1 Introduction

This document demonstrates how to use the on-chip temperature sensor of Freescale S08 microcontrollers to compensate temperature changes to external components or internal peripherals. Many S08 microcontrollers contain an on-chip temperature sensor that can be used to perform temperature compensation. This document will:

1. Describe the benefits of using temperature compensation algorithms.
2. Provide information about how temperature readings are taken using S08 microcontrollers’ on-chip temperature sensor.
3. Provide examples of temperature compensation for different types of external components and internal peripherals.

Some common system peripherals that benefit from temperature compensation are:

- Crystal oscillators
- Altimeters
- Barometers
- Pressure gauges
- pH instruments
- Internal reference clocks

Contents

1 Introduction .................................................. 1
2 Benefits of Using Temperature Compensation ........ 2
3 Integrated Temperature Sensor on S08 Devices ....... 3
4 Temperature Compensation Methods ................... 4
4.1 Internal Reference Compensation .................... 5
5 Temperature Compensation Algorithm ................ 7
5.1 Adding Temperature Trim Software to Your Application 8
Using the Demonstration Code 8
Functions 9
Improvements 9
6 Temperature Compensation without Trim Registers ... 10
7 Conclusions .................................................. 10
Benefits of Using Temperature Compensation

For the listed peripherals, changes in temperature result in changes to performance. For example, the output of a crystal oscillator changes at hot and cold temperatures. The change in the crystal oscillator performance at hot and cold temperatures are shown in the manufacturer’s data sheet for the crystal oscillator. The cost of crystal oscillators with less deviation across temperature is higher than the cost of crystal oscillators with more deviation. Therefore you can save the BOM cost and the overall system cost.

Using methods described in this application note allows end applications to achieve benefits such as higher accuracy sensor readings and extended operating ranges.

This application note applies to Freescale microcontrollers that contain an on-chip temperature sensor that is specified in the analog-to-digital converter (ADC) section of the data sheet. Even though the examples and software pertain to the S08 family, any microcontroller with an integrated temperature sensor can benefit from the methods described in this document. Below is a list of some of the S08 families that contain the on-chip temperature sensor:

- S08AW
- S08QG
- S08QD
- S08SG
- S08LC
- S08QE
- S08SH
- S08DZ

2 Benefits of Using Temperature Compensation

Many system peripherals contain information on how the performance of the peripheral changes as the temperature changes. For example, the Freescale MPX10 series of uncompensated 10 kPa pressure sensors contain information regarding how this sensor behaves across its operating temperature range. This information is used to create temperature compensation described in AN840 Temperature Compensation Methods that can be found at www.freescale.com. Sensors with internal compensation are also available, but at a higher cost. The benefit of lower overall system cost can be achieved by using a method where the internal temperature sensor of the S08 microcontroller is used as the reference to perform temperature compensation.

Another benefit of using temperature compensation is extending the operating range of the end application. For example an application that requires accurate serial communications requires an accurate reference clock. If the reference clock loses accuracy at a hot temperature, the end application can not accomplish the serial communications. Using temperature compensation to modify the serial communications at a hot temperature allows the end application to operate across a wider range.

Both of the benefits listed above add value to the end application by using only the available on chip features.
3 Integrated Temperature Sensor on S08 Devices

Some S08 devices contain an integrated temperature sensor. To use the on-chip temperature sensor to perform temperature compensation relies on the understanding of how to use the ADC to take a temperature reading. A procedure for doing this is outlined in AN3031 Temperature Sensor for the HCS08 Microcontroller Family. The most basic representation of how to use the on-chip temperature sensor is shown in the flow chart below.

![Temperature Reading Flowchart](image)

In AN3031 Temperature Sensor for the HCS08 Microcontroller Family, software is used to implement the flowchart above. The parameters $V_{\text{temp25}}$ and $m$ are provided by the reference manual of the MCU that is used. The temperature reading begins with using the ADC to perform a conversion on the temperature input channel. This data represents $V_{\text{temp}}$. Software algorithms are then used to perform the flow chart above.

In AN3031 Temperature Sensor for the HCS08 Microcontroller Family, two separate software implementations are described. The floating point method is the least complex for user software development. This method requires software libraries be used to implement the floating point math necessary to do the temperature sensor calculations. The second method, fixed point, is an alternative that uses less application code space. The drawback is a limitation to the accuracy of the method and more intensive software that must be used.

Along with different software methods, to use the on-chip temperature sensor you must also decide if calibration must be done. The more calibration done the more accurate the final temperature sensor reading is. The drawback to calibration is that each part requires time and resource to calibrate the temperature sensor.

Table 1 summarizes the various benefits and drawbacks for different temperature sensor procedures.

Temperature Compensation, Rev. 0
Temperature Compensation Methods

The methods used for this application note is a variation of the fixed point method discussed in *AN3031 Temperature Sensor for the HCS08 Microcontroller Family*. This method’s accuracy is improved by extending the size of key parameters so vital information does not get lost in the fixed point method. For more information on using the on-chip temperature sensor please refer to *AN3031 Temperature Sensor for the HCS08 Microcontroller Family*.

4 Temperature Compensation Methods

The theory behind performing temperature compensation is straightforward. It involves taking a temperature reading and changing the peripheral parameters based on a knowledge base of how the peripheral performs across temperature.

Peripheral parameters are external or internal factors that affect the operation of the peripheral. In the case of an internal reference clock, such as the ICS internal reference clock on many S08 microcontrollers, the peripheral parameter that affects the output frequency is the TRIM register. Many system peripherals contain a trim register that allows compensation of the output of the peripheral.

Another example of a peripheral parameter are equation variables. For example, if the peripheral is a sensor that outputs an analog voltage and the output contains an offset that is temperature dependant, then the peripheral parameter that must be changed at different temperatures are the equation variables that account for the offset voltage.

To understand the peripheral parameters and how they must be changed, a knowledge base must be referenced. An example of a knowledge base is a specification. For example, the specification for the Freescale MPX10 series contains the parameters that define how the output of the pressure sensor changes across temperature ranges.

A knowledge base can also be generated by data collection. For example, the output of an internal reference signal is measured across the operating temperature range. This data can then be used to modify the system peripheral parameter to compensate for the changes across the temperature range. The following example looks closely at a performing oscillator trim compensation, using a temperature calibration reading to explain temperature compensation.

<table>
<thead>
<tr>
<th>Temperature Reading Method</th>
<th>Accuracy</th>
<th>Code Size</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating point support — no calibration</td>
<td>+ / – 10 °C</td>
<td>&gt;2 K</td>
<td>Implementation is simple, no work is done for calibration.</td>
</tr>
<tr>
<td>Floating point — 3 point calibration.</td>
<td>+ / – 2.5 °C</td>
<td>&gt;2 K</td>
<td>Each unit must be calibrated. This implementation is complex.</td>
</tr>
<tr>
<td>Fixed point — no calibration</td>
<td>+ / – 15 °C</td>
<td>&lt;1 K</td>
<td>Implementation requires software to manage equations using the fixed point method, no work is done for calibration.</td>
</tr>
<tr>
<td>Fixed point — 3 point calibration</td>
<td>+ / – 8 °C</td>
<td>&lt;1 K</td>
<td>Implementation requires work to calibrate each unit. This implementation is complex.</td>
</tr>
</tbody>
</table>
4.1 Internal Reference Compensation

Many S08 microcontrollers contain an internal reference signal. This internal reference signal can be used as the basis for the system and bus clock for the microcontroller. The internal reference is specified to be ±2% across the full operating temperature range of the device. Figure 2 shows how the internal reference is deviated in frequency output over a range of temperatures for a single unit.

![Deviation % vs. Temperature (Celsius)](image)

**Figure 2. Uncompensated Deviation Versus Temperature**

This data shows that as temperature increases, the internal reference clock (IRC) slows down. This change to the IRC can lead to unwanted application behavior at hot temperatures if the reference is not compensated.

Compensation of the internal reference is accomplished by using the TRIM bits. The final bus frequency if using the internal reference clock is based on the value placed in the ICSTRIM. The TRIM is composed of the ICSTRIM register and the fine trim bit (FTRIM). The FTRIM is used to increase or decrease the resulting internal reference by approximately 0.1%. The FTRIM bit is half the resolution of the TRIM register bits. The TRIM register bit resolution increases or decreases the resulting IRC by 0.2%. Using the TRIM register along with the FTRIM allows the ICS reference to be set to a value that is less than 0.2% of the desired frequency.

Calibration is done by placing the MCU at different temperatures and determining what the TRIM value for that temperature must be. Table 2 represents the calibration data taken for one unit. This data represents the knowledge base that is used to perform temperature compensation.
Temperature Compensation Methods

Table 2. Calibration Data

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Desired Trim Value</th>
<th>FTRIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>9A</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>9C</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>9C</td>
<td>1</td>
</tr>
<tr>
<td>–20</td>
<td>9C</td>
<td>1</td>
</tr>
<tr>
<td>–40</td>
<td>9C</td>
<td>0</td>
</tr>
</tbody>
</table>

MCU software can use the data to set the corresponding trim value at different temperatures. Using the calibration data, when – 40 °C is reached the TRIM value must be set to 9 °C and FTRIM = 0.

The calibration data shows the exact trim value that must be set for six different temperatures. For temperatures that fall between these six temperatures a linear interpolation can be used to determine what the trim value must be. For example, between 25 °C and 85 °C there are four steps in the trim value. Each of these four steps can be divided throughout temperatures between 25 °C and 85 °C and compensation can be done across this range. At 32.5 °C the TRIM must be set to 9 °C and FTRIM = 1. At 47.5 °C the TRIM must be set to 9B and FTRIM = 1 as shown below.

Using this data an algorithm that changes the setting of the ICSTRIM based on a temperature reading can be created. This algorithm takes a temperature reading then adds or subtracts to the ICSTRIM value based on the reading. The C code below shows the compensation algorithm.

Figure 3. Trim Compensation across temperature

Using this data an algorithm that changes the setting of the ICSTRIM based on a temperature reading can be created. This algorithm takes a temperature reading then adds or subtracts to the ICSTRIM value based on the reading. The C code below shows the compensation algorithm.
# Temperature Compensation Algorithm

```c
if(Temperature <= 33){
    ICSTRM = 0x9C;
    ICSSC_FTRIM = 1;
} else if((Temperature <= 48))
{  ICSTRM = 0x9C;
    ICSSC_FTRIM = 0;
} else if((Temperature <= 63))
{ ICSTRM = 0x9B;
    ICSSC_FTRIM = 1;
} else if((Temperature <= 78))
{ ICSTRM = 0x9B;
    ICSSC_FTRIM = 0;
} else if((Temperature >= 79)){
    ICSTRM = 0x9A;
    ICSSC_FTRIM = 1;
}
```

As shown in the C code example, as temperature changes the Trim register is changed. Each bit in the ICS TRIM register changes the resulting ICS reference by 0.2%. Also, by changing the FTRIM bit, the ICS reference changes by approximately 0.1%.

The assumption that this algorithm uses is that the deviation in the internal reference between calibration points is linear.

The algorithm relies on current temperature readings as they relate to the calibration data of the ICS reference. At temperatures where the expected ICS deviation reaches above 0.05% the trim value (9 bits, TRIM and FTRIM) changes therefore ICS reference is corrected and the deviation is kept as low as possible.

The results for a single unit are shown below. Figure 4 shows how temperature compensation improves the accuracy of the internal reference across the operating temperature range for this single unit. Without temperature compensation, the internal reference deviation is up to 1% at 125 °C; with temperature compensation, the result is within 0.2%. 

---

Temperature Compensation Algorithm

Freescale Semiconductor

7
5.1 Adding Temperature Trim Software to Your Application

5.1.1 Using the Demonstration Code

The demonstration code contains all of the functions necessary to perform trim compensation across temperature. The algorithm implemented in the code uses the calibration data taken for this application note as the knowledge base for the compensation. Therefore the benefits of using this algorithm on the typical unit are not as great as the example shown in this application note.

Table 3 outlines the steps necessary to program the code into an MCU. The attached software is configured to operate on a DEMOQG8 board. After the code has been programmed, PTA1 can be monitored. The signal on PTA1 is 100 kHz. The TPM is configured to output this signal. The ICS reference deviation can be calculated by taking the measured TPM signal and subtracting 100 kHz. This, divided by 100 kHz results in the percent deviation.
Temperature can be varied while the PTA1 output is monitored. Make sure the percent error stays low as shown in Figure 4.

### 5.1.2 Functions

Figure 4 lists the functions used to perform trim compensation across temperatures. Understanding how these functions are used and how to call the functions helps in implementing the ICS trim software into the final application code.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
<th>Calling procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitTempSensorParameters</td>
<td>Initializes the parameters used to calculate the temperature in °C.</td>
<td>After the ADC is initialized  \nInitTempSensorParameters();</td>
</tr>
<tr>
<td>GetTemperature</td>
<td>Returns the value of the temperature in °C</td>
<td>After Init TempSensorParameters is called  \nTemperature = GetTemperature();</td>
</tr>
</tbody>
</table>

### 5.1.3 Improvements

The attached software and algorithms can be improved. The software was created using calibration data for a single unit. From unit to unit the performance of the internal reference varies. Even though this code
shows some improvement to the internal reference, the best improvement are accomplished by calibrating the specific unit being tested.

6 Temperature Compensation without Trim Registers

Not all peripherals contain trim registers. Sometimes compensation must be done through software calculations. For example, the 1 kHz low-power oscillator available on many S08s does not contain a trim register. This clock source changes across a temperature range. To compensate this clock source software must change the equations used to keep track of time intervals.

For example: The low power oscillator is used to generate 16 ms interrupts, and 100 of these interrupts are counted to generate a 1.6 s time interval. If the low power oscillator output changes to 16.33 ms at 50 °C, temperature compensation can then be done by changing the software and 98 interrupts can represent 1.6 ms.

In this scenario, temperature compensation is achieved by changing a count value for a time calculation. The same concepts can be used to compensate for voltage variations across temperature from a voltage reference.

For Example: A voltage reference IC is used to bias a sensor. The output of a voltage reference IC is specified to change by 100 ppm/ °C. This data can be used to compensate the voltage reference output and sensor readings can account for the different value of the bias voltage.

7 Conclusions

Using the on-chip temperature sensor to perform compensation across temperature adds value to end applications. Lower cost can be achieved by using compensation for peripherals such as uncompensated pressure sensors or crystal oscillators. Temperature compensation can also add accuracy that is critical in many applications. Finally temperature compensation can be used to extend the operating range of the application.
Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2008. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as 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.

Document Number: AN3570
Rev. 0
01/2008