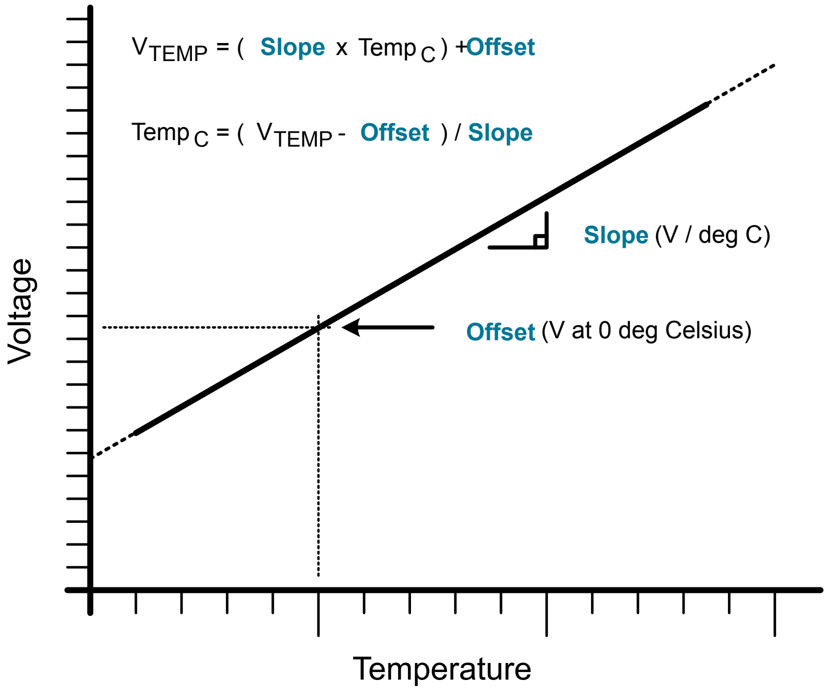
AN929: Accurate Temperature Sensing with the EFM8 Laser Bee MCU Family

The EFM8 Laser Bee family is a performance line of 8-bit micro-controllers that is well-suited to a variety of applications requiring high performance from both digital and analog peripherals in a small package.

Digital features include an operating frequency of up to 72 MHz, up to 64 KB of flash memory, four channels of configurable logic, and five serial communication modules. Analog features include a 14-bit ADC, two precision voltage references, up to four 12-bit DACs, two comparators having built-in reference DACs, and a factory-calibrated temperature sensor. This application note explains how to achieve high measurement accuracy when using Laser Bee's temperature sensor.

KEY POINTS

- The Laser Bee family includes an on-chip temperature sensor that is capable of higher levels of sensing accuracy than the legacy sensor included on most other Silicon Labs MCUs.
- The MCU must be set up properly in order to achieve the highest accuracy, and self-heating effects need to be taken into account.
- Statistical information is provided that will enable the estimation of the accuracy of the temperature sensor in a particular application.



1. Temperature Sensors on Silicon Labs MCUs

Three different types of temperature sensors are used on Silicon Labs 8-bit MCUs.

1.1 Legacy Temperature Sensors

Most Silicon Labs MCUs include a temperature sensor that can be monitored using the on-chip ADC. The sensor produces a voltage that is proportional to absolute temperature (PTAT). The slope of the temp sensor's voltage versus temperature characteristic is quite consistent from device to device, but there is considerable part-to-part variation in the temperature sensor's offset voltage, which is defined as the output voltage at 0 degrees Celsius. The total temperature error at any given temperature is proportional to the sum of the offset voltage error and the slope error multiplied by the temperature difference between the temperature being measured and 0 °C.

For example, the EFM8SB1, part of the EFM8 Sleepy Bee family, has these specifications for its temperature sensor offset and slope:

- Typical offset voltage — 940 mV
- Offset voltage error (1 standard deviation) — 18 mV
- Typical slope — 3.4 mV/°C
- Slope error (1 standard deviation) — 40 µV/°C

A quick calculation shows that three standard deviations of the offset voltage is 3 * (18 mV) = 54 mV. Dividing this value by the typical slope of 3.40 mV/°C shows that three standard deviations (or three sigma) of offset voltage error corresponds to a temperature error of almost 16 degrees Celsius. Therefore, the uncalibrated temperature sensor is not well-suited for applications requiring high accuracy. On the EFM8SB1 and some other Silicon Labs MCUs, a temp sensor offset measurement is performed by the ADC during factory testing while ambient temperature is measured using a sensor on the test fixture; a corrected offset value is then stored in SFR registers TOFFH and TOFFL for use by the firmware. Possible differences in temperature between the test fixture and the MCU being tested limit the accuracy of this calibration to about ±5 °C.

However, note that the slope error is low enough that an accurate calibration performed at one temperature would provide a high level of temperature sensor accuracy across the full temperature range. For the EFM8SB1, three standard deviations of the slope error is 3 * (40 µV/°C) = 120 µV/°C. Given an ideal 1-point calibration near 25 °C, this amount of slope error will contribute a three-sigma temperature error of only ±2.3 °C across a range of -40 °C to +90 °C.

1.2 Precision Temperature Sensor (C8051F39x/37x)

In addition to a legacy temperature sensor, MCUs in the C8051F39x/37x family include a proprietary, self-contained temperature-to-digital converter that directly produces a two's complement representation of the temperature with a resolution of better than 0.01 °C. This converter is factory-calibrated and achieves better than ±2 °C accuracy across the full temperature range of the device (-40 °C to +105 °C for the C8051F39x, and -40 °C to +85 °C for the C8051F37x).

This precision temperature sensor adds to the cost of the device, and realizing its full potential can be challenging on a programmable, general-purpose MCU, due to self-heating effects that vary with the activity levels of the CPU, memories, GPIO, and each of the analog and digital peripherals. But this sensing technology is also employed in Silicon Lab's Si70xx family of temperature and humidity sensors, where deterministic operation and very low power consumption enable self-heating effects to be minimized. Those sensors achieve inaccuracies as low as ±0.3 °C across their full operating voltage and temperature ranges.

1.3 Temperature Sensing with the Laser Bee Family

The EFM8 Laser Bee family incorporates a temperature sensor that is similar in architecture to the legacy temperature sensors found on most other Silicon Labs MCUs, with two key improvements:

1. Lessons learned during the development and characterization of the precision temperature sensor on the C8051F39x/37x family have been incorporated into the EFM8 Laser Bee family.
2. A new on-chip sensing element has been added, which works in conjunction with a custom interface on the production tester to enable accurate measurement of the device temperature during the device calibration sequence. With knowledge of the device temperature and the temp sensor slope, an accurate offset code is computed and stored in a specific location in the read-only area of flash memory.

The first improvement serves to minimize the slope error, and the second improvement minimizes the offset error. The resulting sensor provides much higher accuracy than the legacy temperature sensor with lower cost than the precision sensor included on the C8051F39x/37x family. The remainder of this application note provides guidelines for achieving the best possible accuracy from the Laser Bee temperature sensor.

2. Making Accurate Temperature Measurements

2.1 Offset Calibration

During the factory calibration process, the tester measures the temperature of the silicon die inside the package using an on-chip sensing element that is accessible using a special testing mode. This is a “room temperature” operation (no temperature forcing equipment is used), and the measured die temperature will be typically between 20 °C and 35 °C. At the same time, Laser Bee’s 14-bit ADC measures the output of the temperature sensor, using the 1.65 V internal fast-settling voltage reference. The tester uses the ideal temperature sensor slope (provided in the Electrical Specifications section of the data sheet) to translate the ADC’s output code to a value corresponding to 0 °C; this translated value is then stored in the read-only area of flash, in these locations:

- 0xFFD4 = low byte of the ADC data word (ADC0L)
- 0xFFD5 = high byte of the ADC data word (ADC0H)

The result is a 14-bit, right-justified value, so the top two bits of ADC0H are always zero. The remaining bits will vary from part to part; for most devices, the value stored in flash will have a decimal (base 10) value between 6900 and 7900. Since the full-scale output of the 14-bit ADC is a decimal value of 16384, and the reference voltage is 1.65 V, this range of values corresponds to a temp sensor output voltage range of approximately 700 mV to 800 mV at 0 °C.

The temperature of 0 °C is chosen to simplify temperature calculations within the firmware. To calculate the die temperature, the firmware measures the temperature sensor output using the ADC, and the calibrated value at 0 °C (stored in flash) is subtracted from the result. The die temperature in degrees Celsius is the result of this subtraction divided by the temp sensor slope, where the slope is given in units of “ADC codes per °C” (or LSBs per °C). For a temp sensor slope of 2.83 mV/°C, a temperature change of 1 °C will cause a change at the ADC input of 2.83 mV, which corresponds to a change in the ADC output of (2.83 mV / 1.65V) * 16384 = 28.1 codes. Using a value of 28 instead of 28.1 for the slope will simplify the computation and result in a relatively small difference of ~0.3 °C at 85 °C.

Here is a numerical example, with all values given in decimal notation:

1. Calibration value stored in flash = 7408
2. ADC output when measuring the temperature sensor = 9245
3. Die temperature = (9245 – 7408) / 28 = 65.6 °C

2.2 Setting up the ADC for Temperature Sensor Measurements

Since the temperature calculation involves subtracting the calibration value from an ADC measurement made by the firmware, the most accurate result will be achieved by setting up the ADC the same way it was set up when the MCU was characterized and calibrated at the factory. It may be helpful to start with a code example provided through Simplicity Studio using the [**Software Examples**] tile, such as the [**EFM8LB1_TempSensor_WithCompensation**] example for the EFM8LB1 STK.

The most important consideration is that the 1.65 V internal voltage reference must be used. The characterization and calibration results assume that the temp sensor, ADC, and 1.65 V reference all operate as one unit. For example, the published slope value for the temperature sensor incorporates all temperature variation occurring in any of those three modules. Using a different reference will result in significant deviation from the temp sensor specs published in the data sheet. Two other important factors are averaging of the ADC results to minimize noise and considering the effects of self-heating, which are also discussed in this document.

Beyond these three key considerations, most other ADC settings do not significantly affect the results, and can be chosen based on preference. Table 2.1 Recommended ADC Register Settings on page 3 shows the ADC register settings that were used for the temp sensor characterization described in 3. Temperature Sensor Performance.

Table 2.1. Recommended ADC Register Settings

Register	Bit Field	Descriptions and Comments
ADC0CN0	ADEN = ENABLED	Enable ADC0
	IPOEN = POWER_DOWN	Power down when ADC idle (minimize self-heating)
	ADGN = GAIN_1	The on-chip PGA Gain is 1
	TEMPE = TEMP_ENABLED	Enable the Temperature Sensor
ADC0CN1	ADBITS = 14_BIT	ADC0 operates in 14-bit mode
	ADSJST = RIGHT_SHIFT_3	Right justified. Shifted right by 3 bits
	ADRPT = ACC_8	Perform and Accumulate 8 conversions (see 2.3 Accuracy Versus Amount of Averaging for additional details)
ADC0CF0	ADSC = 0x01	F(SARCLK) = F(ADCCLK) / 2 = 12.25 MHz
	ADCLK_SEL = HFOSC0	Selects HFOSC0 as ADCLK (24.5 MHz)
ADC0CF1	ADLPM = LP_DISABLED	Disable low power mode (gives shorter conversion times and slightly lower average power dissipation)
	ADTK = 0x04	Sets tracking time to 326 ns
ADC0CF2	REFSL = INTERNAL_VREF	ADC0 voltage reference is the Internal 1.65 V reference
	ADPWR = 0x06	Power up delay time = 1.22 µs
ADC0MX	ADC0MX = TEMP	Selects the temperature sensor as the ADC input

2.3 Accuracy Versus Amount of Averaging

The ADC, voltage reference, and the temperature sensor all contribute noise. The majority of the noise is white noise, so the measurement accuracy can be improved by averaging multiple measurements. The table below shows the statistics of 1024 readings made on the temp sensor at 25 °C with varying amounts of averaging used for each reading.

Table 2.2. Measurement Accuracy versus Amount of Averaging

Number of ADC samples averaged for each reading	1 standard deviation (in 14 bit LSBs)	3 standard deviations (in 14 bit LSBs)	3 standard deviations (temperature error)
1	6.215	18.645	0.67°C
4	3.255	9.765	0.35°C
8	2.482	7.446	0.27°C
16	1.934	5.802	0.21°C
32	1.609	4.827	0.17°C
50	1.460	4.380	0.16°C
100	1.118	3.354	0.12°C
200	0.789	2.367	0.08°C

Averaging 4, 8, 16, or 32 samples can be automatically performed using the ADC's hardware accumulator. Since the calibration value is a 14-bit, right-justified result, there are a few convenient choices using the ADC's hardware accumulator (configured using the ADC0CN1 SFR) that will also produce a 14-bit result and simplify the temperature calculation. One is to set ADBITS = 14_BIT, ADRPT = ACC_8, and ADSJST = RIGHT_SHIFT_3 to average eight 14-bit samples and produce a 14-bit result. Another choice would be to set ADBITS = 12_BIT, ADRPT = ACC_32, and ADSJST = RIGHT_SHIFT_3, which will also produce a 14-bit result; the larger amount of averaging will produce similar results using either 14-bit or 12-bit mode.

Firmware may also be used to average ADC samples and shift the result to a 14-bit representation. Experiments have shown negligible differences in the results from averaging samples using the hardware accumulator and results using firmware averaging, or when the samples to be averaged are taken over a relatively short timeframe (e.g. 100 microseconds) or over a longer timeframe (e.g. 100 milliseconds).

Note that the 10-bit mode does not provide sufficient resolution for accurately reading the temperature sensor. Additional background information on reducing noise through averaging is available in Silicon Labs Application Note AN118, "Improving ADC Resolution by Oversampling and Averaging". Application notes can be found on the Silicon Labs website (www.silabs.com/8bit-appnotes) or in Simplicity Studio by clicking the [**Application Notes**] tile.

2.4 Self-Heating

Self-heating of the MCU during temperature measurements can significantly affect the measurement result. With self-heating, the temp sensor is reporting the correct silicon die temperature inside the MCU package, but that temperature is elevated above the ambient temperature (which is what most applications want to measure) due to power dissipation within the MCU. The amount of self-heating is the product of the power dissipation and the overall thermal resistance (in °C/W) between the MCU and the surrounding air. When the power dissipation and thermal resistance are known, the amount of self-heating is known, and the ambient temperature can be computed by subtracting the amount of self-heating from the temperature recorded by the temp sensor.

2.4.1 Limitations of Using Published Values for Thermal Resistance

The Thermal Conditions section of the data sheet's Electrical Specifications chapter provides the thermal resistances (θ_{JA}) for each package type. While these numbers may be used to compare the thermal performance of different packages, they are seldom useful for calculating the self-heating that will be seen in applications, since θ_{JA} is calculated under a controlled set of PC board parameters such as board thickness and composition, copper plating thickness, trace length, number of layers, and air flow. Changing any of these factors can greatly affect θ_{JA}. For example, an EFM8LB12 in a QFN32 package has an overall thermal resistance of about 95°C/W when measured on a 6 cm by 6 cm socketed test board used during device characterization, while the same device soldered to a 10 cm x 10 cm test board exhibits a thermal resistance of about 50°C/W. Many MCUs are used in applications that result in more die heating than would be predicted by θ_{JA} alone, due to factors such as small PC boards (which heat up with the MCU and add to the overall die temperature change) and small enclosures which trap heat. Furthermore, the type and amount of physical materials in close proximity to the MCU will determine the thermal mass distribution, which affects the change in temperature with time that occurs when the MCU's power dissipation changes.

2.4.2 In-situ Calculation of Thermal Resistance

For the reasons given previously, it is best to measure the overall thermal resistance between the MCU's silicon die and ambient in the actual physical application, and then measure the power dissipation when running the application firmware to determine the amount of self-heating. The thermal resistance may be measured using the calibrated on-chip temperature sensor to measure the die temperature at two different power dissipation values; the thermal resistance is then the change in temperature divided by the change in power dissipation.

To minimize measurement errors when measuring thermal resistance, the two values of power dissipation should differ by a significant amount, preferably 30 mW or more. Two methods may be used to provide the two different values of power dissipation: changing the system clock frequency, or applying external currents into GPIO pins. Running a firmware loop with SYSCLK derived from HFOSC1 (72 MHz) will dissipate much more power than running the same firmware loop with SYSCLK derived from LFOSC0 (80 kHz), so these two conditions will provide a large difference in power dissipation. The ADC can be clocked from HFOSC0 in both cases to provide an "apples-to-apples" comparison of temp sensor readings. To use the GPIO method, one or more GPIO pins can be driven high with firmware (push-pull output mode, low drive strength), and the die temperature can be recorded while the GPIO pins are open and again while they are shorted to ground. Shorting the GPIO pins to ground will result in higher power dissipation than leaving them open. To ensure uniform heat distribution across the die using this method:

- It is better to use at least two GPIO pins located on opposite sides of the package.
- On the QFP and QFN packages, the corner pins should not be used for this test.
- On the QSOP package, pins 3, 4, 9, 10, 15, 21, and 22 bond to the corners of the die, and they should not be used for this test.

2.4.3 Self-Heating Calculation and Thermal Time Constant

Once the thermal resistance is known, the amount of self-heating can be calculated. In some cases, the self-heating is small enough that it does not materially affect the result. For example, the firmware routines that were used to characterize the slope of the Laser Bee temperature sensor result in a current consumption of about 1.1 mA, giving a power dissipation of 3.63 mW with a 3.3V power supply. The test board had a measured thermal resistance of about 50 °C/W, so the amount of self-heating was (3.63 mW) * (50 °C/W) = 0.18 °C. The slope characterization is not affected by a fixed temperature offset, but even if it were, this small amount of self-heating could be ignored in many cases.

However, many applications will operate with higher thermal resistance and with much higher power dissipation than this example. Since the power dissipation may vary over a wide range during different operating modes of a general-purpose MCU, the thermal time constant of the MCU must be known in order to calculate how long a particular power dissipation level must be maintained to ensure that the die temperature has stabilized at its steady-state value, so that firmware can apply a valid self-heating correction to the output of the temp sensor. This thermal time constant can easily be determined when measuring the thermal resistance. If the ADC is set up to read the temp sensor, the conversions are triggered by a timer, and the results are stored in XRAM, then a profile of the die temperature versus time can be recorded as the power dissipation is switched between low and high values. This profile can then be used to determine how long the firmware must wait before taking a temperature reading when the power dissipation changes in the final application.

3. Temperature Sensor Performance

This section provides information on the performance of the Laser Bee temperature sensor if all of the error sources described in the previous section are minimized. To obtain the data in this section, the MCUs were placed in a circulating oil bath temperature chamber having a set point accuracy of better than ±0.1°C.

3.1 Slope Error

The temperature sensor slope is fairly consistent from part to part, but this section quantifies the part-to-part variation in the slope. The data on slope error is based on the characterization of 238 MCUs taken from six “split wafers”, in which the wafer manufacturer deliberately introduces specific variations in each wafer that correspond to the outer limits of part-to-part manufacturing variations expected during the manufacturing life of the product. Little difference was seen when comparing the temp sensor’s statistical performance on MCUs from each of the wafers, which indicates that most of the statistical variation is random rather than being correlated to a particular process variable.

The figure below shows the temperature deviation of the actual slope versus a reference linear slope of 28 ADC codes per degree Celsius (2.82 mV/°C); this reference slope was chosen to simplify temperature calculations in firmware. The center curve represents the average, and the upper and lower curves represent 3 standard deviations above and below the average, respectively. The deviation is normalized to a temperature of 25 °C, which is why the three curves converge there. Note that the 25 °C point is shifted down to a temperature deviation of -0.5 °C, rather than 0 °C; this value is arbitrary, but it was chosen to make the upper and lower curves symmetrical around a deviation of 0°C at the minimum temperature, which facilitates a symmetric accuracy specification in the product data sheet. The bias of -0.5 °C is added during factory calibration and does not need to be included in the firmware’s temperature calculation.

The figure below shows that with a perfect calibration at 25 °C, all temperature sensors will report a temperature of 24.5 °C at an actual temperature of 25 °C, due to the bias of -0.5 °C. At -20 °C, an average temp sensor will report a temperature very close to -20 °C, and the three sigma range shown by the lower and upper curves will be approximately (-20 °C - 1.7 °C) = -21.7 °C at the lower three-sigma point, and (-20 °C + 1.6 °C) = -18.4 °C at the upper three-sigma point.

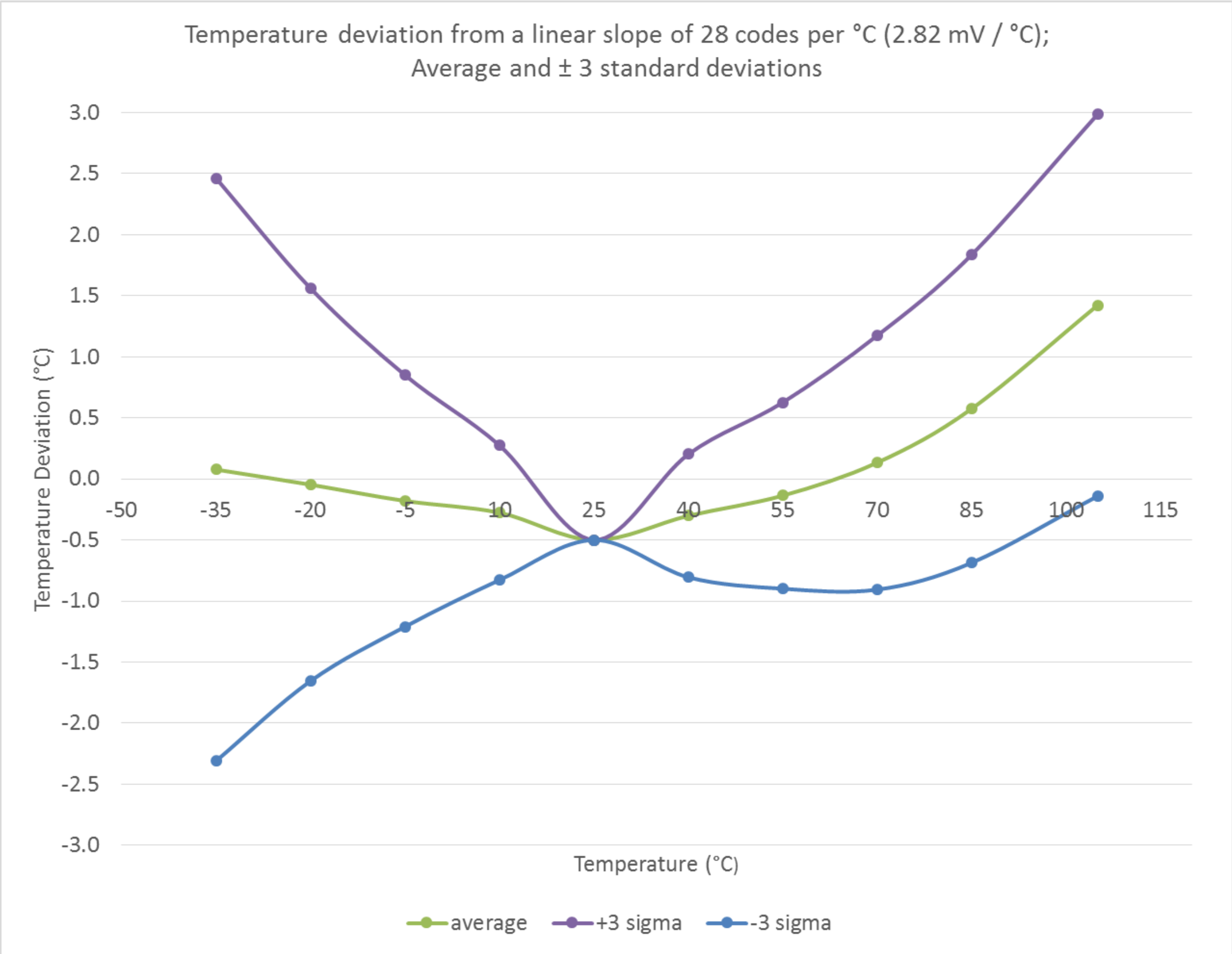


Figure 3.1. Deviation from a Linear Temperature Sensor Slope

Using 28 ADC codes per degree Celsius for the reference slope is a good overall choice for most common application temperature ranges, such as 0 °C to +70 °C, -20 °C to +85 °C, or -40 °C to +105 °C. The shape of the middle (average) curve in Figure 3.1 suggests that the average slope is slightly less than 28 ADC codes per °C at temperatures below 25 °C, and it is slightly more than 28 codes per °C at temperatures above 25 °C. 3.4 Improving High-Temperature Accuracy Using a Two-Slope Model shows how this information may be used to derive a more accurate temperature estimate at higher temperatures, at the cost of a small increase in firmware complexity.

3.2 Offset Calibration Error

2.1 Offset Calibration describes the offset calibration process, in which an offset code at 0 °C is programmed into flash memory. The value of 0 °C is chosen to simplify the firmware temperature calculation.

The offset codes stored in flash by the production tester can be compared to the ADC output values taken while the MCUs are held at precisely 0 °C using the circulating oil bath chamber to derive statistics on the offset calibration error. However, recall that the offset values are actually measured by the tester at room temperature, and the tester uses the average slope to translate this code to a value that should, on average, correspond to the temperature sensor output at 0 °C. But this translation process introduces an additional error due to any difference between the average slope and the actual slope of the particular MCU being calibrated. Therefore, the original offset value that is measured at room temperature will have less error than the 0 °C offset value stored in flash, and this error will not be correlated to the device’s slope error. The latter point is important when performing a statistical combination of the slope and offset calibration errors in 3.3 Overall Accuracy Using a Linear Slope Model.

For purposes of error calculation, then, the offset calibration is translated from its room-temperature value (typically close to 25 °C) to a value of precisely 25 °C, rather than a value of 0 °C. This small temperature translation introduces very little of the slope error described above, but using a common offset calibration temperature (rather than a unique calibration temperature for every device) simplifies the statistical analysis in 3.3 Overall Accuracy Using a Linear Slope Model.

Initial measurements have shown the 25 °C offset calibration value to have a mean error of 0.2 °C and one standard deviation of 0.293 °C. The calibration procedure is being refined as more devices are calibrated and verified in the oil bath chamber, and the mean error is expected to converge to the -0.5 °C bias value described in 3.1 Slope Error.

3.3 Overall Accuracy Using a Linear Slope Model

The figure below shows the total intrinsic temperature sensor error, which includes the mean and standard deviation of the slope error and the standard deviation of the offset calibration error. This figure includes ±4.5 standard deviation curves in addition to the average and ±3 standard deviation curves. The ±3 standard deviation curves represent the expected accuracy limits when using sample quantities of a few hundred units, such as the 238 units that were used for the slope characterization; in fact, the min and max errors of that sample were very close to the ±3 standard deviation curves. The ±4.5 standard deviation curves represent the expected accuracy limits when measuring many devices. A large sample with a normal distribution will have 3.4 parts per million that fall outside the ±4.5 standard deviation limits.

Note that these curves represent the measurement error when reading the die temperature; they do not include the effects of any self-heating, described in detail in 2.4 Self-Heating.

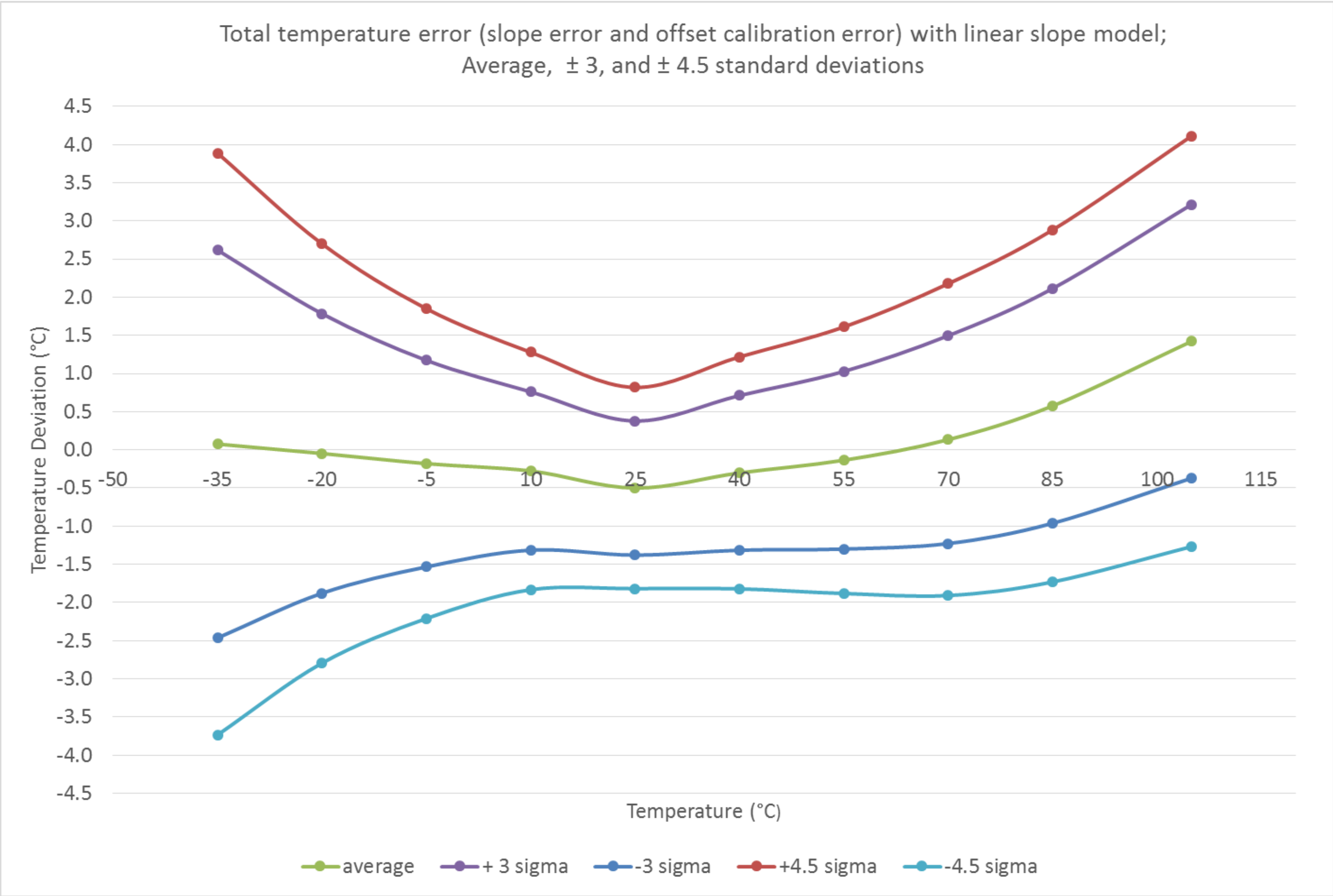


Figure 3.2. Total Intrinsic Temperature Sensor Error with Linear Slope Model

3.4 Improving High-Temperature Accuracy Using a Two-Slope Model

Figure 3.2 Total Intrinsic Temperature Sensor Error with Linear Slope Model on page 9 shows that while the average error curve is not completely flat at low temperatures, the ±3 and ±4.5 sigma curves are quite symmetrical above and below a temperature deviation of 0 °C. This means that there would be little benefit in applying a more complex slope calculation for temperatures below 25 °C. However, for temperatures above 25 °C, all of the curves trend higher, with the four different "sigma" curves having a similar shape to the average curve. By applying a different reference slope value for temperatures above 25 °C, the error curves can be made more symmetrical around a temperature deviation of 0 °C, which will allow a small improvement in accuracy. The figure below is similar to the previous figure, except that instead of using a reference slope of 28 codes per °C at all temperatures, two different slopes are used. For temperatures below 25 °C, a slope of 28 codes per °C is still used, but for temperatures above 25 °C, a slope of 28.3 codes per °C is used. The choice of 28.3 for the secondary slope achieves a symmetrical accuracy at temperatures up to 85 °C; that is, the ±4.5 sigma curves are within ±2.5 °C at 85 °C, rather than being within a window of -2 °C to +3 °C at 85 °C as shown with the previous method. Choosing a secondary slope of 28.5 would provide a symmetrical accuracy of about ±2.7 °C at 105 °C, rather than having an error window of -1.3 °C to +4.1 °C.

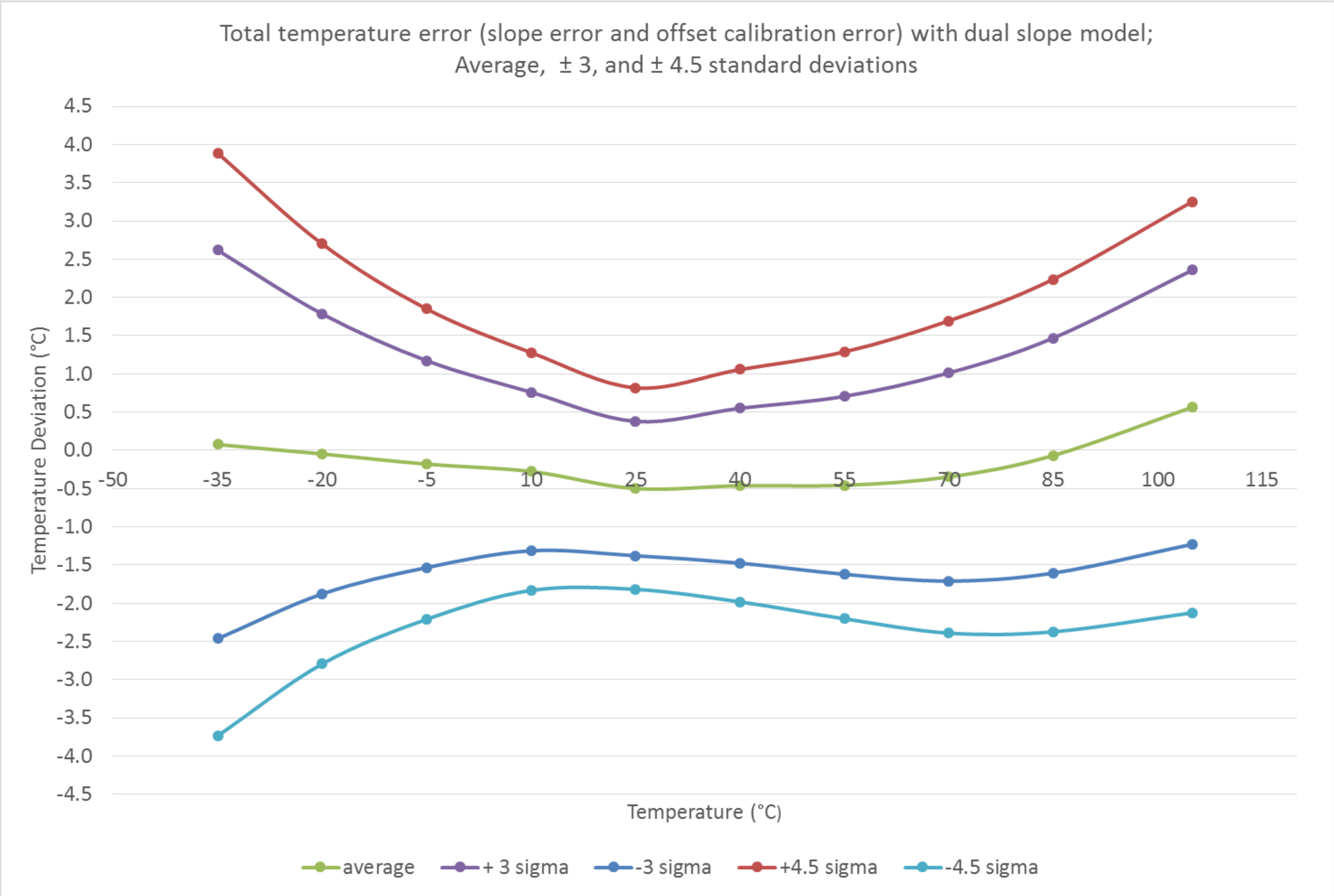


Figure 3.3. Total Intrinsic Temperature Sensor Error with Dual Slope Model

Other, more complex reference slope curves could be used, but going beyond the dual linear slope model is unlikely to provide enough improvement to be worth the additional computation required by the firmware.

Simplicity Studio

One-click access to MCU tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

www.silabs.com/simplicity

MCU Portfolio
www.silabs.com/mcu

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com