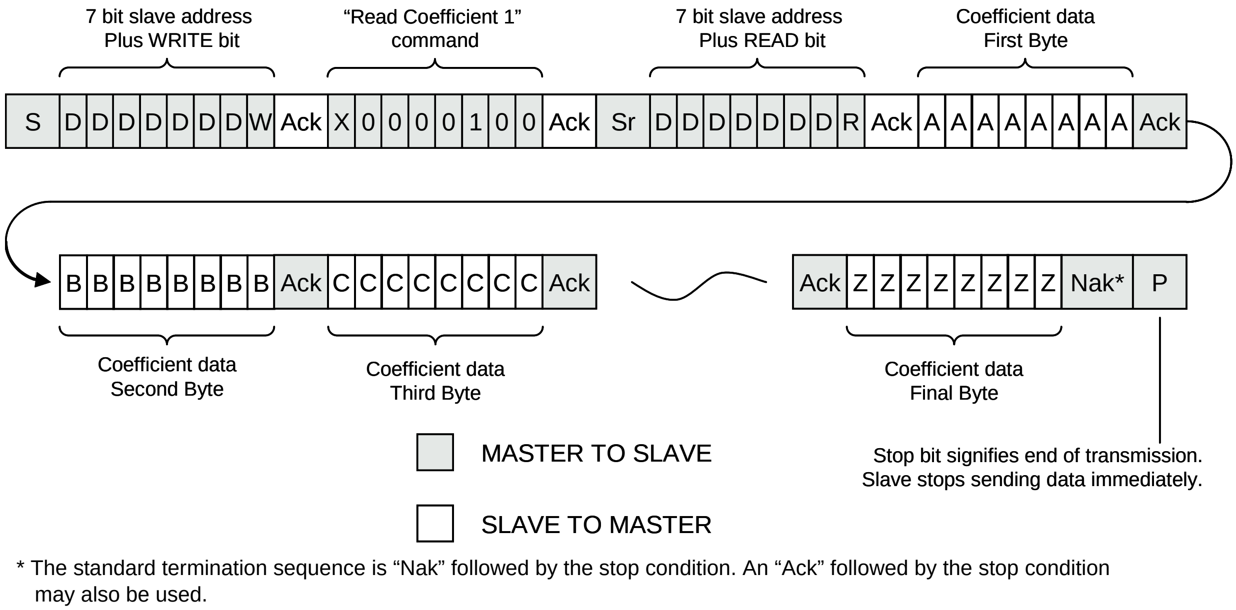


Figure 7. I²C Read Coefficients Sequence

I²C Bus Timing

Data is valid on the SDA line while the SCL line is stable and high. The slave may only change the state of SDA line while SCL is low. This is typically controlled by the falling edge of SCL.

Timing of the SCL and SDA lines is shown in Figure 8.

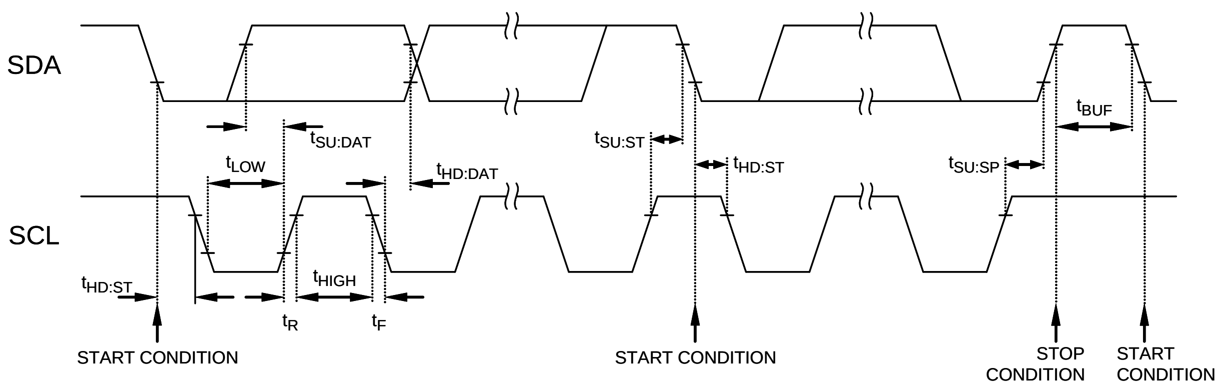


Figure 8. I²C Bus Timing

I²C RST

The I²C RST function prevents the I²C circuits from drawing power while the I²C inputs are active. Ideally this is accomplished by removing power from the I²C I/O circuits. The part ignores I²C communications when RST is asserted and does not draw additional power due to transitions on the I²C connections. Drive line low to disable I²C communications. I²C pins are high impedance and communications are ignored. If RST is asserted during an I²C transmission then the transmission is aborted (and lost). All other internal functions operate normally. No other functions or registers are reset as a result of asserting this function.

SPI Interface

In SPI mode MPL115A operates as a half duplex 4-wire SPI slave capable of bus speeds up to 8 Mb/sec.

SPI 4-Wire Mode

4 lines make up the SPI interface: SCLK, DIN, DOUT and $\overline{CS}$. Data is read from the port in 2 byte sequence: address byte plus Read/Write bit received on DIN followed by data byte transmitted on DOUT. Write commands are a 2 byte sequence: address byte plus Read/Write bit followed by data byte, both received on DIN. Exceptions to this are the “action” addresses e.g. “*Start Pressure Conversion”*, “*Start Temperature Conversion*” etc. The interface is half duplex and so cannot receive and transmit simultaneously. Minimum data setup time (interval between command receipt and data ready on DOUT) is typically 3 ms.

SPI transfers to and from the part are controlled by the $\overline{CS}$ line. $\overline{CS}$ is driven low by the master at the start of communication and held low for the duration of the two byte transfer.

SPI data transfers occur on the rising edges of SCLK. Data on the DIN and DOUT lines is changed on the falling edges of SCLK and clocked on the rising edges of SCLK: the rising edges provide the “data valid” condition.

Driving $\overline{CS}$ high places DOUT in a high impedance state. DOUT remains high impedance while $\overline{CS}$ is high. DOUT is held low while $\overline{CS}$ is low and there is no transmission activity on DOUT.

The next data transfer commences on the next $\overline{CS}$ high to low transition.

4-Wire Mode

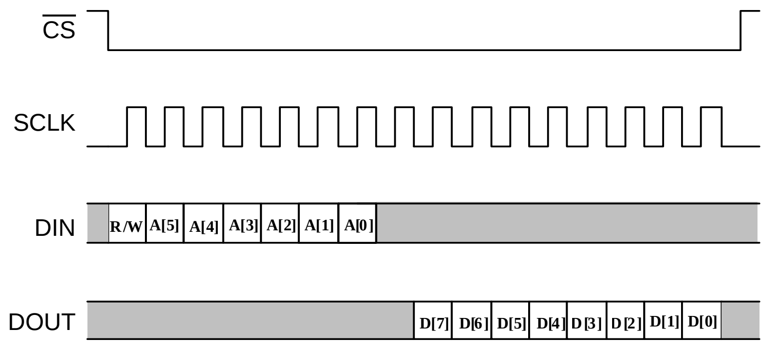


Figure 9. SPI Read Operation

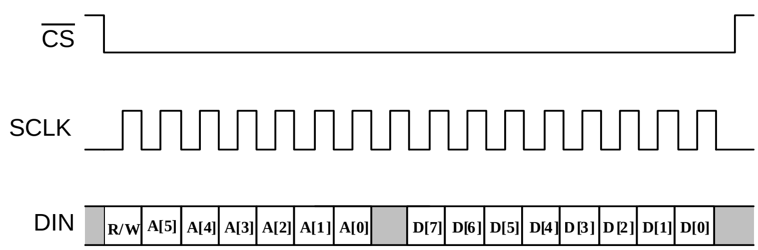


Figure 10. SPI Write Operation

DEVICE READ WRITE OPERATIONS

All device read/write operations are memory mapped. Device actions e.g. “Start Pressure Conversion” are controlled by writing to the appropriate memory address location. All memory address locations are 6-bit (see Table 5). A read/write bit (SPI) and a “don’t care” bit(s) are added to make up the total number of bits to 8.

For I^2C devices the MSB (bit 7) is “don’t care” and bits 6 to 0 are available for addressing e.g. bit 6 is always 0 with the remaining bits (5:0) making up the 6-bit address in MPL115A.

For SPI devices the MSB is R/W and the LSB is “don’t care” with the remaining bits (6:1) making up the 6-bit address. The appropriate memory location is written to/read from in response to the condition of the R/W bit which is ‘1’ for read operations and ‘0’ for write operations.

The basic device write commands are shown in Table 1 (SPI versions) and Table 2 (I^2C versions). The availability of particular commands depends on the operating mode of the device. There are three basic operating modes: Test, Debug and User (normal).

Table 1. SPI Write Commands

Command	8-bit code	Mode	Bus type
Start Pressure Conversion	0010000X	All	SPI
Start Temperature Conversion	0010001X	All	SPI
Start both Conversions	0010010X	All	SPI

X = don't care

Table 2. I^2C Write Commands

Command	8-bit Code	Mode	Bus Type
Start Pressure Conversion	X0010000	User	I^2C
Start Temperature Conversion	X0010001	User	I^2C
Start both Conversions	X0010010	User	I^2C

X = don't care

The actions taken by the part in response to each command is as follows:

Command	Action Taken
Start Pressure Conversion	Wake up main circuits. Start clock. Allow supply stabilization time. Select pressure sensor input. Perform A to D conversion, load register.
Start Temperature Conversion	Wake up main circuits. Start clock. Allow supply stabilization time. Select temperature input. Perform A to D conversion.
Start Both Conversions	Wake up main circuits. Start clock. Allow supply stabilization time. Select pressure sensor input. Apply positive sensor excitation and perform A to D conversion. Select temperature input. Perform A to D conversion. Load the *Temperature* register with the result. Shut down main circuits and clock.

SPI Read operations are performed by sending the required address with a leading *Read* bit. SPI operations require that each byte is addressed individually. During I^2C transfers the device increments the address pointer after each byte such that continuous bytes may be read from the device in address sequence. Table 3 (SPI versions) and Table 4 (I^2C versions) show some basic read commands. All data is transmitted MSB first.

Table 3. Example SPI Read Commands

Command	8-bit Code	Mode	Bus Type
Read Pressure Hi byte	1000000X	All	SPI
Read Pressure Lo byte	1000001X	All	SPI
Read Temperature Hi byte	1000010X	All	SPI
Read Temperature Lo byte	1000011X	All	SPI
Read Coefficient data byte 1	1000100X	All	SPI

X = don't care

Table 4. Example I^2C Read Commands

Command	8-bit code	Mode	Bus type
Read Pressure Hi byte	X0000000	User	I^2C
Read Pressure Lo byte	X0000001	User	I^2C
Read Temperature Hi byte	X0000010	User	I^2C
Read Temperature Lo byte	X0000011	User	I^2C
Read Coefficient data byte 1	X0000100	User	I^2C

X = don't care

DEVICE MEMORY MAP

Table 5. Device Memory Map

Address	Name	Description	Size (bits)	NVM	Accessibility (mode)		
					User	Test	Debug
$00	POUTH	10-bit pressure output value MSB	8		Yes	Yes	Yes
$01	POUTL	10-bit pressure output value LSB	2		Yes	Yes	Yes
$02	TOUTH	10-bit temperature output value MSB	8		Yes	Yes	Yes
$03	TOUTL	10-bit temperature output value LSB	2		Yes	Yes	Yes
$04	COEF1	96-bit coefficient data 1st byte	8	8	Yes	Yes	Yes
$05	COEF2	96-bit coefficient data 2nd byte	8	8	Yes	Yes	Yes
$06	COEF3	96-bit coefficient data 3rd byte	8	8	Yes	Yes	Yes
$07	COEF4	96-bit coefficient data 4th byte	8	8	Yes	Yes	Yes
$08	COEF5	96-bit coefficient data 5th byte	8	8	Yes	Yes	Yes
$09	COEF6	96-bit coefficient data 6th byte	8	8	Yes	Yes	Yes
$0A	COEF7	96-bit coefficient data 7th byte	8	8	Yes	Yes	Yes
$0B	COEF8	96-bit coefficient data 8th byte	8	8	Yes	Yes	Yes
$0C	COEF9	96-bit coefficient data 9th byte	8	8	Yes	Yes	Yes
$0D	COEF10	96-bit coefficient data 10th byte	8	8	Yes	Yes	Yes
$0E	COEF11	96-bit coefficient data 11th byte	8	8	Yes	Yes	Yes
$0F	COEF12	96-bit coefficient data 12th byte	8	8	Yes	Yes	Yes
$10	PRESS	Start Pressure Conversion			Yes	Yes	Yes
$11	TEMP	Start Temperature Conversion			Yes	Yes	Yes
$12	BOTH	Start both Conversions			Yes	Yes	Yes

I^2C SIMPLIFIED FOR COMMUNICATION

Note (Write/Read) W = 0, R = 1

7bit address = 0x60

Command to Write “Convert both Pressure and Temperature” = 0x12

Command to Read “Pressure High Byte” = 0x00

Command to Read “Coefficient data byte 1 High byte” = 0x04

Start Pressure and Temperature Conversion, Read raw Pressure:

[Start], 0x60+[W], 0x12, 0x01,[Stop]
[Restart], 0x60+[W], 0x00
[Start], 0x60+[R], 0xMSB Pressure, 0xLSB Pressure, [Stop]

Read Coefficients:

[Start], 0x60+[W], 0x04, [Stop]
[Restart], 0x60+[R], [0x04], [0x05], [0x06], [0x07], [0x08], [0x09], [0x10], [0x11], [0x12], [Stop]

Master Slave

SPI SIMPLIFIED FOR COMMUNICATION

Command to Write “Convert both Pressure and Temperature” = 0x24

Command to Read “Pressure High byte” = 0x80

Command to Read “Pressure Low byte” = 0x82

Command to Read “Temperature High byte” = 0x84

Command to Read “Temperature High byte” = 0x86

Command to Read “Coefficient data byte 1 High byte” = 0x88

Start Pressure and Temperature Conversion, Read raw Pressure:

[CS][0x24],[0x00],[0x80],[0x00],[0x82],[0x00],[0x84],[0x00,][0x86],[0x00],[0x00][CS]

Read Coefficients:

[CS][0x88][0x00][0x8A][0x00][0x8C][0x00][0x8E][0x00][0x90][0x00][0x92][0x00]
[0x94][0x00][0x96][0x00][0x98][0x00][0x9A][0x00][0x9C][0x00][0x9E][0x00][0x00]
[CS]

Master

NOTE: Extra [0x00] at the ends to output the last data byte on the slave side of the SPI.

SPI TIMING

Table 6 and Figure 11 describe the timing requirements for the SPI system.

Table 6. SPI Timing

No.	Function	Symbol	Min	Max	Unit
	Operating Frequency	—	—	8	MHz
1	SCLK Period	tSCLK	125	—	ns
2	SCLK High time	tCLKH	62.5	—	ns
3	SCLK Low time	tCLKL	62.5	—	ns
4	Enable lead time	tSCS	125	—	ns
5	Enable lag time	tHCS	125	—	ns
6	Data setup time	tSET	30	—	ns
7	Data hold time	tHOLD	30	—	ns
8	Data valid (after SCLK low edge)	tDDLY	—	32	ns
9	Width CS High	tWCS	30	—	ns

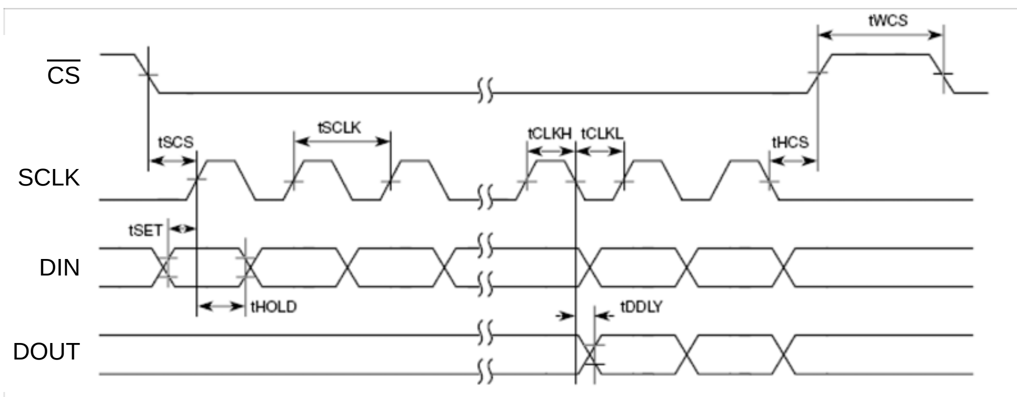


Figure 11. SPI Timing Diagram

COMPENSATION

The pressure compensation for MPL115A is based on a 2-dimensional, second order polynomial based on Microsystems mst_trimlib library.

The 10-bit compensated pressure output for MPL115A, Pcomp, is calculated as follows:

Pcomp = a0 + (b1 + c11*Padc + c12*Tadc) * Padc + (b2 + c22*Tadc) * Tadc

Where:

- Padc is the 10-bit pressure output of the MPL115A ADC,
- Tadc is the 10-bit temperature output of the MPL115A ADC,
- a0 is the pressure offset coefficient,
- b1 is the pressure sensitivity coefficient,
- c11 is the pressure linearity (2nd order) coefficient,
- c12 is the coefficient for temperature sensitivity coefficient (TCS),
- b2 is the 1st order temperature offset coefficient (TCO),
- c22 is the 2nd order temperature offset coefficient.

Ideally, Pcomp will produce a value of 0 with an input pressure of 50 kPa and will produce a full-scale value of 1023 with an input pressure of 115 kPa.

EVALUATION SEQUENCE, ARITHMETIC CIRCUITS

The following is a possible sequence for the calculation Pcomp, the trimmed pressure output. Input values are in **bold.**

c11x1 = **c11** * **Padc**

a11 = **b1** + c11x1

c12x2 = **c12** * **Tadc**

a1 = a11 + c12x2

c22x2 = **c22** * **Tadc**

a2 = **b2** + c22x2

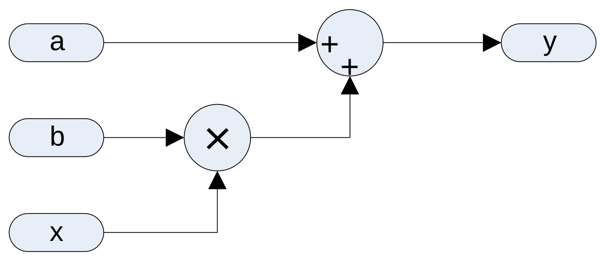
a1x1 = a1 * **Padc**

y1 = **a0** + a1x1

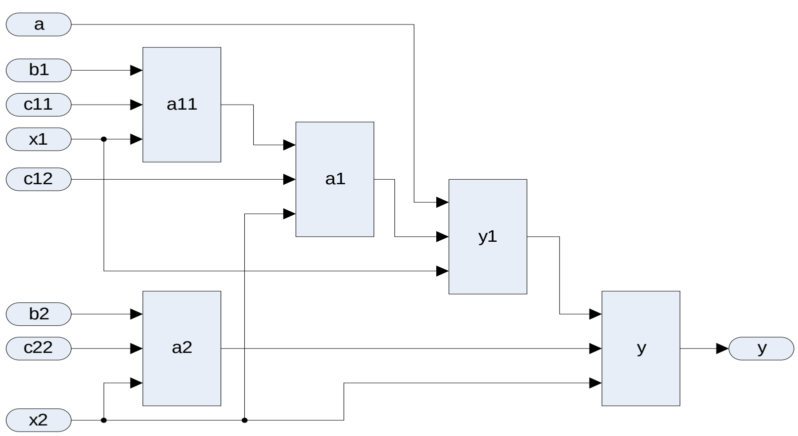
a2x2 = a2 * **Tadc**

Pcomp = y1 + a2x2

This is just a succession of MACs of the form *y = a + b * x:*



The polynomial *Pcomp = a0+(b1 + c11*Padc + c12*Tadc)*Padc + (b2 + c22*Tadc)*Tadc* can be evaluated as a sequence of 5 MACs:



COEFFICIENT BIT-WIDTH SPECS

The table below specifies the initial coefficient bit-width specs for the compensation algorithm.

10-bit Output: Compensation Coefficient Specs							Total Coeff. Bits
	a0	**b1**	**b2**	**c12**	**c11**	**c22**	
Total Bits	16	16	16	14	11	11	84
Sign Bits	1	1	1	1	1	1	
Integer Bits	12	2	1	0	0	0	
Fractional Bits	3	13	14	13	10	10	
dec pt zero pad	-	-	-	9	11	15	

Example Binary Format Definitions:

1. Sign = 0, Integer Bits = 8, Fractional Bits = 4 : Coeff = $S\ I_7\ I_6\ I_5\ I_4\ I_3\ I_2\ I_1\ I_0\ .\ F_3\ F_2\ F_1\ F_0$
2. Sign = 1, Integer Bits = 4, Fractional Bits = 7 : Coeff = $S\ I_3\ I_2\ I_1\ I_0\ .\ F_6\ F_5\ F_4\ F_3\ F_2\ F_1\ F_0$
3. Sign = 0, Integer Bits = 0, Fractional Bits = 6, dec pt zero pad = 2 : Coeff = $S\ 0\ .\ 0\ 0\ F_5\ F_4\ F_3\ F_2\ F_1\ F_0$
4. Sign = 0, Integer Bits = 0, Fractional Bits = 5, dec pt zero pad = 3 : Coeff = $S\ 0\ .\ 0\ 0\ 0\ F_4\ F_3\ F_2\ F_1\ F_0$

NOTE: Negative coefficients (Sign = 1) are coded in 2’s complement notation.

The coefficients derived from mst_trimlib are designed to add no more than 1 LSB of error relative to floating-point calculations. The output bit widths of the intermediate calculations (c11x1, a11, etc.) should be fixed to 16-bits in order to limit the number of operations using an 8-bit MCU, such as the S08. Simulations with Matlab using these fixed-point limits on the intermediate calculations show the pressure specs can be met with Cpks > 2.

COEFFICIENT ADDRESS MAP

Address	Coefficient
$04	a0 MS Byte
$05	a0 LS Byte
$06	b1 MS Byte
$07	b1 LS Byte
$08	b2 MS Byte
$09	b2 LS Byte
$0A	c12 MS Byte
$0B	c12 LS Byte
$0C	c11 MS Byte
$0D	c11 LS Byte
$0E	c22 MS Byte
$0F	c22 LS Byte

For coefficients with less than 16-bits, the lower LSBs are zero. For example, c11 is 11-bits and is stored into 2 bytes as follows:

c11 MS byte = c11[10:3] = [$c11_{b10}$, $c11_{b9}$, $c11_{b8}$, $c11_{b7}$, $c11_{b6}$, $c11_{b5}$, $c11_{b4}$, $c11_{b3}$]

c11 LS byte = c11[2:0] & “00000” = [$c11_{b2}$, $c11_{b1}$, $c11_{b0}$, 0 , 0 , 0 , 0 , 0]

```
//=====================================================
// MPL115A Placing Coefficients into 16 bit variables
//=====================================================
      //coeff a0              16bit
      sia0MSB= I2CCoeff[0];
      sia0LSB= I2CCoeff[1];

      sia0 =  (S16)sia0MSB <<8;       //s16 type //Shift to MSB
      sia0 += (S16)sia0LSB & 0x00FF;  //Add LSB to 16bit number

      //coeff b1    16bit
      sib1MSB= I2CCoeff[2];
      sib1LSB= I2CCoeff[3];

      sib1 =  sib1MSB <<8;       //Shift to MSB
      sib1 += sib1LSB & 0x00FF;  //Add LSB to 16bit number

      //coeff b2    16bit
      sib2MSB= I2CCoeff[4];
      sib2LSB= I2CCoeff[5];

      sib2 =  sib2MSB <<8;       //Shift to MSB
      sib2 += sib2LSB & 0x00FF;  //Add LSB to 16bit number

      //coeff c12   14bit
      sic12MSB= I2CCoeff[6];
      sic12LSB= I2CCoeff[7];

      sic12 =  sic12MSB <<8;  //Shift to MSB only by 8 for MSB
      sic12 += sic12LSB & 0x00FF;

      //coeff c11 11bit

      sic11MSB= I2CCoeff[8];
      sic11LSB= I2CCoeff[9];

      sic11 =  sic11MSB <<8;  //Shift to MSB only by 8 for MSB
      sic11 += sic11LSB & 0x00FF;

      //coeff c22   11bit

      sic22MSB= I2CCoeff[10];
      sic22LSB= I2CCoeff[11];

      sic22 =  sic22MSB <<8; //Shift to MSB only by 8 for MSB
      sic22 += sic22LSB & 0x00FF;
```

```
//==================================================
//Coefficient 9 equation compensation
//==================================================
//
//Variable sizes:
//For placing high and low bytes of the Memory addresses for each of the 6 coefficients:
//signed char (S8) sia0MSB, sia0LSB, sib1MSB,sib1LSB, sib2MSB,sib2LSB, sic12MSB,sic12LSB, sic11MSB,sic11LSB, sic22MSB,sic22LSB;
//
//Variable for use in the compensation, this is the 6 coefficients in 16bit form, MSB+LSB.
//signed int (S16) sia0, sib1, sib2, sic12, sic11, sic22;
//
//Variable used to do large calculation as 3 temp variables in the process below
//signed long (S32) lt1, lt2, lt3;
//
//Variables used for Pressure and Temperature Raw.
//unsigned int (U16) uiPadc, uiTadc.


    //signed (N=number of bits in coefficient, F-fractional bits)
    //s(N,F)

    //The below Pressure and Temp or uiPadc and uiTadc are shifted from the MSB+LSB values to remove the zeros in the LSB since this
    // 10bit number is stored in 16 bits.     i.e 0123456789XXXXXX becomes 0000000123456789

    uiPadc=PressCntdec>>6; //Note that the PressCntdec is the raw value from the MPL115A data address.  Its shifted >>6 since its 10 bit.
    uiTadc=TempCntdec>>6; //Note that the TempCntdec is the raw value from the MPL115A data address.  Its shifted >>6 since its 10 bit.



    //******* STEP 1 c11x1= c11 * Padc
    lt1 = (S32)sic11;            // s(16,27)    s(N,F+zeropad) goes from s(11,10)+11ZeroPad = s(11,22) => Left Justified = s(16,27)
    lt2 = (S32)uiPadc;           // u(10,0)     s(N,F)
    lt3 = lt1 * lt2;             // s(26,27)         /c11*Padc
    si_c11x1 = (S32)(lt3);       // s(26,27)- EQ 1     =c11x1   /checked
                                 //divide this hex number by 2^30 to get the correct decimal value.

                           //b1 =s(14,11) => s(16,13) Left justified
    //******* STEP 2 a11= b1 + c11x1
    lt1 = ((S32)sib1<<14);       // s(30,27)       b1=s(16,13)   Shift b1 so that the F matches c11x1(shift by 14)
    lt2 = (S32)si_c11x1;         // s(26,27)  //ensure fractional bits are compatible
    lt3 = lt1 + lt2;             // s(30,27)         /b1+c11x1
    si_a11 = (S32)(lt3>>14);     // s(16,13) - EQ 2     =a11     Convert this block back to s(16,X)

    //******* STEP 3 c12x2= c12 * Tadc
                                 // sic12 is s(14,13)+9zero pad = s(16,15)+9 => s(16,24) left justified
    lt1 = (S32)sic12;            // s(16,24)
    lt2 = (S32)uiTadc;           // u(10,0)
    lt3 = lt1 * lt2;             // s(26,24)
    si_c12x2 = (S32)(lt3);       // s(26,24) - EQ 3     =c12x2   /checked

    //******* STEP 4 a1= a11 + c12x2
    lt1 = ((S32)si_a11<<11);  // s(27,24) This is done by s(16,13) <<11 goes to s(27,24) to match c12x2's F part
    lt2 = (S32)si_c12x2;         // s(26,24)
    lt3 = lt1 + lt2;             // s(27,24)          /a11+c12x2
    si_a1 =(S32)(lt3>>11);    // s(16,13) - EQ 4     =a1   /check

    //******* STEP 5 c22x2= c22 * Tadc
                                 // c22 is s(11,10)+9zero pad = s(11,19) => s(16,24) left justified
    lt1 = (S32)sic22;            // s(16,30)  This is done by s(11,10) + 15 zero pad goes to s(16,15)+15, to s(16,30)
    lt2 = (S32)uiTadc;           // u(10,0)
    lt3 = lt1 * lt2;             // s(26,30)          /c22*Tadc
    si_c22x2 = (S32)(lt3);       // s(26,30) - EQ 5     /=c22x2

    //******* STEP 6 a2= b2 + c22x2
    //WORKS and loses the least in data. One extra execution.  Note how the 31 is really a 32 due to possible overflow.

                               // b2 is s(16,14) User shifted left to  => s(31,29) to match c22x2 F value
    lt1 = ((S32)sib2<<15);   //s(31,29)
```

```
lt2 = ((S32)si_c22x2>>1);     //s(25,29)  s(26,30) goes to >>16 s(10,14) to match F from sib2
lt3 = lt1+lt2;                //s(32,29) but really is a s(31,29) due to overflow the 31 becomes a 32.
si_a2 = ((S32)lt3>>16);       //s(16,13)

//******* STEP 7  a1x1= a1 * Padc

lt1 = (S32)si_a1;             // s(16,13)
lt2 = (S32)uiPadc;            // u(10,0)
lt3 = lt1 * lt2;              // s(26,13)          /a1*Padc
si_a1x1 = (S32)(lt3);         // s(26,13)  - EQ 7    /=a1x1  /check

//******* STEP 8  y1= a0 + a1x1
                              // a0 = s(16,3)
lt1 = ((S32)sia0<<10);        // s(26,13)  This is done since has to match a1x1 F value to add.  So S(16,3) <<10 = S(26,13)
lt2 = (S32)si_a1x1;           // s(26,13)
lt3 = lt1 + lt2;              // s(26,13)          /a0+a1x1
si_y1 = ((S32)lt3>>10);       // s(16,3)  - EQ 8     /=y1    /check

//******* STEP 9  a2x2= a2 *Tadc

lt1 = (S32)si_a2;             // s(16,13)
lt2 = (S32)uiTadc;            // u(10,0)
lt3 = lt1 * lt2;              // s(26,13)          /a2*Tadc
si_a2x2 = (S32)(lt3);         // s(26,13)  - EQ 9     /=a2x2

//******* STEP 10 pComp = y1 +a2x2
                              // y1= s(16,3)
lt1 = ((S32)si_y1<<10);       // s(26,13)  This is done to match a2x2 F value so addition can match.  s(16,3) <<10
lt2 = (S32)si_a2x2;           // s(26,13)
lt3 = lt1 + lt2;              // s(26,13)          /y1+a2x2


// FIXED POINT RESULT WITH ROUNDING:
siPcomp = ((S16)lt3>>13);     // goes to no fractional parts since this is an ADC count.

//decPcomp is defined as  a floating point number.
//Conversion to Decimal value from 1023 ADC count value.  ADC counts are 0 to 1023.  Pressure is 50 to 115kPa correspondingly.
decPcomp = ((65.0/1023.0)*siPcomp)+50;
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3785
Rev. 5
7/2009