Document Name: AN3785
Rev 6, 6/2012

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^2C or SPI bus. The part's operating voltage is from 2.375 V to 5.5 V. The part is available as either the MPL115A1 (SPI) or MPL115A2 (I^2C) 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^2C 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.

Updates to this Application Note:

This application note has been updated to reflect changes in the pressure compensation algorithm - the 2nd order coefficients have now been removed. This modifies the compensated Pressure equation to be simpler than the previous version. Both are detailed so a comparsion can be made. Particular details have also migrated to the datasheet for MPL115A1 and MPL115A2 to expand on content in those documents.

Sample Boards:

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

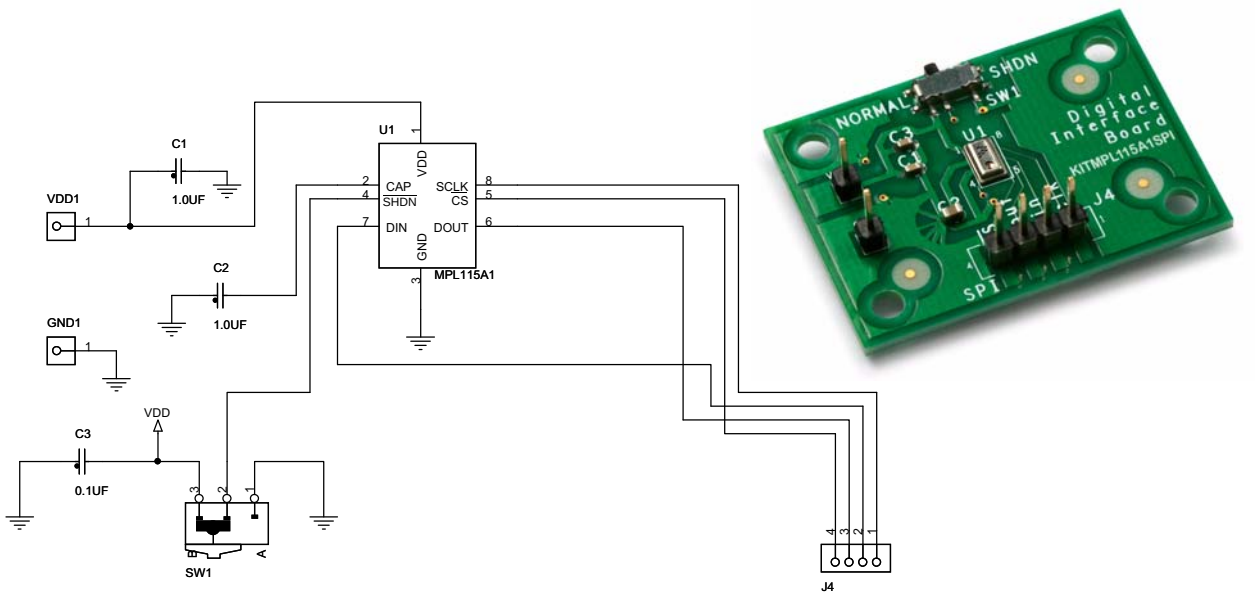


Figure 1. KITMPL115A1SPI: MPL115A1 (SPI) Interface Board

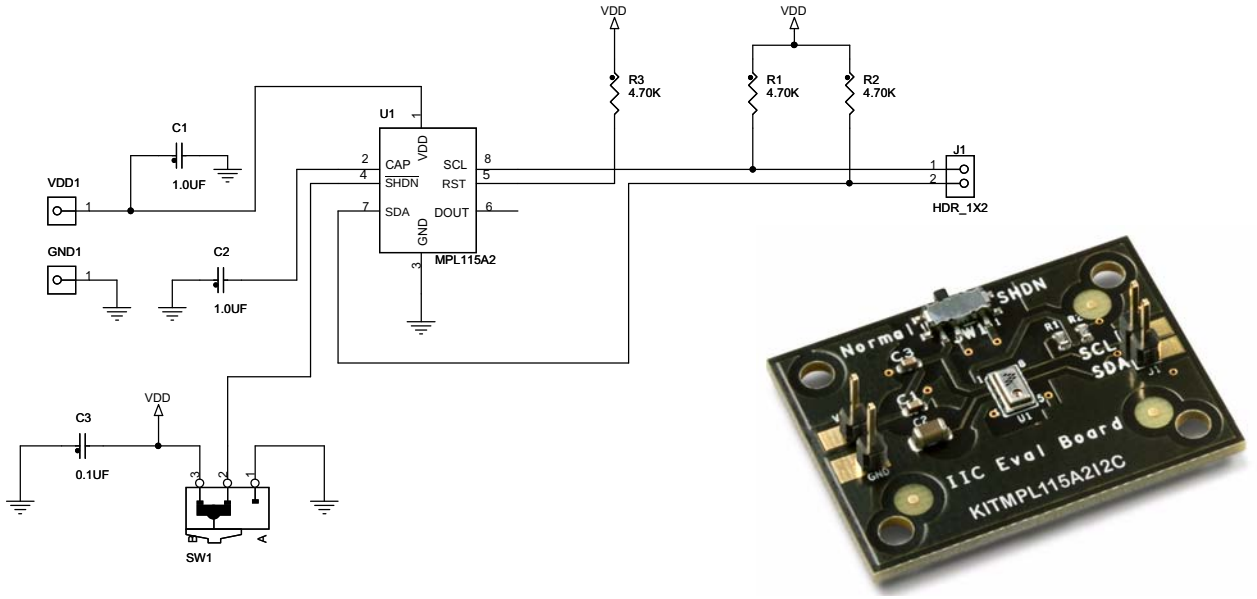


Figure 2. KITMPL115A2I2C: MPL115A2 (I^2C) Interface Board

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

1 Mechanical and Electrical Specifications

1.1 Maximum Ratings

Voltage (with respect to GND unless otherwise noted)

V_{DD} ..... -0.3 V to +5.5 V

$\overline{SHDN}$, 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

Overpressure ..... 1000 kPa

1.2 Operating 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 ($\overline{SHDN}$ = GND)	—	—	1	µA
			Standby	—	3.5	10	µA
			Average – at one measurement per second	—	5	6	µA
Pressure Sensor							
3	Range			50	—	115	kPa
4	Resolution			—	0.15	—	kPa
5	Accuracy		-20ºC to 85ºC	—	—	±1	kPa
6	Power Supply Rejection		Typical operating circuit at DC		0.1	—	kPa/V
			100 mV p-p 217 Hz square wave plus 100 mV pseudo random noise with 10 MHz bandwidth		0.1	—	kPa
7	Conversion Time *(Start Pressure and Temperature Conversion)*	tc	Time between start convert command and data available in the Pressure and Temperature registers	—	1.6	3	ms
8	Wakeup Time	tw	Time between leaving Shutdown mode ($\overline{SHDN}$ goes high) and communicating with the device to issue a command or read data.	—	3	5	ms
I^2C I/O Stages: SCL, SDA							
9	SCL Clock Frequency	f_{SCL}		—	—	400	kHz
10	Low Level Input Voltage	VIL		—	—	$0.3V_{DD}$	V
11	High Level Input Voltage	VIH		$0.7V_{DD}$	—	—	V
I^2C Outputs: SDA							
12	Data Setup Time	t_{SU}	Setup time from command receipt to ready to transmit	0	—	0.4	s
I^2C Addressing							
MPL115A2 uses 7-bit addressing, does not acknowledge the general call address 0000000. Slave address has been set to 0x60 or 1100000.							

1.3 Pin Connections SPI

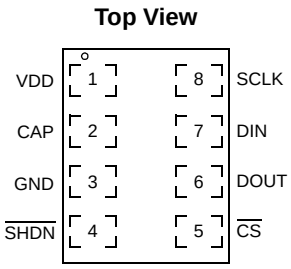


Figure 3. Pin Connections SPI

1.4 Pin Connections I²C

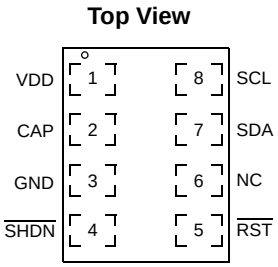


Figure 4. Pin Connections I²C

2 Overview of Functions/Operation

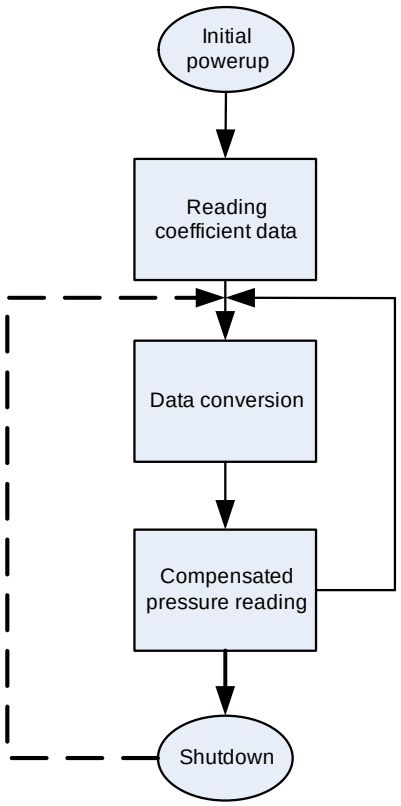


Figure 5. Sequence Flow Chart

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

Initial Power-up

All circuit elements are active. SPI or I^2C port pins are high impedance and associated registers are cleared. The device then enters standby mode.

Reading Coefficient Data

The user then typically accesses the part and reads the coefficient data. The main circuits within the slave device are disabled during read activity. The coefficients are usually stored in the host microcontoller local memory but can be re-read at any time.

It is suggested to read the coefficients once and store the values in the host microcontroller for speed. It is not necessary to read this multiple times because the coefficients within a device are constant and do not change. However, note that the coefficients will be different from device to device, and cannot be used for another part.

Data Conversion

This is the first step that is performed each time a new pressure reading is required which is initiated by the host sending the CONVERT command. The main system circuits are activated (wake) in response to the command and after the conversion completes, the result is placed into the Pressure and Temperature ADC output registers.

The conversion completes within the maximum conversion time, tc of 1.6ms typical, or 3ms max. The device then enters standby mode.

Compensated Pressure Reading

After the conversion has been given sufficient time to complete, the host microcontroller reads the result from the ADC output registers and calculates the Compensated Pressure, a barometric/atmospheric pressure value which is compensated for changes in temperature and pressure sensor linearity. This is done using the coefficient data from the MPL115A and the raw sampled pressure and temperature ADC output values, in a compensation equation (detailed later). Note that this is an absolute pressure measurement with a vacuum as a reference.

From this step the host controller may either wait and then return to the Data Conversion step to obtain the next pressure reading or it may go to the Shutdown step with the full supply voltage range.

I²C Communication

The I²C protocol is explained below along with how to implement this pressure sensor. For additional information on I²C, please see AN4481 Sensors I²C Setup and FAQ. I²C is compliant to the Philips I²C Bus specification, version 2.1.

The MPL115A2 uses a 7 bit device address such that in an 8 bit representation, the device address is followed by a least significant bit that toggles a I²C read/write value. This leads to a I²C MPL115A2 address of 0xC0 for a write command, and 0xC1 for a read command.

MPL115A2 Commands	Address
Write Command	0xC0
Read Command	0xC1

Figure 6. MPL115A2 Commands

Write Mode

I²C writes to the device are conducted using normal I²C protocol. An I²C start condition is followed by the 7-bit slave address and *Write* bit (0). 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 include CONVERT to Start Pressure and Temperature conversion. These commands are shown in Table 2. An example of the user mode I²C write sequence is shown in Figure .

	7 bit slave address plus WRITE bit									8 bit command									Stop bit (or repeated start)
S	D	D	D	D	D	D	D	W	Ack	0	X	X	X	X	X	X	X	Ack	P
Start									Acknowledge										End of Transmission

Figure 7. I²C Write Sequence – User Mode

Conversion Time

Note that when sending a write command to Start Conversion, there is a conversion time of 1.6ms typical for the part to internally sample and output results. After this conversion time, the data will be ready to be read. Before entering a repeated start or new start, ensure this delay is inserted before reading new data.

Read Mode

I²C reads from the device are conducted using normal I²C protocol.

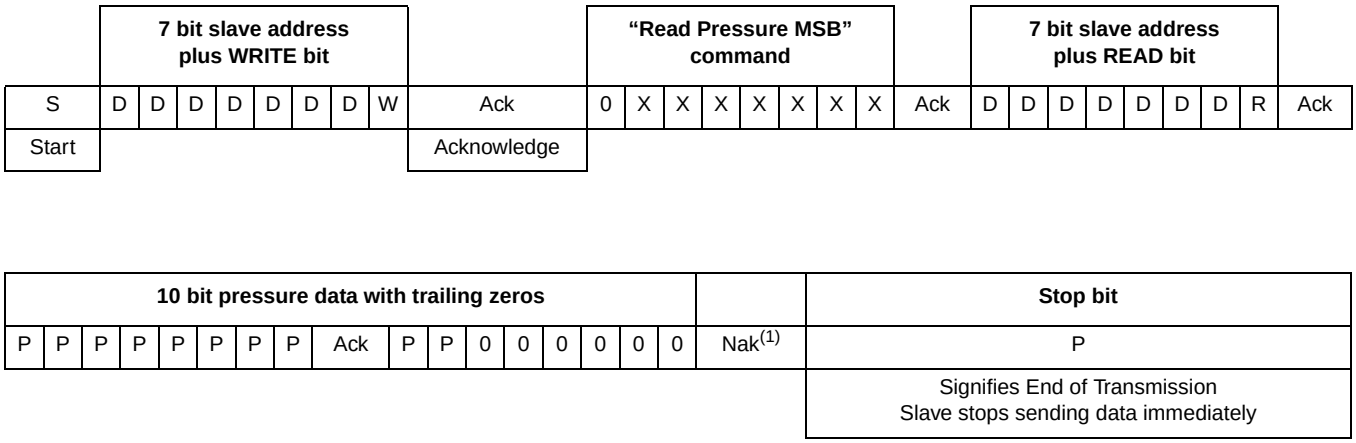
To read from the device the master must first send a data address or command that points to the required data. To do this the master sends the 7-bit slave address + *Write* bit followed by data address byte. Transmission is initiated by an I²C start condition and ended by either a *Repeated Start* or 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 I²C 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 I²C reads for pressure data (2 bytes) and coefficient data (up to 12 bytes) are shown in Figure 7 and Figure 9.

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.

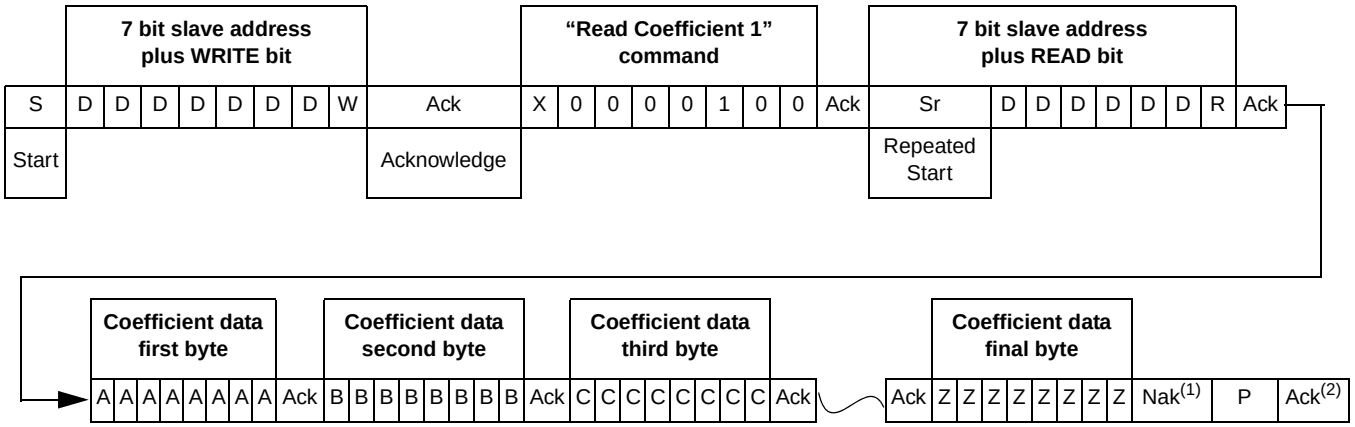
I²C Read Pressure Sequence



1. The standard termination sequence is "Nak" followed by the stop condition. An "Ack" followed by the stop condition may also be used.

Figure 8. I²C Read Pressure Sequence

I²C Read Coefficients Sequence



1. The standard termination sequence is "Nak" followed by the stop condition. An "Ack" followed by the stop condition may also be used.
2. Stop bit signifies end of transmission. Slave stops sending data immediately.

Figure 9. I²C Read Coefficients Sequence

I²C RST

The I²C RST function that can be seen in the pin connection, 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 is the "action" address "*Start Conversion."* The interface is half duplex and so cannot receive and transmit simultaneously. Minimum data setup time (interval between Start Conversions receipt and data ready on DOUT) is typically 1.6 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

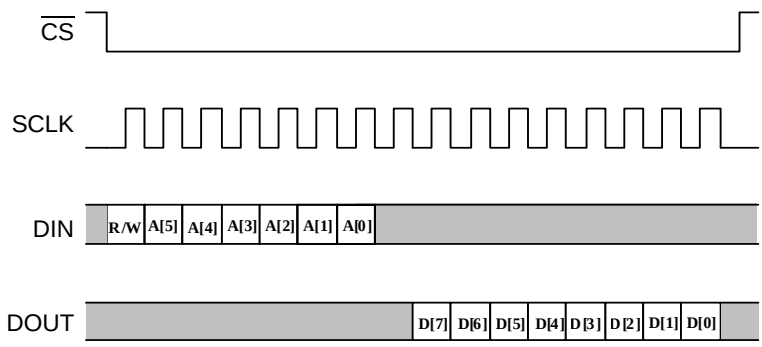


Figure 10. SPI Read Operation

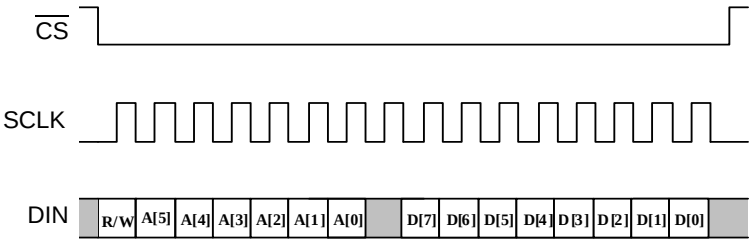


Figure 11. SPI Write Operation

DEVICE READ/WRITE OPERATIONS

All device read/write operations are memory mapped. Device actions "Start Conversions" are controlled by writing to the appropriate memory address location. All memory address locations are 6-bit (see Table 1). 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^{2}C 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^{2}C 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. I^{2}C Write Commands

Command	Binary	HEX(1)	Command
Devices Address + Write bit	11000000	0xC0	Devices Address + Write bit
Start Conversions	X0010010	0x12	Start Conversions

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of Start Conversions.

Table 2. I^{2}C Write Commands description

Command	Action Taken
Start Conversions	Wake 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 Pressure and Temperature registers with the result. Shut down main circuits and clock. This action takes a typical time of 1.6ms to complete and have data available in the Pressure and Temperature registers.

Table 3. I^{2}C Read Command Description

Command	Binary	HEX(1)
Device Address + Read bit	11000001	0xC1
Read Pressure MSB	X0000000	0x00
Read Pressure LSB	X0000001	0x01
Read Temperature MSB	X0000010	0x02
Read Temperature LSB	X0000011	0x03
Read Coefficient data byte 1	X0000100	0x04

X = don’t care

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of Start Conversions.

Table 4. SPI Write Commands

Command	Binary	HEX(1)
Start Conversions	0010010X	0x24

X = don’t care

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of Start Conversions.

Table 5. SPI Write Command Description

Command	Action Taken
Start Conversions	Wake 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 Pressure and Temperature registers with the result. Shut down main circuits and clock. This action takes a typical time of 1.6ms to complete and have data available in the Pressure and Temperature registers.

Table 6. Example SPI Read Commands

Command	Binary	HEX[1]
Read Pressure MSB	1000000X	0x80
Read Pressure LSB	1000001X	0x82
Read Temperature MSB	1000010X	0x84
Read Temperature LSB	1000011X	0x86
Read Coefficient data byte 1	1000100X	0x88

X = don't care

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of stated command.

DEVICE MEMORY MAP

Table 1. Device Memory Map

Address	Name	Description	Size (bits)
0x00	Padc_MSB	10-bit Pressure output value MSB	8
0x01	Padc_LSB	10-bit Pressure output value LSB	2
0x02	Tadc_MSB	10-bit Temperature output value MSB	8
0x03	Tadc_LSB	10-bit Temperature output value LSB	2
0x04	a0MSB	a0 coefficient MSB	8
0x05	a0LSB	a0 coefficient LSB	8
0x06	b1MSB	b1 coefficient MSB	8
0x07	b1LSB	b1 coefficient LSB	8
0x08	b2MSB	b2 coefficient MSB	8
0x09	b2LSB	b2 coefficient LSB	8
0x0A	c12MSB	c12 coefficient MSB	8
0x0B	c12LSB	c12 coefficient LSB	8
0x0C	Reserved*	—	—
0x0D	Reserved*	—	—
0x0E	Reserved*	—	—
0x0F	Reserved*	—	—
0x10	Reserved	—	—
0x11	Reserved	—	—
0x12	CONVERT	Start both Pressure and Temperature Conversion	—

*These registers are set to 0x00. These are reserved, and were previously utilized as Coefficient values, now set to a value to 0x00 for simplifying compensation.

2.1 Pressure, Temperature and Coefficient Bit-Width Specifications

The table below specifies the initial coefficient bit-width specifications for the compensation algorithm and the specifications for Pressure and Temperature ADC values.

Pressure, Temperature and Compensation Coefficient Specifications						
	a0	b1	b2	c12	Padc	Tadc
Total Bits	16	16	16	14	10	10
Sign Bits	1	1	1	1	0	0
Integer Bits	12	2	1	0	10	10
Fractional Bits	3	13	14	13	0	0
dec pt zero pad	0	0	0	9	0	0

Example Binary Format Definitions:

a0 Signed, Integer Bits = 12, Fractional Bits = 3 :	Coeff a0 = S I_{11} I_{10} I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0 . F_2 F_1 F_0
b1 Signed, Integer Bits = 2, Fractional Bits = 7 :	Coeff b1 = S I_1 I_0 . F_{12} F_{10} F_9 F_8 F_7 F_6 F_5 F_4 F_3 F_2 F_1 F_0
b2 Signed, Integer Bits = 1, Fractional Bits = 14 :	Coeff b2 = S I_0 . F_{13} F_{12} F_{10} F_9 F_8 F_7 F_6 F_5 F_4 F_3 F_2 F_1 F_0
c12 Signed, Integer Bits = 0, Fractional Bits = 13, dec pt zero pad = 9 :	Coeff c12 = S 0 . 000 000 000 F_{12} F_{10} F_9 F_8 F_7 F_6 F_5 F_4 F_3 F_2 F_1 F_0
Padc Unsigned, Integer Bits = 10 :	Padc U = I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0
Tadc Unsigned, Integer Bits =10 :	Tadc U = I_9 I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0

NOTE: Negative coefficients are coded in 2's complement notation.

2.2 Compensation

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

$$Pcomp = a0 + (b1 + c12 \cdot Tadc) \cdot Padc + b2 \cdot Tadc \qquad \textit{Eqn. 1}$$

Where:

Padc is the 10-bit pressure ADC output of the MPL115A

Tadc is the 10-bit temperature ADC output of the MPL115A

a0 is the pressure offset coefficient

b1 is the pressure sensitivity coefficient

b2 is the 1st order temperature offset coefficient (TCO)

c12 is the coefficient for temperature sensitivity coefficient (TCS)

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.

$$\text{Pressure (kPa)} = Pcomp \cdot \left[\frac{115 - 50}{1023}\right] + 50 \qquad \textit{Eqn. 2}$$

2.3 Evaluation Sequence, Arithmetic Circuits

The following is an example of the calculation for Pcomp, the compensated pressure output. Input values are in **bold.**

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

a1 = **b1** + c12x2

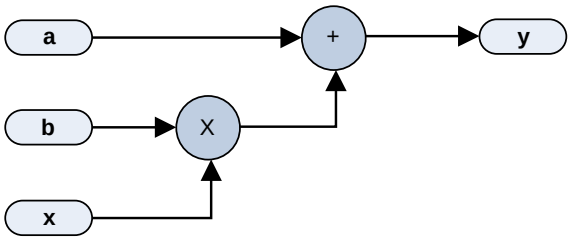
a1x1 = a1 * **Padc**

y1 = **a0** + a1x1

a2x2 = **b2** * **Tadc**

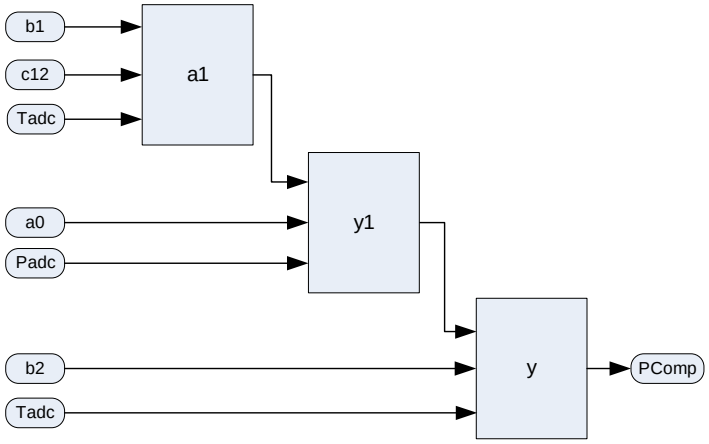
Pcomp = y1 + a2x2

This can be calculated as a succession of Multiply Accumulates (MACs) operations of the form *y = a + b * x:*



The polynomial can be evaluated (Equation 1) as a sequence of 3 MACs:

$$Pcomp = a0 + (b1 + c12 \cdot Tadc) \cdot Padc + b2 \cdot Tadc$$



I²C COMMUNICATION AND EXAMPLE

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

Table 2. I²C Write Command Description

Command	Action Taken
Start Conversions	Wake 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 Pressure and Temperature registers with the result. Shut down main circuits and clock.

Table 3. I²C Read Command Description

Command	Binary	HEX(1)
Device Address + Read bit	1100 0001	0xC1
Read Pressure MSB	X000 0000	0x00
Read Pressure LSB	X000 0001	0x01
Read Temperature MSB	X000 0010	0x02
Read Temperature LSB	X000 0011	0x03
Read Coefficient data byte 1	X000 0100	0x04

X = don't care

These are MPL115A2 I²C commands to read coefficients, execute Pressure and Temperature conversions, and to read Pressure and Temperature data. The sequence of the commands for the interaction is given as an example to operate the MPL115A2.

Utilizing this gathered data, an example of the calculating the Compensated Pressure reading is given in floating point notation.

I²C Commands (simplified for communication)

Device Address + write bit "To Write" = 0xC0

Device Address + read bit "To Read" = 0xC1

Command to Write "Convert Pressure and Temperature" = 0x12

Command to Read "Pressure ADC High byte" = 0x00

Command to Read “Pressure ADC Low byte” = 0x01

Command to Read “Temperature ADC High byte” = 0x02

Command to Read “Temperature ADC Low byte” = 0x03

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

Read Coefficients:

[0xC0], [0x04], [0xC1], [0x3E], [0xCE], [0xB3], [0xF9], [0xC5], [0x17], [0x33], [0xC8]

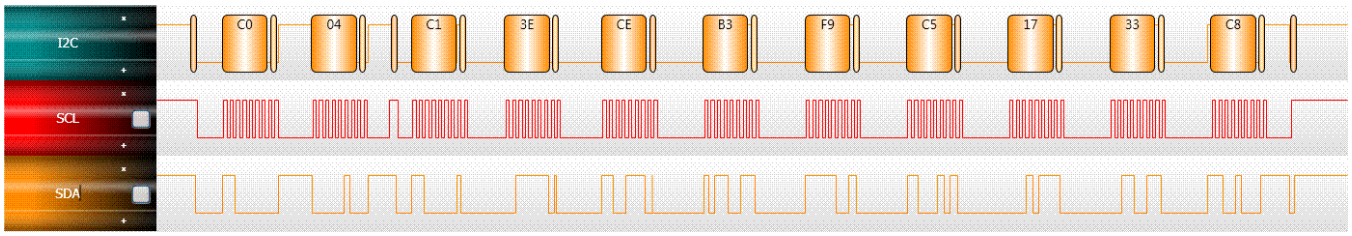


Figure 12. I²C Read Coefficient Datagram

a0 coefficient MSB = 0x3E

a0 coefficient LSB = 0xCE a0 coefficient = 0x3ECE = 2009.75

b1 coefficient MSB = 0xB3

b1 coefficient LSB = 0xF9 b1 coefficient = 0xB3F9 = -2.37585

b2 coefficient MSB = 0xC5

b2 coefficient LSB = 0x17 b2 coefficient = 0xC517 = -0.92047

c12 coefficient MSB = 0x33

c12 coefficient LSB = 0xC8 c12 coefficient = 0x33C8 = 0.000790

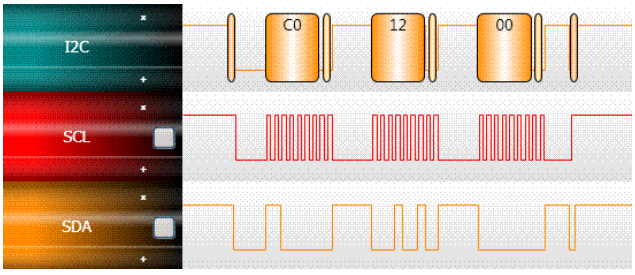


Figure 13. I²C Start Conversion Datagram

Command to I²C Start Conversion, 0x12

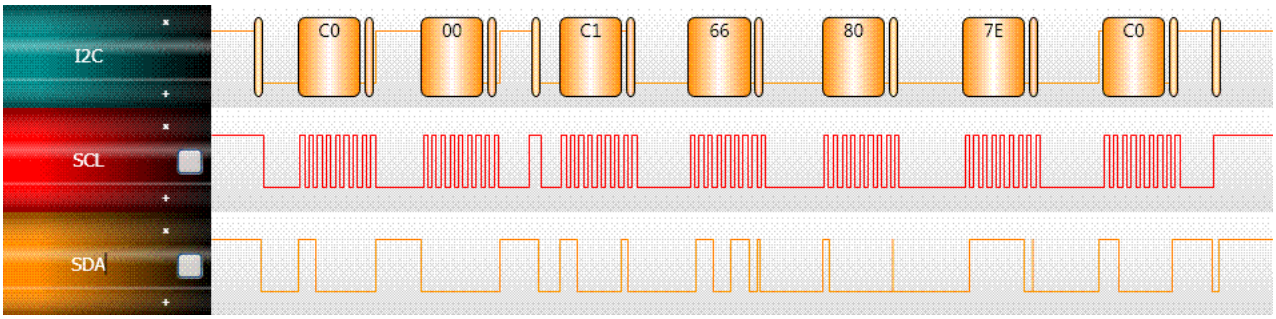


Figure 14. I²C Read Results Datagram

Pressure MSB	=	0x66				
Pressure LSB	=	0x80	Pressure	=	0x6680	= 0110 0110 1100 0000
						= 410 ADC counts
Temperature MSB	=	0x7E				
Temperature LSB	=	0xC0	Temperature	=	0x7EC0	= 0111 1110 1100 0000
						= 507 ADC counts

2.4 Example of Pressure Compensated Calculation in Floating-point Notation

a0 coefficient	=	2009.75
b1 coefficient	=	-2.37585
b2 coefficient	=	-0.92047
c12 coefficient	=	0.000790

Pressure	=	410 ADC counts
Temperature	=	507 ADC counts

Pressure Compensation:

$$Pcomp = a0 + (b1 + c12 \cdot Tadc) \cdot Padc + b2 \cdot Tadc$$

Using the evaluation sequence shown in Section 3.3:

c12x2	=	c12 * Tadc	=	0.000790 * 507	=	0.40053
a1	=	b1 + c12x2	=	-2.37585 + 0.40053	=	-1.97532
a1x1	=	a1 * Padc	=	-1.97532 * 410	=	-809.8812
y1	=	a0 + a1x1	=	2009.75 + (-809.8812)	=	1199.8688
a2x2	=	b2 * Tadc	=	-0.92047 * 507	=	-466.67829
PComp	=	y1 + a2x2	=	1199.8688 + (-466.67829)	=	733.19051

$$\text{Pressure (kPa)} = \text{Pcomp.}\left[\frac{115-50}{1023}\right] + 50$$

$$= 733.19.\left[\frac{115-50}{1023}\right] + 50$$

$$= 96.59\text{kPa}$$

2.5 SPI 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 1).

The 8-bit command word comprises:

- the most significant bit which is the Read/$\overline{\text{Write}}$ identifier which is '1' for read operations and '0' for writes.
- the 6-bit address (from Table 1);
- the least significant bit which is not used and is don't care (X).

The device write commands are shown in Table 1.

Table 4. SPI Write Command

Command	Binary	HEX(1)
Start Conversions	0010010X	0x24

X = don’t care

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of *Start Conversions*.

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

Table 5. SPI Write Command Description

Command	Action Taken
Start Conversions	Wake 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 Pressure and Temperature registers with the result. Shut down main circuits and clock.

SPI Read operations are performed by sending the required address with a leading *Read* bit set to '1'. SPI operations require that each byte be addressed individually. All data is transmitted most significant bit first.

Table 6. Example SPI Read Commands

Command	Binary	HEX(1)
Read Pressure MSB	1000000X	0x80
Read Pressure LSB	1000001X	0x82
Read Temperature MSB	1000010X	0x84
Read Temperature LSB	1000011X	0x86
Read Coefficient data byte 1	1000100X	0x88

X = don't care

1. The command byte needs to be paired with a 0x00 as part of the SPI exchange to complete the passing of stated command.

2.6 SPI Timing

Table 7 and Figure describe the timing requirements for the SPI system.

Table 7. SPI Timing

Ref	Function	Symbol	Min	Max	Unit
1	Operating Frequency	Of	—	8	MHz
2	SCLK Period	tSCLK	125	—	ns
3	SCLK High time	tCLKH	62.5	—	ns
4	SCLK Low time	tCLKL	62.5	—	ns
5	Enable lead time	tSCS	125	—	ns
6	Enable lag time	tHCS	125	—	ns
7	Data setup time	tSET	30	—	ns
8	Data hold time	tHOLD	30	—	ns
9	Data valid (after SCLK low edge)	tDDLY	—	32	ns
10	Width CS High	tWCS	30	—	ns

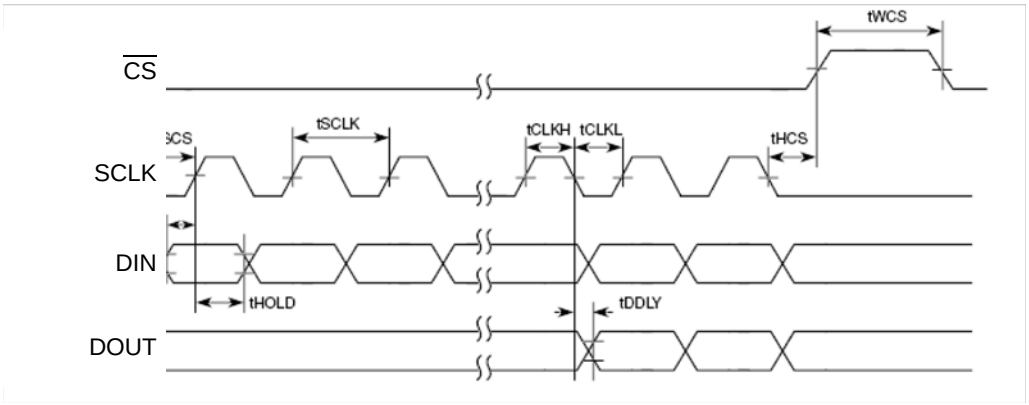


Figure 15. SPI Timing Diagram

2.7 Example of SPI Reading of Coefficients

These are MPL115A2 SPI commands to read coefficients, execute Pressure and Temperature conversions, and to read Pressure and Temperature data. The sequence of the commands for the interaction is given as an example to operate the MPL115A2. Utilizing this gathered data, an example of the calculating the Compensated Pressure reading is given in floating point notation.

SPI Commands (simplified for communication)

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

Command to Read “Pressure ADC High byte” = 0x80

Command to Read “Pressure ADC Low byte” = 0x82

Command to Read “Temperature ADC High byte” = 0x84

Command to Read “Temperature ADC High byte” = 0x86

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

Read Coefficients:

[CS=0], [0x88], [0x00], [0x8A], [0x00], [0x8C], [0x00], [0x8E], [0x00], [0x90], [0x00], [0x92], [0x00], [0x94], [0x00], [0x96], [0x00], [0x00], [CS=1]

Start Pressure and Temperature Conversion, Read raw Pressure:

[CS=0], [0x24], [0x00], [CS=1], [13 ms Delay]
[CS=0], [0x80], [0x00], [0x82], [0x00], [0x84], [0x00,] [0x86], [0x00], [0x00], [CS=1]

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

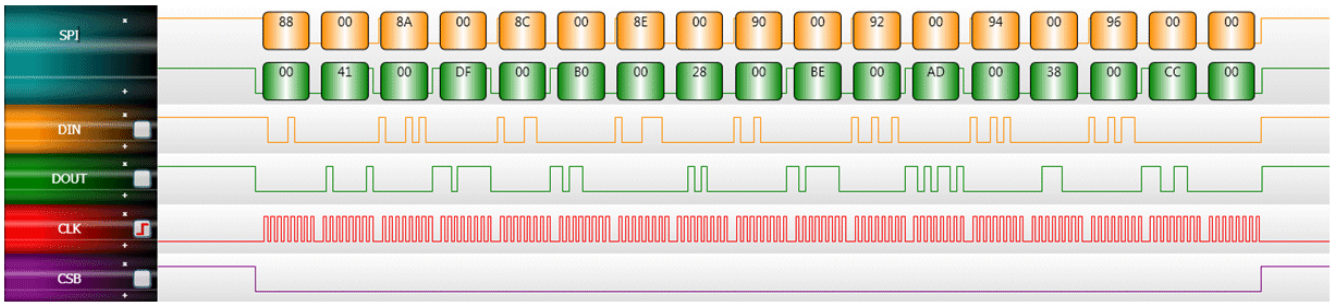


Figure 16. SPI Read Coefficient Datagram

a0 coefficient MSB = 0x41
a0 coefficient LSB = 0xDF a0 coefficient = 0x41DF = 2107.875

b1 coefficient MSB = 0xB0
b1 coefficient LSB = 0x28 b1 coefficient = 0xB028 = -2.495117188

b2 coefficient MSB = 0xBE
b2 coefficient LSB = 0xAD b2 coefficient = 0xBEAD = -1.02069

c12 coefficient MSB = 0x38
c12 coefficient LSB = 0xCC c12 coefficient = 0x38CC = 0.000867

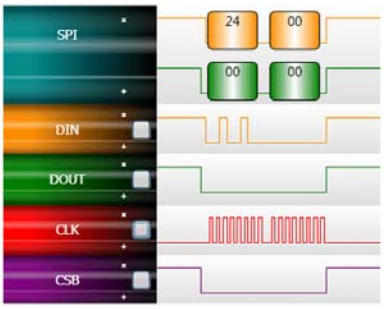


Figure 17. SPI Start Conversion Datagram

Command to Start Pressure and Temperature Conversion, 0x24

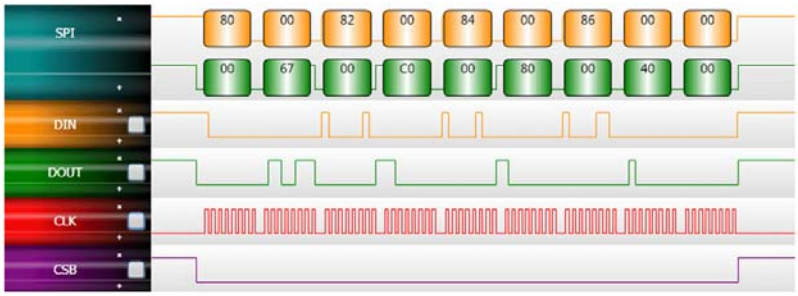


Figure 18. SPI Read Results Datagram

Pressure MSB = 0x67

Pressure LSB = 0xC0 Pressure = 0x67C0 = 0110 0111 11 00 0000

= 415 ADC counts

Temperature MSB = 0x80

Temperature LSB = 0x40 Temperature = 0x8040 = 1000 0000 01 00 0000

= 513 ADC counts

2.8 Example of Pressure Compensated Calculation in Floating-point Notation

a0 coefficient = 2107.875

b1 coefficient = -2.495117188

b2 coefficient = -1.02069

c12 coefficient = 0.000867

Pressure = 415 ADC counts

Temperature = 513 ADC counts

Pressure Compensation:

$$Pcomp = a0 + (b1 + c12 \cdot Tadc) \cdot Padc + b2 \cdot Tadc$$

Using the evaluation sequence shown in Section 3.3:

c12x2	=	c12 * Tadc	=	0.00086665 * 513	=	0.44459
a1	=	b1 + c12x2	=	-2.49512 + 0.44459	=	-2.05052
a1x1	=	a1 * Padc	=	-2.05052 * 415	=	-850.96785
y1	=	a0 + a1x1	=	2107.875 + (-850.96785)	=	1256.90715
a2x2	=	b2 * Tadc	=	-1.02069 * 513	=	-523.61444
PComp	=	y1 + a2x2	=	1256.90715 + (-523.61444)	=	733.29270

$$\text{Pressure (kPa)} = \text{Pcomp} \cdot \left[\frac{115 - 50}{1023}\right] + 50$$

$$= 733.29 \cdot \left[\frac{115 - 50}{1023}\right] + 50$$

$$= 96.59\text{kPa}$$

Coding Section

```
/*************************************************************\
* Calculate the compensated pressure PComp from the last set
* of ADC and coefficient values.
\*************************************************************/
sint16 mpl115a1_CalculatePComp(void)
{
 uint16 Padc, Tadc;
 sint16 a0, b1, b2, c12;
 sint16 PComp;

 // extract adc outputs
 Padc = (mpl115a1_regs[0x00] << 8) | mpl115a1_regs[0x01];
 Tadc = (mpl115a1_regs[0x02] << 8) | mpl115a1_regs[0x03];

 // extract coefficients
 a0  = (mpl115a1_regs[0x04] << 8) | mpl115a1_regs[0x05];
 b1  = (mpl115a1_regs[0x06] << 8) | mpl115a1_regs[0x07];
 b2  = (mpl115a1_regs[0x08] << 8) | mpl115a1_regs[0x09];
 c12 = (mpl115a1_regs[0x0A] << 8) | mpl115a1_regs[0x0B];

 // calculate internally compenstated PComp value using either version
 //PComp = calculatePCompLong(Padc, Tadc, a0, b1, b2, c12);
 PComp = calculatePCompShort(Padc, Tadc, a0, b1, b2, c12);

 return (sint16)PComp;
}


/*************************************************************\
* Calculate the pressure in 1/16 kPa from a compensated
* PComp value.
\*************************************************************/
uint16 mpl115a1_CalculatePressure(sint16 PComp)
{
 sint32 Pressure;

 // The final step is to convert the internal PComp value into units of kPa.
 //   Pressure = PComp · ((115.0 - 50.0) / 1023.0) + 50
 //
 // The use of a floating point divide can be eliminated using the following approximation:
 //   Pressure = ( ( PComp · 1041 ) >> 14 ) + 50
 //
```

```
 // Note that in this implementation the final pressure value is reported with a 4 bit fractional
 // part. This may be eliminated by right shifting the result four additional bits.

 Pressure = ((((sint32)PComp) * 1041) >> 14) + 800;

 return (uint16)Pressure;
}


/*********************************************************\
* Calculate the compensated pressure PComp value using the detailed description.
\*********************************************************/
sint16 calculatePCompLong(uint16 Padc, uint16 Tadc, sint16 a0, sint16 b1, sint16 b2, sint16 c12)
{
 // TEMPORARY DATA VARIABLES:
 sint32 lt1, lt2, lt3;
 sint32 c12x2, a1, a1x1, y1, a2x2, PComp;

 // Pressure calculation (long)
 //============================
 // This version of the pressure calculation function has the long description showing exaclty
 // how the bit widths of the coefficients align through the calculation.
 //
 // Variables used to do large calculation as 3 temp variables in the process below
 //  signed long (sint32) lt1, lt2, lt3;
 //
 // Variables used for Pressure and Temperature Raw.
 //  unsigned short (uint16) Padc, Tadc.
 //
 // In order to optimize the fixed point arithmetic, each value is annotated with a descriptor x(N,F).
 //  x is 's' for signed or 'u' for unsigned
 //  N is the number of significant digits in the value
 //  F is the number of fractional bits, right of the decimal point
 //
 // Each of the input values and coefficients are identified below, based upon the coefficient bit
 // width table in the data sheet:
 //   Padc  : u(10.0)
 //   Tadc  : u(10,0)
 //   a0    : s(16,3)
 //   b1    : s(16,13)
 //   b2    : s(16,14)
 //   c12   : s(16,24) // s(14,13) + 9 zero pad = s(16,15+9) => s(16,24) left justified
 //   PComp : s(16,4) // compensated pressure value contains a 8 bit integer part and a 4 bit fractional part
 //
 // The compensation formula is:
 //   PComp = a0 + (b1 + c12 · Tadc) · Padc + b2 · Tadc

 // The calculation can be broken down into individual steps

 Padc>>=6; //Note that the Padc is the raw value from Pegasus >>6 since its 10 bit unsigned
 Tadc>>=6;  //Note that the Tadc is the raw value from Pegasus >>6 since its 10 bit unsigned
 //******* STEP 1 : c12x2 = c12 * Tadc
 lt1 = c12;         // s(16,24) // c12 is s(14,13)+9 zero pad = s(16,15)+9 => s(16,24) left justified
 lt2 = (sint16)Tadc; // u(10,0)
 lt3 = lt1 * lt2;    // s(26,24) = c12 * Tadc
 c12x2 = lt3 >> 11;  // s(15,13) - EQ 3 = c12x2
 //******* STEP 2 : a1 = b1 + c12x2
 lt1 = (sint16)b1;   // s(16,13)
 lt2 = c12x2;        // s(15,13)
 lt3 = lt1 + lt2;    // s(16,13) = b1 + c12x2
 a1  = lt3;          // s(16,13) - EQ 4 = a1
 //******* STEP 3 : a1x1 = a1 * Padc
 lt1 = a1;          // s(16,13)
 lt2 = (sint16)Padc; // u(10,0)
 lt3 = lt1 * lt2;    // s(26,13) = a1 * Padc
```

```
  a1x1 = lt3;       // s(26,13) - EQ 5 = a1x1
  //******* STEP 4   y1 = a0 + a1x1
  lt1 = ((sint32)a0) << 10; // s(26,13) shifted to match a1x1 F value to add. So s(16,3)<<10 = s(26,13)
  lt2 = a1x1;       // s(26,13)
  lt3 = lt1 + lt2;  // s(26,13) = a0 + a1x1
  y1  = lt3;        // s(26,13) - EQ 6 = y1
  //******* STEP 5 : a2x2 = b2 * Tadc
  lt1 = (sint32)b2;  // s(16,14)
  lt2 = (sint32)Tadc; // u(10,0)
  lt3 = lt1 * lt2;  // s(26,14) = b2 * Tadc
  a2x2 = lt3 >> 1;  // s(25,13) - EQ 7 = a2x2
  //******* STEP 6 : PComp = y1 + a2x2
  lt1 = y1;         // s(26,13)
  lt2 = a2x2;       // s(25,13)
  lt3 = lt1 + lt2;  // s(26,13) = y1 + a2x2
  PComp = lt3 >> 9;  // s(17,4) - EQ 8 = PComp

  return (sint16)PComp; // By calibration this is less than 16 bits
}

/*************************************************************\
* Calculate the compensated pressure PComp value using the brief version
\*************************************************************/
sint16 calculatePCompShort(uint16 Padc, uint16 Tadc, sint16 a0, sint16 b1, sint16 b2, sint16 c12)
{
  sint32 c12x2, a1, a1x1, y1, a2x2, PComp;

  // Pressure calculation (short)
  //============================
  // This version of the pressure calculation function has the same function as the long
  // version and gets exaclty the same result, but is implemented more succinctly.

  Padc >>= 6; //Note that the Padc is the raw value from Pegasus >>6 since its 10 bit unsigned
  Tadc >>= 6; //Note that the Tadc is the raw value from Pegasus >>6 since its 10 bit unsigned

  c12x2 = (((sint32)c12) * Tadc) >> 11;  // c12x2 = c12 * Tadc
  a1    = (sint32)b1 + c12x2;          // a1   = b1  + c12x2
  a1x1  = a1 * Padc;                   // a1x1  = a1  * Padc
  y1    = (((sint32)a0) << 10) + a1x1;  // y1    = a0  + a1x1
  a2x2  = (((sint32)b2) * Tadc) >> 1;   // a2x2  = b2  * Tadc
  PComp = (y1 + a2x2) >> 9;            // PComp = y1  + a2x2

  return (sint16)PComp;
}
```

Table 7. Revision History

Revision number	Revision date	Description of changes
6	06/2012	• In Section 1, added Operating Characteristics table and figures for SPI and I^2C Pin Connections, in Section 2, updated Figures 7, 8 and 9.

How to Reach Us:

Home Page:
www.freescale.com

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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http:/www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

AN3785
Rev. 6
06/2012