EPSON
EXCEED YOUR VISION

S1C17W18 Temperature Compensation Application Note

SEIKO EPSON CORPORATION

Rev.1.0

Table of Contents

1. Summary of Frequency Errors

In most modern MCUs there are multiple clock sources, each with a different frequency and frequency error (tolerance) range. The S1C17W18 specifically has the following clock sources:

- OSC3 – 250kHz, 384kHz, 500kHz, 1MHz, 2MHz, 4MHz, up to +/- 5% (50,000 ppm) at 25°C.
- IOSC – 700kHz internal oscillator, up to +/-7% (70,000 ppm) over temperature.
- OSC1 – 32.768kHz external crystal circuit, typically +/-20ppm tolerance at 25°C.

There are two main factors for frequency error/tolerance:

- Manufacturing variations.
- Temperature drift

Crystals and oscillators typically have a frequency tolerance specification at 25°C which accounts for manufacturing variations.

The frequency also drifts over the operating temperature range of the MCU.

This document describes methods to compensate for manufacturing variations and temperature drift. The S1C17W18 has a built-in temperature sensor whose temperature data can be used in the frequency error compensation algorithms, while other MCUs can use an external temperature measurement component to provide temperature data. Temperature calibration methods will also be described for the built-in temperature sensor.

This application note has a companion sample software which implements some of the algorithms mentioned in this document and facilitates measurements of frequencies for calibration. The sample software operation is described in Section 4.

1. Summary of Frequency Errors

1.1 OSC3 Frequency Error

The internal OSC3 oscillator of the S1C17W18 has a selectable number of nominal fixed frequencies (6 total) to choose from: 250kHz, 384kHz, 500kHz, 1MHz, 2MHz, and 4MHz. Section 23.5 of the S1C17W18 Technical Manual provides a table of the tolerance/error specifications for the OSC3 frequencies. The worst-case frequency error at 25°C, 1.6 to 3.6V, is +/-5% (50,000 ppm).

Section 23.5 of the S1C17W18 Technical Manual also shows the frequency versus temperature graph for OSC3, and it shows that the frequency versus temperature graph of OSC3 has a negative linear slope and frequency decreases as temperature increases.

The frequency of OSC3 can be trimmed/adjusted by the OSC3AJ[4:0] value in the CLGTRIM register. The figure below shows a graph of measured frequency versus temperature for OSC3=4MHz with minimum and maximum frequency adjustment trim values.

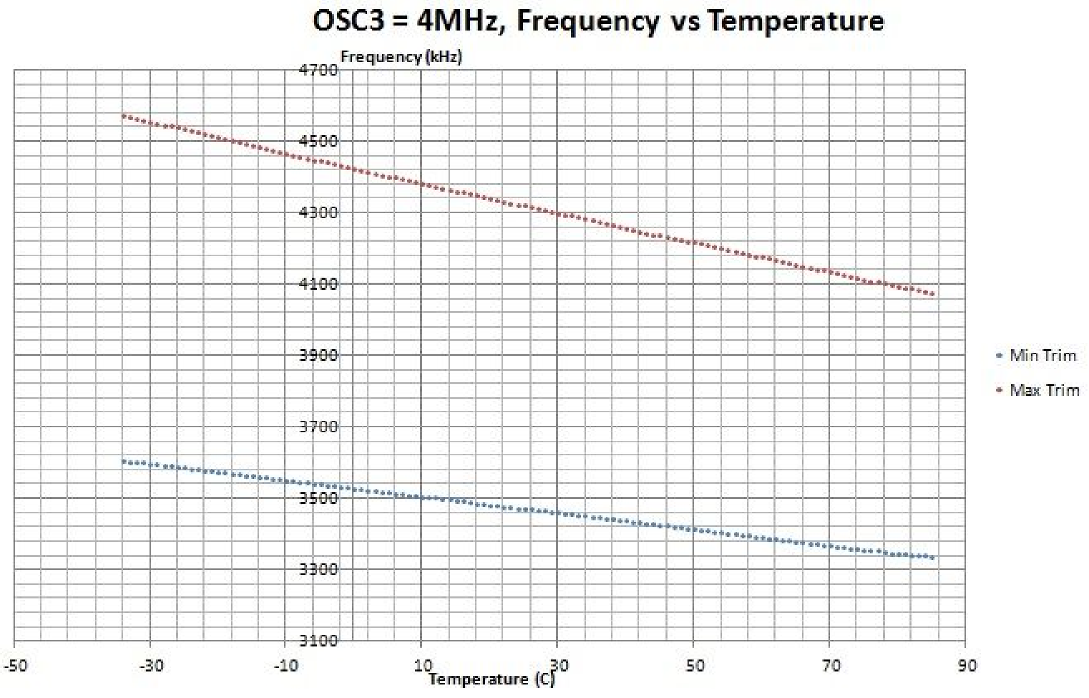


Figure 1-1: OSC3 – 4MHz, Frequency vs Temperature with Min and Max Trim Values

1.2 IOSC Frequency Error

The internal IOSC oscillator of the S1C17W18 has a nominal frequency of 700kHz. Section 23.5 of the S1C17W18 Technical Manual provides a table of the frequency tolerance of the IOSC oscillator over temperature and voltage range. The worst-case frequency error is +/-7% (70,000 ppm) over -40 to 85°C and VDD = 1.6 to 3.6V.

Section 23.5 of the S1C17W18 Technical Manual also shows the frequency versus temperature graph for IOSC, and it shows that the frequency versus temperature graph of IOSC has a positive linear slope and frequency increases as temperature increases.

The frequency of IOSC can be trimmed/adjusted by the IOSCAJ[5:0] value in the CLGTRIM register. It can also be auto-trimmed by writing a 1 to the IOSCSTM bit of the CLGIOSC register.

1.3 OSC1 Frequency Error

OSC1 is a “tuning fork” crystal with a nominal frequency of 32.768kHz and a typical tolerance/error specification of +/- 20ppm at 25°C. The frequency versus temperature curve for a tuning fork crystal is an “inverted parabola” with the peak at 25°C (“turn-over” temperature), as shown in the figure below for a typical crystal with 0ppm error at 25°C.

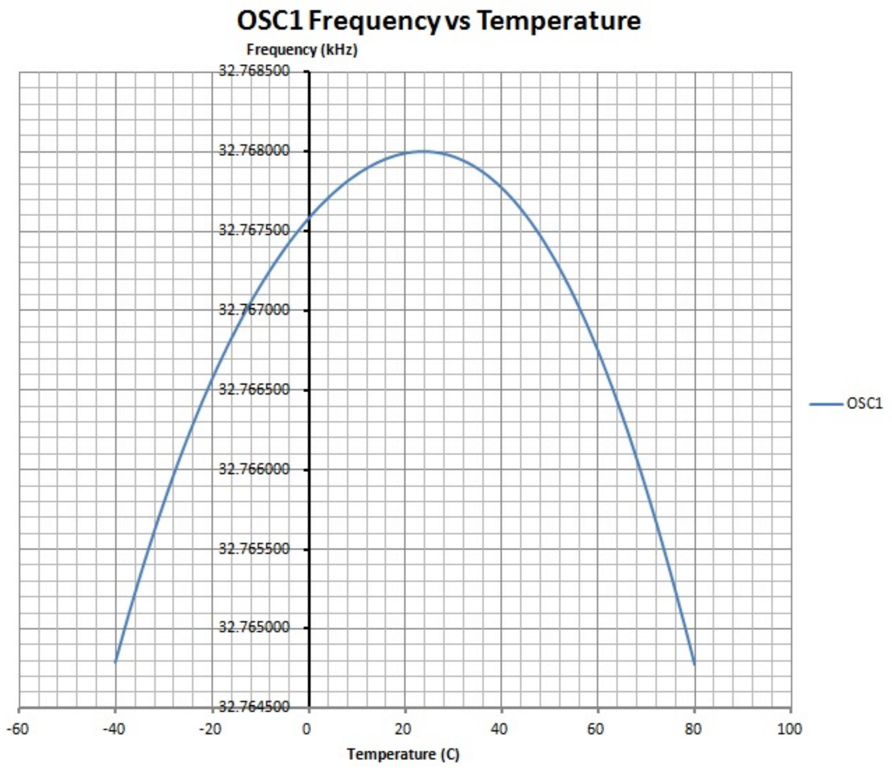


Figure 1-2: OSC1 Frequency Variation with Temperature

2. Temperature Sensor Calibration

2.1 Temperature Sensor Characteristics

The S1C17W18 has a Temperature Sensor / Voltage Reference Generator (TSRVR). The TSRVR generates a voltage which linearly depends on temperature. This voltage is read by the internal 12-bit A/D converter, which then generates a value which depends on the reference voltage supplied to the A/D converter. Figure 21.1.1 of the S1C17W18 Technical Manual shows a diagram of the TSRVR.

The voltage versus temperature graph of the temperature sensor is shown in Section 23.15 of the S1C17W18 Technical Manual. As shown in the graph, the relationship between the voltage generated and the temperature is linear. The value of the slope is the voltage-temperature coefficient, and is designated by $\Delta V_{TEMP.}$ The typical value of the voltage-temperature coefficient is 3.6 mV/°C. Using a known voltage, $V_{TREF,}$ at a reference temperature T_{REF}, the temperature can be calculated from the voltages generated by the temperature sensor using the following formula:

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} \qquad (Equation\ 2-1)$$

where T_{SEN} is the calculated temperature of the environment (in °C), and V_{TSEN} is the voltage generated by the temperature sensor at temperature T_{SEN} (in volts).

The temperature sensor voltage, V_T, can be calculated using the following equation:

$$V_T = \frac{ADC}{4096} * V_{REFA} \qquad (Equation\ 2-2)$$

where V_T is the voltage generated at a temperature T, ADC is the value generated by the A/D converter at the temperature T, and V_{REFA} is the reference voltage supplied to the A/D converter. The voltage generated at a room temperature of 25 °C is typically 1.07 V, with a range between 1.04 V and 1.11V. The TSRVR can supply a reference voltage to the internal A/D converter by setting the TSRVR*n*VCTL.VREFAMD[1:0] bits according to the following table:

Table 2-1. TSRVR Reference Voltages

TSRVR*n*VCTL.VREFAMD[1:0]	Vref Voltage
0x3	2.5 V
0x2	2.0 V
0x1	V_{DD}
0x0	Hi-Z (An external voltage can be applied)

As shown in the example calculations section, the operating temperature range from -40 °C to 85 °C generates a voltage range from ~0.830 V to ~1.290 V, assuming the voltage-temperature coefficient is the typical value of 3.6mV/°C. As will also be shown in the example calculations section, the temperature sensor can detect changes in temperature to a minimum of around 0.3 °C with a Vref of 2.0V.

Since the S1C17W18 reads ADC values instead of voltages, the two equations above can be combined to get the following equation which calculates the temperature (T_{SEN}) given the current ADC reading (ADC_{SEN}), a reference ADC reading (ADC_{REF}), and a reference temperature (T_{REF}):

$$T_{SEN} = \frac{(ADC_{TSEN} - ADC_{TREF})}{\Delta ADC_{TEMP}} + T_{REF} \qquad (Equation\ 2-3)$$

where ΔADC_{TEMP} is the ADC-temperature coefficient (slope of ADC versus temperature). The ΔADC_{TEMP} value is calculated from the following equation:

$$\Delta ADC_{TEMP} = \frac{(\frac{\Delta V_{TEMP}}{1000})}{Vref} * 4096 \qquad (Equation\ 2-4)$$

The following table shows the expected ΔADC_{TEMP} values for two Vref voltages given the typical voltage-temperature coefficient of 3.6 mV/°C and a nominal voltage of 1.07V at 25°C :

Table 2-2. Nominal ΔADC_{TEMP} Values

Vref	ΔADC_{TEMP}
2.0V	7.3728
2.5V	5.89824

If integer multiplication/division is used to calculate the temperature, the equation for T_{SEN} can be written as

$$\boldsymbol{T_{SEN} = \frac{(ADC_{TSEN} - ADC_{TREF})}{M} * 65536 + T_{REF}} \qquad (Equation\ 2-5)$$

where M = ΔADC_{TEMP} * 65536.

The following table shows the expected M values for two Vref voltages:

Table 2-3. Nominal M Values (ΔADC_{TEMP} *65536)

Vref	M
2.0V	483184
2.5V	386547

The sample software for this application note uses Equation 2-5 to calculate temperature from ADC readings.

2.2 Example Calculations

As an example, assume that the voltage-temperature coefficient, ΔV_{TEMP} is exactly 3.6 mV/°C across the entire range of operating temperatures, and also that the reference voltage at a room temperature of 25 °C is 1.07 V. The possible range of voltages is shown in the following calculation:

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF}$$

$$V_{TSEN} = \frac{T_{SEN} - T_{REF}}{1000} * \Delta V_{TEMP} + V_{TREF}$$

The minimum voltage is found by substituting the minimum temperature of -40 °C for T_{SEN} and the maximum voltage is found by substituting the maximum temperature of 85 °C for T_{SEN}.

$$V_{TSEN,MIN} = \frac{(-40\ °C) - 25\ °C}{1000} * 3.6\frac{mV}{°C} + 1.07\ V = 0.836\ V$$

$$V_{TSEN,MAX} = \frac{(85\ °C) - 25\ °C}{1000} * 3.6\frac{mV}{°C} + 1.07\ V = 1.286\ V$$

Therefore, the range of voltages can be expected to be between roughly 0.83 V and 1.29 V. Due to the variation of the voltage-temperature coefficient and the voltage at 25 °C across devices, these values may vary slightly.

To determine the minimum temperature difference that the temperature sensor can detect, assume again that the voltage-temperature coefficient is 3.6 mV/°C across the entire temperature range, the calibration voltage at room temperature is 1.07 V, and the reference voltage being supplied to the A/D converter is 2.0 V. The ADC value for room temperature is then found as follows:

$$V_T = \frac{ADC}{4096} * \ V_{REFA}$$

$$ADC = \frac{4096}{V_{REFA}} * V_T = \frac{4096}{2.0\ V} * 1.07\ V = 2191.36 \searrow 2191$$

The minimum change of temperature which the sensor can detect would cause the ADC value to increase or decrease by a value of 1. Using this new ADC value, the corresponding temperature can be calculated as follows:

$$V_T = \frac{ADC}{4096} * \ V_{REFA} \ = \frac{2190}{4096} * 2.0\ V = 1.0693\ V$$

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} = \frac{(1.0693\ V - 1.07\ V) * 1000}{3.6\ {}^{mV}\!/_{°\mathrm{C}}} + \ 25\ °\mathrm{C} = 24.8055\ °\mathrm{C}$$

Because the original ADC value was rounded down to 2191 from 2191.36, the temperature found from using the ADC value of 2191 will not be fully accurate, as some error was introduced when the rounding was introduced. To minimize this, the temperature corresponding to the ADC value of 2192 will also be calculated, and the average temperature difference will be taken.

$$V_T = \frac{ADC}{4096} * \ V_{REFA} \ = \frac{2192}{4096} * 2.0\ V = 1.0703\ V$$

$$T_{SEN} = \frac{(V_{TSEN} - V_{TREF}) * 1000}{\Delta V_{TEMP}} + T_{REF} = \frac{(1.0703\ V - 1.07\ V) * 1000}{3.6\ {}^{mV}\!/_{°\mathrm{C}}} + \ 25\ °\mathrm{C} = 25.0833\ °\mathrm{C}$$

$$\Delta T_1 = 25\ °\mathrm{C} - 24.8055\ °\mathrm{C} = 0.1945\ °\mathrm{C}$$

$$\Delta T_2 = 25.0833\ °\mathrm{C} - 25\ °\mathrm{C} = 0.0833\ °\mathrm{C}$$

$$\Delta T_{AVG} = \frac{0.1945\ °\mathrm{C} + \ 0.0833\ °\mathrm{C}}{2} = 0.1389\ °\mathrm{C}$$

Therefore, it is reasonable to say that the temperature sensor can detect changes of temperature within ~.2 °C. Using a different reference voltage would give different values for the ADC values; however the change in temperature would not be different to any reasonable degree.

2.3 Calibration Methods

Due to IC manufacturing tolerance/variation, the **M** and $\mathbf{ADC_{REF}}$ at $\mathbf{T_{REF}}$ will vary from one unit to the next. Therefore, to get higher accuracy temperature measurement, calibration of the temperature sensor is needed. The key parameter to measure is the value of **M** (slope of ADC versus temperature). The $\mathbf{ADC_{REF}}$ at $\mathbf{T_{REF}}$ is typically recorded at room temperature (25 °C).

There are different methods to determine **M**. The most comprehensive method would be to record ADC readings for the full operating temperature range of the product at small degree increments (for example, 1-degree steps), and then perform sophisticated data analysis (such least-squares method) to determine the slope of the ADC versus temperature trend-line. A more practical/simpler method would be to just record the ADC reading at the operating temperature range extremes (lowest and highest temperature) and calculate the slope (**M**) based on these two points. Which method to use depends on trade-off between accuracy needed and practicality/cost of calibration.

The following are options for production calibration of the temperature sensor:

METHOD 1 – Full Temperature Range Characterization in Production Line
For each unit in the production line, the ADC value is recorded over the operating temperature range of the product. The slope (**M**) is calculated from the data using least squares or some other method. An $\mathbf{ADC_{REF}}$ value at $\mathbf{T_{REF}}$ is chosen. The **M**, $\mathbf{ADC_{REF}}$, and $\mathbf{T_{REF}}$ values are stored in EEPROM. During normal operation, the temperature is calculated using Equation 2-5. This method is the most accurate. However, it is the most costly with respect to product line resources (test time and equipment) and it is not practical for most products.

METHOD 2 – Fixed Set of M, $\mathbf{ADC_{REF}}$, and $\mathbf{T_{REF}}$ **Values for All Production Units**
In this method, all production line units are programmed with a fixed set of values for **M**, $\mathbf{ADC_{REF}}$, and $\mathbf{T_{REF}}$, and Equation 2-5 is used to calculate temperature in normal operation. During product development and pilot runs, the ADC values versus temperature over the operating temperature range are measured from a sample set of units and an average set of **M**, $\mathbf{ADC_{REF}}$, and $\mathbf{T_{REF}}$ values is generated which is used for all production units.

METHOD 3 – ADC Measurement at Room Temperature + Average M Value
This method uses Method 2 to determine the fixed average **M** value (slope) to use for all units, but the $\mathbf{ADC_{REF}}$ value at room temperature ($\mathbf{T_{REF}}$) is recorded for each unit and stored in EEPROM.

2.4 Example Calibration on One Unit

The following graph shows an example ADC versus temperature calibration of an SVTmini17W18 board:

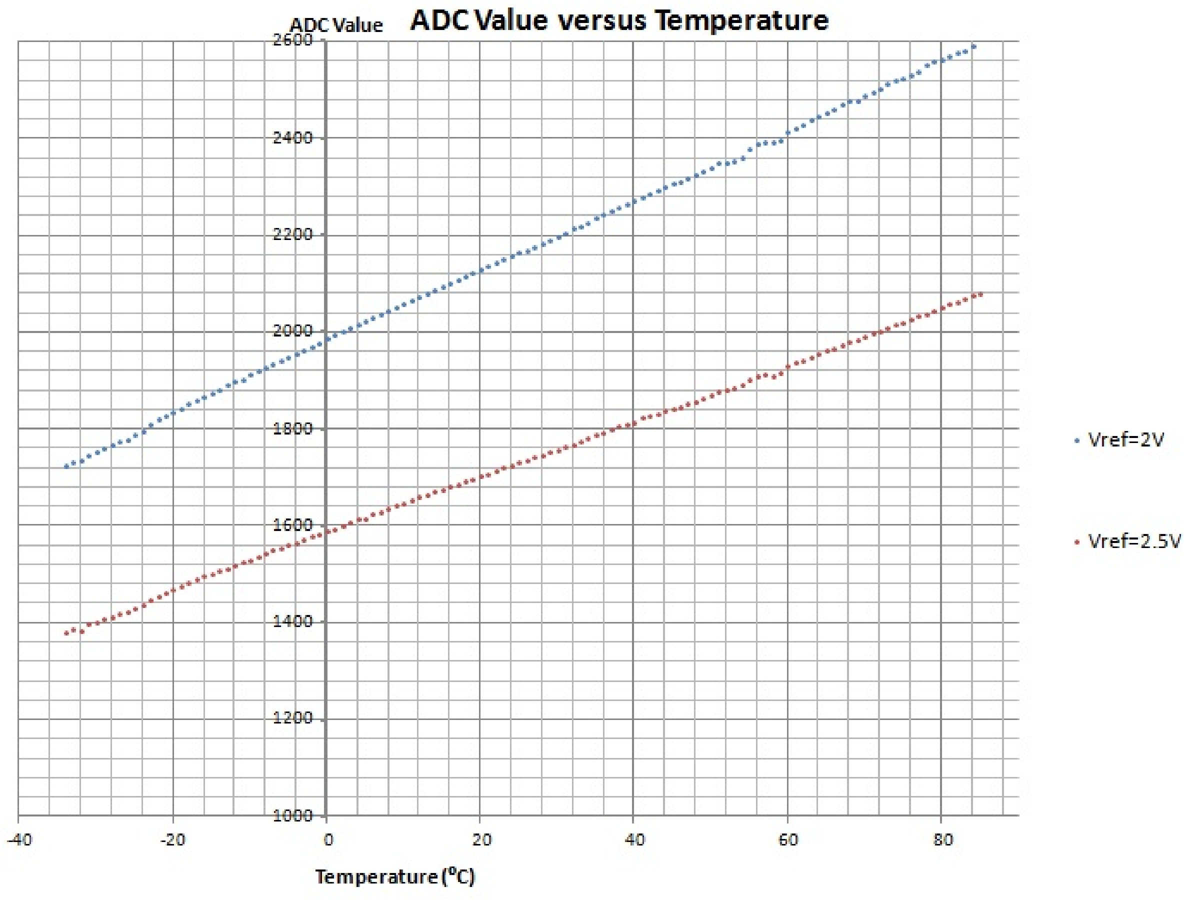


Figure 2-1: Example ADC versus Temperature Curve

The graph shows the ADC versus temperature data for Vref=2V and Vref=2.5V. The following slope values were calculated by taking the difference between the lowest and highest temperature endpoints, and the **ADC$_{REF}$** values at **T$_{REF}$** = 30°C are shown:

Table 2-4. Measured ΔADC$_{TEMP}$ and M Values

Vref	ΔADC$_{TEMP}$	M	ADC$_{REF}$ @ T$_{REF}$ = 30°C
2.0V	7.361344	482433	2200
2.5V	5.882353	385506	1759

The **M**, **ADC$_{REF}$**, and **T$_{REF}$** values were then used by the sample software to calculate temperature, and the board was retested in the temperature chamber to compare what the software reports (calculated temperature) and the actual temperature. The following table shows the results for Vref=2V and Vref=2.5V:

Table 2-5. Calculated versus Actual Temperature Comparison

	Vref = 2.0V			Vref = 2.5V		
Temperature (°C)	**ADC Reading**	**Temperature Reading (°C)**	**Temperature Error (°C)**	**ADC Reading**	**Temperature Reading (°C)**	**Temperature Error (°C)**
-35	1734	-35.0	0.0	1385	-35.3	-0.3
-30	1772	-29.9	0.1	1415	-29.8	0.2
-25	1808	-24.9	0.1	1444	-25.0	0.0
-20	1837	-20.2	-0.2	1472	-19.9	0.1
-15	1881	-14.8	0.2	1505	-15.0	0.0
-10	1913	-10.3	-0.3	1529	-10.3	-0.3
-5	1952	-5.2	-0.2	1560	-5.0	0.0
0	1990	0.2	0.2	1590	0.0	0.0
5	2025	5.1	0.1	1619	5.0	0.0
10	2066	10.2	0.2	1647	9.8	-0.2
15	2098	14.6	-0.4	1677	15.0	0.0
20	2135	19.9	-0.1	1705	19.8	-0.2
25	2164	24.8	-0.2	1727	24.8	-0.2
30	2200	30.0	0.0	1759	29.8	-0.2
35	2239	35.2	0.2	1789	35.0	0.0
40	2274	40.2	0.2	1817	39.8	-0.2
45	2312	45.2	0.2	1843	45.1	0.1
50	2343	49.9	-0.1	1873	49.9	-0.1
55	2383	55.1	0.1	1903	55.2	0.2
60	2416	60.0	0.0	1931	60.0	0.0
65	2448	65.1	0.1	1958	64.9	-0.1
70	2489	70.0	0.0	1991	70.1	0.1
75	2526	74.9	-0.1	2019	75.1	0.1
80	2564	80.0	0.0	2051	80.0	0.0
85	2601	85.0	0.0	2079	85.0	0.0

Conclusion from Example Calibration of One Unit

The example calibration shows that calculated temperature can be within +/- 0.5 °C accuracy for Vref=2V and Vref=2.5V using **M** values which are determined just based on the end points of the operating temperature range. So, perhaps calibration Method 2 and Method 3 are adequate to get fairly accurate temperature measurement. It may also be good enough to only take ADC reading at 3 temperature points around room temperature (for example, at 20°C, 24°C, and 28°C) and derive the **M**, $\mathbf{ADC_{REF}}$, and $\mathbf{T_{REF}}$ values to use for temperature calculation.

3. Frequency Error Compensation

Each clock source has its own options and algorithms for frequency error compensation as described in this section.

3.1 OSC3 Frequency Error Compensation

3.1.1 OSC3 Frequency Characteristics and Adjustment

The OSC3 internal oscillator frequency can be trimmed/adjusted by writing to the CLGTRIM.OSC3AJ[4:0] register. The following set of graphs show an example set of OSC3 frequency versus temperature for the different OSC3 frequencies (250kHz, 384kHz, 500kHz, 1MHz, 2MHz, 4MHz) at minimum and maximum CLGTRIM.OSC3AJ[4:0] values:

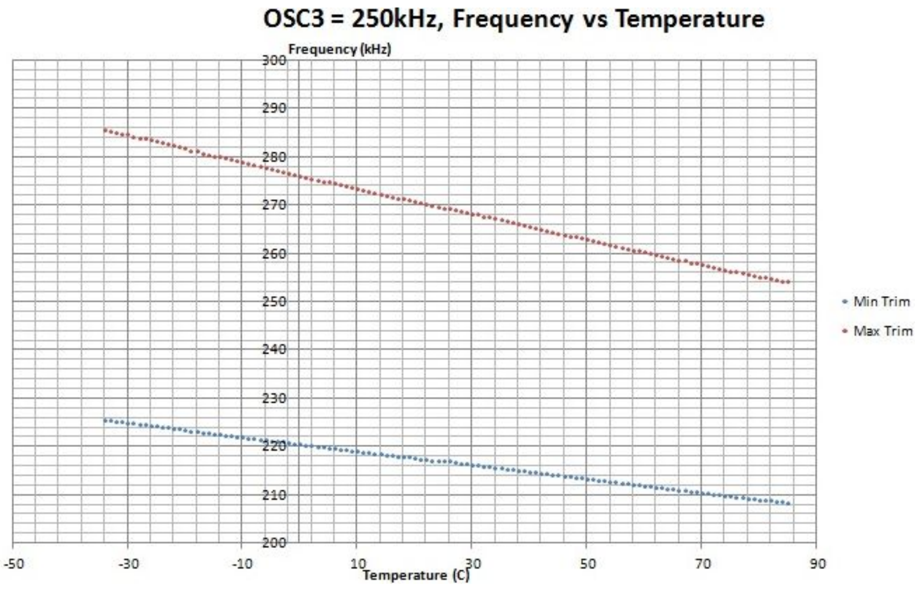


Figure 3-1: OSC3 – 250kHz, Frequency vs Temperature with Min and Max Trim Values

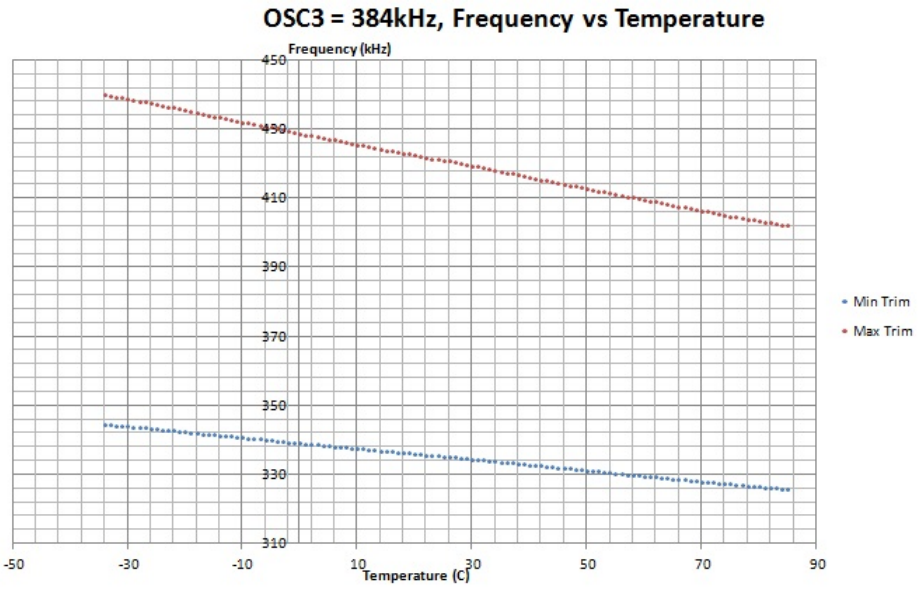


Figure 3-2 OSC3 – 384kHz, Frequency vs Temperature with Min and Max Trim Values

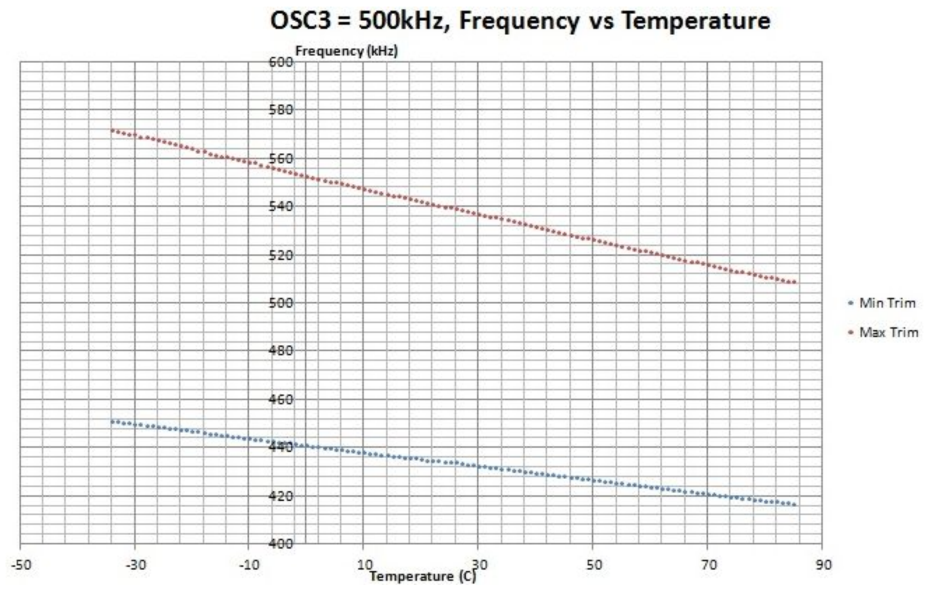


Figure 3-3 OSC3 – 500kHz, Frequency vs Temperature with Min and Max Trim Values

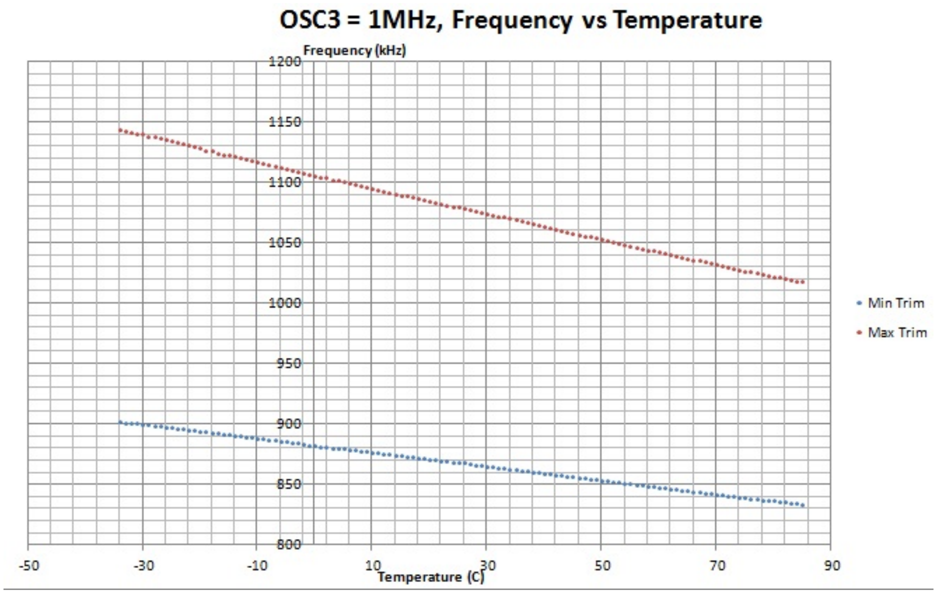


Figure 3-4 OSC3 – 1MHz, Frequency vs Temperature with Min and Max Trim Values

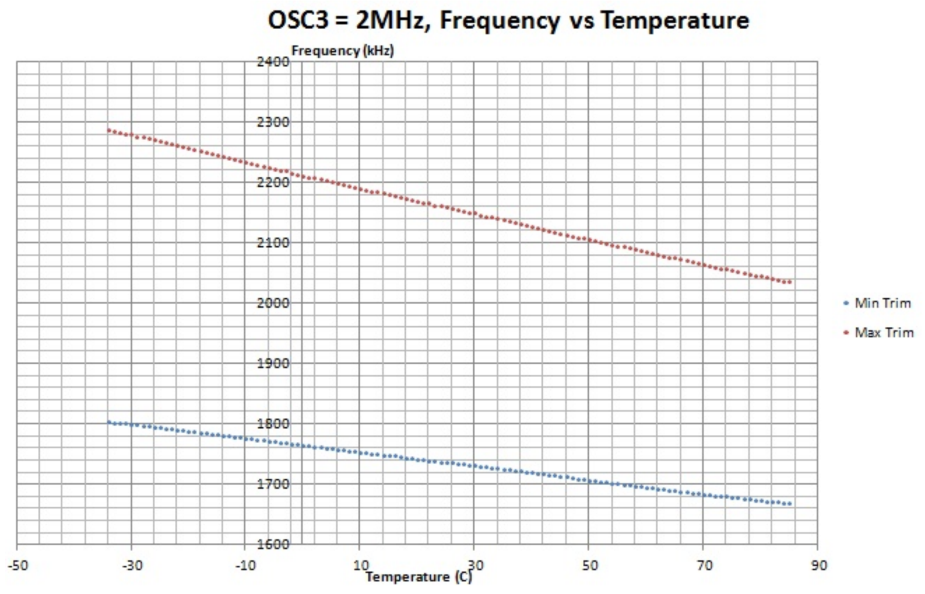


Figure 3-5 OSC3 – 2MHz, Frequency vs Temperature with Min and Max Trim Values

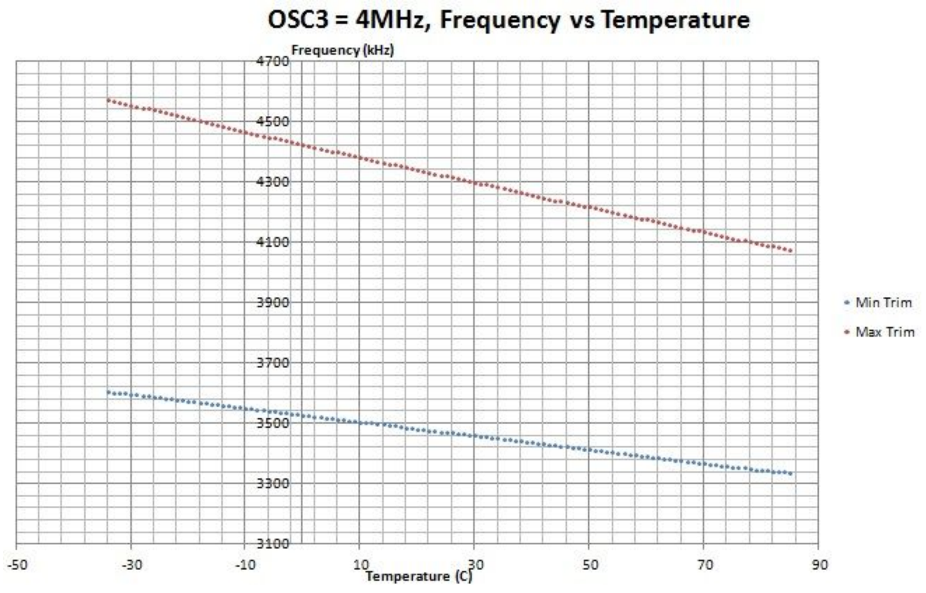


Figure 3-6 OSC3 – 4MHz, Frequency vs Temperature with Min and Max Trim Values

The frequency of OSC3 can be measured on port P16 of the S1C17W18 by programming it to output FOUT and select OSC3 as the FOUT source.

The graphs show that the CLGTRIM.OSC3AJ[4:0] register and OSC3 trimming circuit has enough range to be able to adjust the OSC3 to the nominal frequency over the whole operating temperature range.

3.1.2 Software Trimming of OSC3 Using OSC1

This software compensation method measures the OSC3 frequency using OSC1 as the clock source and adjusts the CLGTRIM.OSC3AJ[4:0] trim value in an iterative loop until the frequency error from nominal frequency is minimized. The higher accuracy OSC1 clock (<300ppm or <0.03% frequency error) is used as a reference clock to trim the lower accuracy OSC3 (up to 10% frequency error) to a higher accuracy (~1%).

In this method, two T16 timers are used. One timer is clocked by OSC1 and the other is clocked by OSC3. The OSC1 timer is programmed to count *m* clock cycles and interrupt after *m* clock cycles has elapsed. Both timers are started (from 0). When the OSC1 interrupt occurs, the OSC3 timer is stopped and the value of the counter, *n*, is read. The frequency of OSC3 can be calculated from *m*, *n*, and the frequency of OSC1 (32.768kHz nominal) using the following equation:

$$Fosc3 = \frac{(\boldsymbol{n} * Fosc1)}{\boldsymbol{m}} \qquad (Equation\ 3-1)$$

The software compares the measured Fosc3 to the nominal frequency, adjusts the trim value (CLGTRIM.OSC3AJ[4:0]), and repeats the frequency measurement. This iterative loop is run until the absolute value of the frequency error is minimized.

The frequency measurement error due to software latency and capture error can be lumped/factored into a "*nerr*" value as shown in the following equation:

$$Fosc3 = \frac{\left((\boldsymbol{n} + nerr) * \ Fosc1\right)}{\boldsymbol{m}} \qquad (Equation\ 3-2)$$

To minimize the effect of *nerr*, *n (and m)* must be sufficiently large enough to meet the desired measurement error (*nerr/n*) specification.

Example Calculations for m and n Values

The following table shows calculation examples for the *(nerr/n)* ratio with *nerr* = 10 (number Fosc3 clock cycles) with different *m* and *n* values for Fosc3 frequency of 250kHz:

Table 3-1. Example Calculation of *m* and *n* Values for OSC3 Frequency Measurement Using OSC1

Nominal Fosc1 freq. (Hz)	32768
nerr	10
Fosc3	250000
Fosc3/Fosc1	7.63

m	n	nerr/n
10	76.29	13.107%
100	762.94	1.311%
1000	7629.39	0.131%
8000	61035.16	0.016%

An *m* value of somewhere between 100 (~3ms measurement time) and 1000 (~30ms measurement time) would result in an *nerr/n* ratio of roughly 1%.

3.1.3 Example OSC3 Software Trimming on One Unit

The sample software was used to trim the OSC3 frequency using the method described in Section 3.1.2 on a sample SVTmini17W18 board at various temperature points from -35°C to 85°C and the OSC3 frequency was measured. The following *m* and *n* values (for frequency measurement of OSC3 using OSC1) were used for the different OSC3 frequencies:

Table 3-2. Example *m* and *n* Values Used for OSC3 Trimming for Different Nominal Frequencies

Fosc3	*m*	*n*
250kHz	8000	61035
384kHz	5000	58593
500kHz	4000	61035
1MHz	2000	61035
2MHz	1000	61035
4MHz	500	61035

The following table shows the results of the measurements:

Table 3-3. Result of OSC3 Trimming Over Temperature

	Fosc3 (kHz)						Error					
Temp (°C)	250 kHz	384 kHz	500 kHz	1 MHz	2 MHz	4 MHz	250 kHz	384 kHz	500 kHz	1 MHz	2 MHz	4 MHz
-35	248.2091	381.2728	496.2003	0.993968	1.990044	3.978941	-0.72%	-0.71%	-0.76%	-0.60%	-0.50%	-0.53%
-30	248.3115	381.1763	497.29	0.997008	1.981189	3.96682	-0.68%	-0.74%	-0.54%	-0.30%	-0.94%	-0.83%
-25	247.4197	382.3204	496.4046	0.994296	1.990685	3.953357	-1.03%	-0.44%	-0.72%	-0.57%	-0.47%	-1.17%
-20	247.6933	381.46	497.8195	0.997921	1.984484	3.969056	-0.92%	-0.66%	-0.44%	-0.21%	-0.78%	-0.77%
-15	248.4112	381.0009	496.3384	0.994126	1.990439	3.981598	-0.64%	-0.78%	-0.73%	-0.59%	-0.48%	-0.46%
-10	247.4713	382.0398	497.3162	0.996988	1.996818	3.96672	-1.01%	-0.51%	-0.54%	-0.30%	-0.16%	-0.83%
-5	248.1674	381.6902	498.7376	0.993678	1.989273	3.980129	-0.73%	-0.60%	-0.25%	-0.63%	-0.54%	-0.50%
0	247.8297	382.4103	497.2776	0.996172	1.993677	3.964099	-0.87%	-0.41%	-0.54%	-0.38%	-0.32%	-0.90%
5	247.495	381.9617	497.6373	0.997659	1.985905	3.974854	-1.00%	-0.53%	-0.47%	-0.23%	-0.70%	-0.63%
10	247.3466	380.7218	496.3263	0.994347	1.990254	3.959508	-1.06%	-0.85%	-0.73%	-0.57%	-0.49%	-1.01%
15	248.1812	381.3468	497.056	0.996759	1.989227	3.968671	-0.73%	-0.69%	-0.59%	-0.32%	-0.54%	-0.78%
20	248.0652	381.0207	498.7286	0.993477	1.987868	3.97754	-0.77%	-0.78%	-0.25%	-0.65%	-0.61%	-0.56%
25	248.2492	382.249	498.1023	0.996824	1.996932	3.969753	-0.70%	-0.46%	-0.38%	-0.32%	-0.15%	-0.76%
30	247.7843	381.8519	497.9084	0.998062	1.988705	3.952351	-0.89%	-0.56%	-0.42%	-0.19%	-0.56%	-1.19%
35	247.6091	382.7473	496.6247	0.99475	1.991045	3.965723	-0.96%	-0.33%	-0.68%	-0.53%	-0.45%	-0.86%
40	248.1637	381.3272	497.8699	0.996457	1.994292	3.966153	-0.73%	-0.70%	-0.43%	-0.35%	-0.29%	-0.85%
45	248.0597	381.0818	498.3574	0.993356	1.988863	3.977342	-0.78%	-0.76%	-0.33%	-0.66%	-0.56%	-0.57%
50	248.296	382.1071	497.3051	0.99756	1.986639	3.977203	-0.68%	-0.49%	-0.54%	-0.24%	-0.67%	-0.57%
55	248.1846	380.6994	497.8899	0.996613	1.99646	3.97005	-0.73%	-0.86%	-0.42%	-0.34%	-0.18%	-0.75%
60	247.7203	382.4157	496.9384	0.995465	1.992601	3.96304	-0.91%	-0.41%	-0.61%	-0.45%	-0.37%	-0.92%
65	248.0809	381.835	497.5432	0.995668	1.994443	3.96975	-0.77%	-0.56%	-0.49%	-0.43%	-0.28%	-0.76%
70	248.1419	381.0061	498.6667	0.993548	1.989444	3.959367	-0.74%	-0.78%	-0.27%	-0.65%	-0.53%	-1.02%
75	247.7209	381.3081	496.8563	0.99528	1.990856	3.96371	-0.91%	-0.70%	-0.63%	-0.47%	-0.46%	-0.91%
80	248.227	380.7919	497.1257	0.996705	1.990346	3.972165	-0.71%	-0.84%	-0.57%	-0.33%	-0.48%	-0.70%
85	248.2741	381.8569	496.295	0.994263	1.990977	3.962823	-0.69%	-0.56%	-0.74%	-0.57%	-0.45%	-0.93%

The results in Table 3-3 show that OSC3 can be trimmed using the algorithm described in Section 3.1.2 to select the trim value setting which is closest to the nominal frequency. The frequency accuracy of the closest trim setting depends on where the nominal frequency can lies between two trim steps. The data in Table 3-3 show that generally the closest trim settings are within approximately -1% of the nominal frequency.

3.1.4 Compensation Methods

3.1.4.1 OSC3 Trimming Using OSC1

In this method, the software-based trimming algorithm describe in Section 3.1.2 can be used to compensate for OSC3 frequency error. Afterwards, the temperature sensor can be used to check for changes in temperature and periodically trim OSC3 using the algorithm described in Section 3.1.2 to compensate for frequency error/drift as temperature changes.

3.1.4.2 OSC3 Trimming by Calibration Data

In this method, the OSC3 trim value (to get a frequency nearest to the nominal frequency) versus temperature (for all 6 frequency options for OSC3) calibration data is gathered/acquired during development to generate a look-up table for production firmware.

3.2 IOSC Frequency Error Compensation (Auto-Trim)

3.2.1 Frequency Error Correction

The IOSC internal oscillator frequency can be trimmed/adjusted by writing to the CLGTRIM.IOSCAJ[5:0] register. The S1C17W18 also has a hardware auto-trim function for IOSC which automatically trims the IOSC frequency using the higher accuracy OSC1 clock as the calibrating clock source (same trimming method described in Section 3.1.2 for OSC3). The auto-trimming is triggered by writing a 1 to CLGIOSC.IOSCTM bit. This bit stays high until auto-trimming is finished. The CLGINTF.IOSCSTAIF interrupt flag is set to 1 when auto-trimming is finished.

3.2.2 Compensation Methods

Initially on power-up, either the hardware auto-trim or the software-based trimming algorithm describe in Section 3.1.2 can be used to compensate for IOSC frequency error. Afterwards, the temperature sensor can be used to check for changes in temperature and periodically trim IOSC using either hardware auto-trim or the software-based algorithm described in Section 3.1.2 to compensate for frequency error/drift as temperature changes.

3.3 OSC1 Frequency Error Compensation

3.3.1 RTC Time Error Correction

The OSC1 clock is normally used for clocking the RTC (real-time clock) inside the S1C17W18 which keeps track of time and date, and its frequency error needs to be minimized. For example, a frequency error of 20ppm will cause the time to be off by 1.7 seconds per day, 52 seconds per month, and 10.5 minutes per year. The S1C17W18 does not have a built-in software-programmable circuit to trim/adjust the external OSC1 crystal oscillator circuit. However, the RTC for all the S1C17W00 series MCUs has a frequency error compensating logic circuit called Theoretical Regulation which effectively corrects for frequency error in OSC1.

The RTC of the S1C17W00 is clocked by a 256 Hz internal clock. This internal 256Hz clock is generated by the OSC1 clock (32.768kHz nominal) divided down by 128 (32768/128 = 256), and it is used to clock an 8-bit counter which counts from 0 to 255. When this counter reaches its maximum value of 255 it rolls over to 0 and generates the RTC1S pulse. If OSC1 has no frequency error, the RTC1S pulse will be generated at a frequency of exactly 1 Hz. However, because of manufacturer variations of the oscillation circuit components and temperature drift, the frequency of the OSC1 oscillator will deviate from its nominal frequency of 32.768 kHz and the RTC1S pulse does not occur at exactly 1-second intervals.

In order to compensate for frequency error, the Theoretical Regulation hardware, when triggered, modifies the 8-bit counter's value in order to maintain the 1 Hz signal of the RTC1S. For example, if the OSC1 oscillator was running at a frequency lower than the nominal frequency of 32.768 kHz, the 8-bit counter would reach the value of 255 at a slower frequency than 1 Hz. To correct this, the Theoretical Regulation hardware, when triggered, will increase the value of the 8-bit counter by a software-programmable value (RTCCTL.RTCTRM[6:0] register) in order to reach the value of 255 faster. The Theoretical Regulation is triggered by writing the desired correction value to the RTCCTL.RTCTRM[6:0] register. The RTCCTL.RTCTRMBSY bit indicates that the Theoretical Regulation is running and the RTCINTF.RTCTRMIF interrupt flag is set to 1 when Theoretical Regulation is finished.

The RTCCTL.RTCTRM[6:0] register value is the 2's complement representation of the Theoretical Regulation correction value, ***cv*** . The Theoretical Regulation circuit can correct the accumulated time error by (***cv*** / 256) seconds. So, <u>it is actually a time error correction circuit</u>.

To derive the equations for accumulated time error as a result of frequency error, the following terms are defined:

$\boldsymbol{T_s}$ = *nominal sampling interval (number of integer seconds) of temperature with frequency error look-up*

$\boldsymbol{f_a(i)}$ = *actual frequency at sample 'i'*

$\boldsymbol{f_0}$ = *nominal frequency of OSC1 = 32768 kHz*

$\boldsymbol{f_{err}(i)}$ = *frequency error at sample 'i',* $f_{err}(i) = f_a(i) - f_0$*, by convention we consider* $f_{err}(i)$ *is positive if* $f_a(i) > f_0$

$\boldsymbol{T_{err}(i)}$ = *time error between samples at sample 'i'*

$\boldsymbol{T_a(i)}$ = *actual time elapsed between samples at sample 'i',* $T_a(i) = T_s + T_{err}(i)$

$\boldsymbol{E_a(n)}$ = *actual elapsed time after'n' samples*

$\boldsymbol{E_0(n)}$ = *nominal elapsed time after'n' samples,* $E_0(n) = n*T_s$

$\boldsymbol{E_{err}(n)}$ =*accumulated elapsed time error after'n' samples*

The following equation for the actual elapsed time after ***n*** samples, $\boldsymbol{E_a(n)}$, can be used to derive the cumulative elapsed time error, $\boldsymbol{E_{err}(n)}$, after ***n*** samples:

$$E_a(n) = \sum_1^n Ta(i) = \sum_1^n (Ts + Terr(i)) = (n * Ts) + \sum_1^n (Terr(i))$$

$$= E_0(n) + E_{err}(n)$$

$$E_0(n) = n * T_s$$

$$\boldsymbol{E_{err}(n) = \sum_1^n (Terr(i)) \quad (Equation\ 3-3)}$$

The following equation for the actual time elapsed between samples, $\boldsymbol{T_a(i)}$, at sample *i* can be used to derive $\boldsymbol{T_{err}(i)}$ as a function of frequency error:

$$Ta(i) = Ts + Terr(i) = Ts * \left(\frac{fo}{fa(i)}\right)$$

$$Terr(i) = Ts * (\left(\frac{fo}{fa(i)}\right) - 1)$$

$$\boldsymbol{Terr(i) = Ts * \left(\frac{fo - fa(i)}{fa(i)}\right) = -Ts * \frac{fa(i) - fo}{fa(i)} = -Ts * \frac{ferr(i)}{fo + ferr(i)} \quad (Equation\ 3-4)}$$

Substituting Equation 3-4 into Equation 3-3 gives the following equation for the cumulative elapsed time error after ***n*** samples:

$$\boldsymbol{E_{err}(n) = \sum_1^n (Terr(i)) = -Ts * \sum_1^n \left(\frac{ferr(i)}{fo+ferr(i)}\right) \quad (Equation\ 3-5)}$$

The sign of the correction value, ***cv***, is the same as the sign of the accumulated elapse time error. If the clock frequency is lower than the nominal frequency (slower clock), it means $\boldsymbol{f_{err}(i)}$ will be negative, so the time error will be positive and the correction value, ***cv***, will also be positive to bump up the 256-cycle 1-second counter (speed it up). If the clock frequency is higher than the nominal frequency (faster clock), it means $\boldsymbol{f_{err}(i)}$ will be positive, so the time error will be negative and the correction value, ***cv***, will be also be negative to bump down the 256-cycle 1-second counter (slow it down).

The correction value at sample ***n***, ***cv(n)***, can be determined from the accumulated elapsed time error by the following equation:

$$\left(\frac{cv(n)}{256}\right) = Eerr(t) = -Ts * \sum_1^n \left(\frac{ferr(i)}{fo + ferr(i)}\right)$$

$$cv(n) = -256 * Ts * \sum_1^n \left(\frac{ferr(i)}{fo + ferr(i)}\right)$$

The term inside the summation is a small fractional number and not easily tracked. For simplicity of calculation, since $\boldsymbol{f_0}$ >> $\boldsymbol{f_{err}(i)}$ ($\boldsymbol{f_0}$ is much bigger than $\boldsymbol{f_{err}(i)}$), then

$$(fo + ferr(i)) \cong fo$$

and ***cv*** can be approximated by

$$\boldsymbol{cv(n) \cong -\frac{256 * Ts}{fo} * \sum_1^n ferr(i)}$$

$$\boldsymbol{= -\frac{256 * Ts}{32768} * \sum_1^n ferr(i) = -\frac{Ts}{128} * \sum_1^n ferr(i) \quad (Equation\ 3-6)}$$

After Theoretical Regulation correction is performed, the accumulated frequency error sum, $\sum_1^n ferr(i)$, should be adjusted by adding a correction factor to reduce the elapse time error as follows:

$$\sum_{1}^{n} ferr(i) \; <== \; \sum_{1}^{n} ferr(i) \; -(-\frac{cv*32768}{256*Ts})$$

$$\sum_{1}^{n} ferr(i) \; <== \; \sum_{1}^{n} ferr(i) \; + \frac{\boldsymbol{cv*128}}{\boldsymbol{Ts}} \quad (\boldsymbol{Equation\ 3-7})$$

In order to perform temperature-compensated time error compensation using Theoretical Regulation correction, calibration is needed during development and/or production to generate a lookup table of frequency error versus temperature values which are needed for the elapsed time error calculation.

3.3.2 Temperature-Compensated RTC Correction Method

A recommend algorithm for performing temperature-compensated time error correction using Theoretical Regulation is described as follows:

1) Decide on the sampling interval/period, T_s, to use. A longer sampling period means less power consumed, but it also means that the software might not track frequency error fluctuations as accurately compared to shorter sampling period. Therefore, it is a trade-off between power consumption and accuracy of tracking frequency fluctuation due to temperature changes.

2) Create three variables to track current frequency error, previous frequency error, and (accumulated) frequency error sum: *freqerr*, *prev_freqerr*, and *freqerrsum*.

3) Initially do the following:
   a. Set *freqerrsum* to 0.
   b. Read the temperature.
   c. Use the lookup table to get a frequency error value from the temperature reading.
   d. Assign this value to *prev_freqerr*.

4) Program the RTC to generate an interrupt in T_s seconds. This is the sampling period/interval.

5) When RTC interrupt occurs,
   a. Read the temperature.
   b. Use the lookup table to get a frequency error value from the temperature reading.
   c. Assign this value to *freqerr*. (This is the current frequency error.)
   d. *freqerr* = (*freqerr* + *prev_freqerr*) / 2
      [This takes median of the previous and current frequency error values to approximate the average frequency error during the time between the samples.]
   e. *freqerrsum* = *freqerrsum* + *freqerr* [update frequency error sum]
   f. Calculate correction value, *cv*, based on Equation 3-6.
   g. Do the following:
      - Trigger Theoretical Regulation correction.
      - Use Equation 3-7 to update *freqerrsum* after correction.
      - Go to Step 4

3.3.3 Example Time Error Correction on One Unit at Room Temperature

The sample software implements the algorithm described in Section 3.3.2 to perform temperature-compensated time error correction. A test was performed on a SVTmini17W18 board which has an OSC1 frequency error at room temperature of about +45ppm. The sampling period, $\boldsymbol{T_s}$, was set to 15 minutes (900 seconds). The OSC1 frequency versus temperature curve was measured (see Figure 3-6) and a table of frequency error versus temperature was generated and used for the temperature-compensated time error correction algorithm.

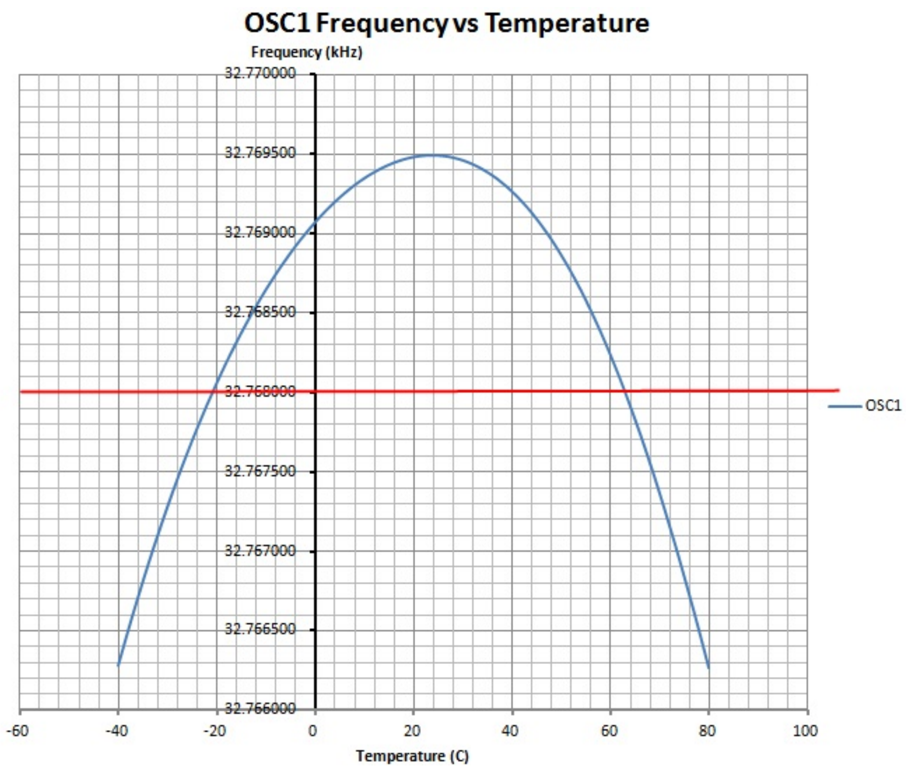


Figure 3-6: OSC1 Frequency versus Temperature Curve

For the test, the date and time displayed by the S1C17W18 sample software and the time displayed by the "time.gov" website were visually captured (by eye) and recorded for a period of 3 days. Therefore, measurement error is about +/-0.5 seconds. The following table shows the results of the test:

Table 3-4. Example Time Error Correction Results

"time.gov" (atomic clock)		S1C17W18 Time						
Date	**Time**	**Day**	**Time**	**"time.gov" Seconds Elapsed**	**S1C17W18 Seconds Elapsed**	**Time Error (Seconds)**	**Time Error Estimate without Correction (Seconds)**	**Estimated time error with correction (ppm)**
12/18/2015	15:00:29.2	0	00:01:00.0	0.0	0.0	0.0	0.0	***0.00***
12/19/2015	15:33:00.1	1	00:33:31.0	88350.9	88351.0	0.1	4.0	***1.13***
12/20/2015	23:07:29.0	2	08:08:00.0	202019.8	202020.0	0.2	9.1	***0.99***
12/21/2015	15:00:29.0	3	00:01:00.0	259199.8	259200.1	0.3	11.7	***1.16***

Without correction, the estimated time error for +45ppm frequency error after 3 days is about 11.7 seconds. With correction, the time error is less than one second and ppm error is around +1ppm.

To perform a more stringent/stressful test of the time error correction algorithm, the unit under test should be placed in a temperature chamber and put through cycles of increasing and decreasing temperature. Data should also be gathered for different sampling period settings to confirm that longer sampling periods result in higher overall ppm error after correction compared to shorter sampling periods.

3.3.4 OSC1 Frequency Error Calibration Methods

In order to perform temperature-compensated time error correction using the recommended algorithm described in Section 3.3.2, a table of frequency error versus temperature is needed. Calibration is required to generate this table.

METHOD 1 – Full Temperature Range Characterization in Production Line
For each unit in the production line, the frequency error is measured over the operating temperature range of the product and a set of frequency error values versus temperature is generated and stored in EEPROM.

This method accounts for both manufacturing tolerance of the crystal oscillator circuit components and temperature drift, and it is the most accurate. However, it is the most costly with respect to product line resources (test time and equipment) and it is not practical for most products.

METHOD 2 – Fixed Set of Frequency Error Values for All Product Units
In this method, all production line units are programmed with a fixed look-up table of frequency error values versus temperature (over the operating temperature range). The set of frequency error values versus temperature are generated based on the average of measurements of frequency error values over temperature during product development and pilot runs on a sample set of units.

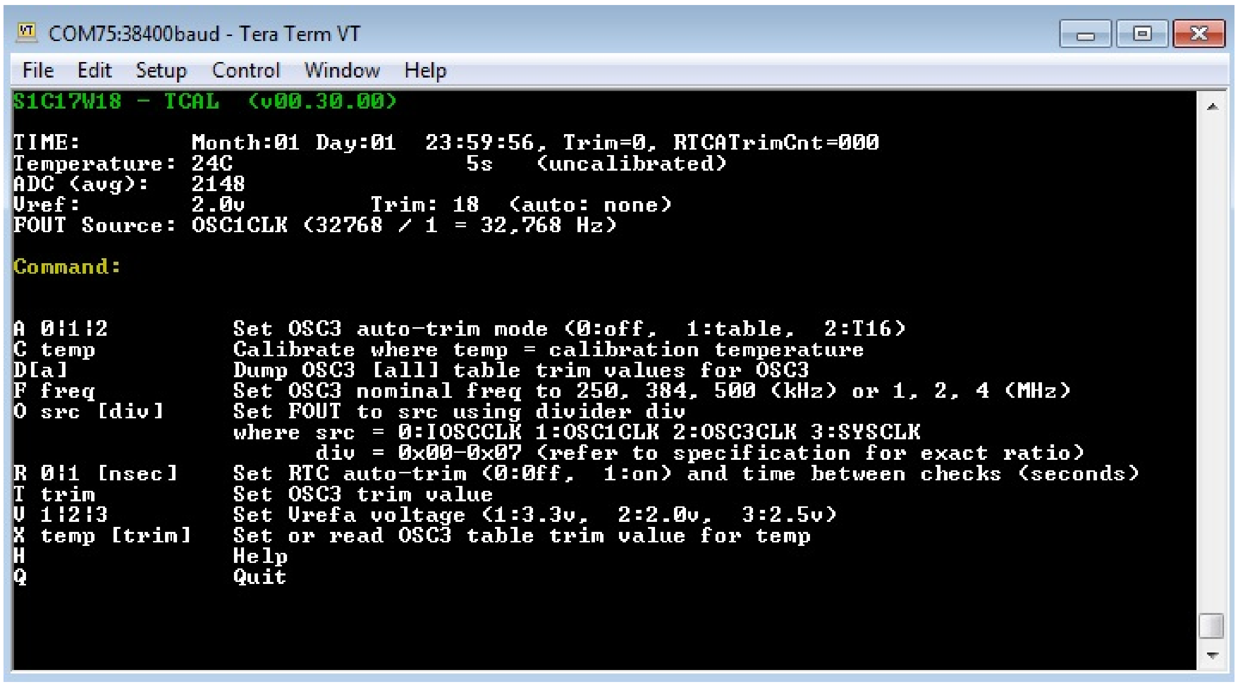
METHOD 3 – Method2 + Room Temperature Offset
In this method, each unit’s frequency error at room temperature is measured in the production line. The difference between the room temperature frequency error value from the look-up table from Method 2 and the measured frequency error is used as an offset for the rest of the fixed set of frequency error values derived from Method 2. This method assumes that the frequency error versus temperature curve shape is the same for all units and the variation between devices shifts the curve up or down.

4. Sample Software

4.1 Operations

The companion sample software for this application note has been developed in order to demonstrate and facilitate temperature calibration and frequency error compensation. The board can be connected to a PC over UART using an external serial board such as the UM232H, which is USB to serial module. The UART interface is set to 38400bps, no parity, 1 Start bit, and 1 Stop bit. Port P32 of the S1C17W18 is used as the USIN (PC->S1C17W18) input and port P33 is used as the USOUT (S1C17W18->PC) output of the UART. A terminal emulation program such as TeraTerm can be used on the PC to allow the user to interact with the sample software.

The sample software displays/updates the current month/day/time, RTC compensation information, ADC reading, and calculated temperature based on the ADC reading every second. The following shows the TeraTerm display generated by the sample software.


COM75:38400baud - Tera Term VT
File Edit Setup Control Window Help

```
S1C17W18 - TCAL  (v00.30.00)

TIME:          Month:01 Day:01  23:59:56, Trim=0, RTCATrimCnt=000
Temperature: 24C                  5s   (uncalibrated)
ADC (avg):    2148
Vref:         2.0v          Trim: 18  (auto: none)
FOUT Source: OSC1CLK (32768 / 1 = 32,768 Hz)

Command:


A 0|1|2          Set OSC3 auto-trim mode (0:off,  1:table,  2:T16)
C temp           Calibrate where temp = calibration temperature
D[a]             Dump OSC3 [all] table trim values for OSC3
F freq           Set OSC3 nominal freq to 250, 384, 500 (kHz) or 1, 2, 4 (MHz)
O src [div]      Set FOUT to src using divider div
                 where src = 0:IOSCCLK 1:OSC1CLK 2:OSC3CLK 3:SYSCLK
                       div = 0x00-0x07 (refer to specification for exact ratio)
R 0|1 [nsec]     Set RTC auto-trim (0:0ff,  1:on) and time between checks (seconds)
T trim           Set OSC3 trim value
V 1|2|3          Set Vrefa voltage (1:3.3v,  2:2.0v,  3:2.5v)
X temp [trim]    Set or read OSC3 table trim value for temp
H                Help
Q                Quit
```

Figure 4-1: Sample Software Display Output

One power up, the default RTC month is 01 and the default RTC day is 01. The sample software initializes the RTC time to 23:59:51.

The “Trim=x” status shows whether or not RTC time error correction algorithm is enabled/active. A ‘1’ means it is active and a ‘0’ means it is disabled. When RTC time error correction is active, the “RTCATrimCnt=xxx” shows how many Theoretical Regulation corrections have been made.

The “ADC (avg): xxxx” shows the current temperature sensor ADC reading. To filter the ADC reading for noise, the sample software routine takes 16 ADC readings and calculates the average.

The temperature is calculated from the average ADC reading and displayed as an integer.

The clock source for the UART interface is the IOSC oscillator. A hardware auto-trim of the IOSC oscillator is performed every 5 minutes in order to keep the baud rate accurate if the device is placed in a temperature chamber and the temperature is cycled.

The user can enter single-key commands to perform various functions. To display the list of commands available, the user can enter '?' or 'h' and press <ENTER>. Commands are processed in the "process()" function in "src/main.c".

The commands available for the user are listed below. Optional parameters are enclosed in square brackets.

Table 4-1. Available Software Commands

Command	Parameters	Description
A or a	0 | 1 | 2	Sets the OSC3 trim mode. 0: off, no trimming of OSC3 1: table, OSC3 is trimmed using a table of trim values vs. temperature 2: auto-trim, OSC3 is trimmed using OSC1 as reference as described in Section 3.1.2.
C or c	*temp*	Sets the current reference temperature to the value of *temp*. This is the **Tref** value in Equation 2-5. The current ADC reading is set as the **ADCref**.
D or d	-	Dumps all the trim values in the table of trim values versus temperature for OSC3.
F or f	*freq*	Selects the OSC3 frequency. Possible values for 'freq' are 250 (kHz), 384 (kHz), 500 (kHz), 1 (MHz), 2 (MHz), or 4 (MHz).
O or o	*src* [*div*]	Sets the source of the frequency to be output on the FOUT pin. A divider can also be applied to the nominal frequency. Possible options for *src* are 0: IOSCCLK 1: OSC1CLK 2: OSC3CLK 3: SYSCLK The value of *div* can range from 0x00 to 0x07.
R or r	0 | 1 [*nsec*]	Enables time error correction of the RTC using the algorithm described in Section 3.3.2. 0: time error correction disabled 1: time error correction enabled The optional *nsec* value is the sampling period Δn (in seconds). The default is 900s. It can be changed to a different value using this command.
T or t	*trim*	Manually sets the trim value for OSC3 to be the value of *trim*. This

		value can range from 0 to 31.
V or v	1 | 2 | 3	Sets the reference voltage for the ADC module. Possible values are 1: VDD 2: 2.0V 3: 2.5V
X	*temp* [*trim*]	Sets or reads the trim value stored in the OSC3 lookup table at the given temperature.
H or ?	-	Displays the help menu.
Q	-	Quit.

4.1.1 Temperature Calculation

The sample software displays the calculated temperature based on Equation 2-5 and Table 2-4. The "readTempTsrvr()" function in "src/temptsrvr.c" is the function which reads the ADC and performs this calculation. The user can change the current reference ADC and temperature by using the 'c' command. The sample software uses the temperature value specified by the "c" command and the current ADC reading as the new reference pair.

The reference voltage for the temperature sensor ADC measurement can be set by the 'v' command to select internal 2V, internal 2.5V, or external VDD.

For the temperature sensor calibration, the unit is placed in a temperature chamber and ADC readings are recorded over the temperature range. A slope of the ADC readings versus temperature can then be obtained.

4.1.2 FOUT Pin Selection

The sample software allows the user to specify the clock signal to output on the FOUT pin (port P16). The'o' command is used to select which clock signal (and divide-down value) to output on the FOUT pin. The "f" command is used to select the frequency of the internal OSC3 oscillator. An oscilloscope can be connected to the FOUT pin to measure clock frequencies over a temperature range.

4.1.3 OSC3 Trimming

The sample software allows the user to select the trim mode for OSC3 using the 'a' command. For manual trim ("a 0" command), the 't' command is used to specify the trim value. For trimming by calibration table ("a 1" command), the "autoOSC3trimTemp()" function in "src/main.c" handles the reading of trim values from a table and programming the trim register. For auto-trimming by using OSC1 as frequency reference ("a 2" command), as described in Section 3.1.2, the "SWTrimOSC3()" function in "src/tcomp.c" is used.

4.1.4 RTC Time Error Correction

Theoretical Regulation correction of the RTC (as described in Section 3.3.2) is enabled/disabled by the 'r' command. The optional *nsec* value is the sampling period $\boldsymbol{T}_s$ (in seconds). The default is 900s. It can be changed to a different value using the 'r'command.

The "tcompSetRtcAlarm()" and "RTCATrim()" functions in "src/tcomp.c" are used to perform the Theoretcial Regulation correction algorithm described in Section 3.3.2.

4.2 Software Modules

The accompanying sample software for this application note has the following modules:

adc12a.c – contains routines for the ADC12A peripheral for analog-to-digital conversion
boot.c – boot routines
clg.c – clock generator routines
init.c – system initialization routines
rtca.c – real-time clock routines
tsrvr.c – temperature sensor related routines
uart.c – UART routines

temptsrvr.c – routines related to temperature reading using the TSRVR temperature sensor
tcomp.c – temperature compensation related routines
main.c – main application, user interface, calibration, and measurement routines

4.2.1 "temptsrvr.c"

This module contains routines related to reading temperature using the built-in TSRVR temperature sensor of the S1C17W18. It uses functions from "adc12a.c" and "tsrvr.c" and it contains the interrupt handler routine for the ADC12A peripheral (*intAd12()*). The following routines are included:

readADC() – reads the temperature sensor voltage using the ADC12A peripheral
initTempTsrvr() – initializes the temperature reading module
readTempTsrvr() – reads the temperature using the TSRVR temperature sensor

The "*initTempTsrvr()*" should be called once in "main()" to initialize the temperature reading module. The parameter to pass to this routine is the selection of VREF voltage to use for the temperature sensor. This routine configures the ADC12A clock source (T16 CH3), the GPIO port for the temperature sensor, and the reference voltage selection for the temperature sensor.

After initialization routine is called, the "*readTempTsrvr()*" routine can be called to read the temperature. This routine returns the temperature in degrees Celsius. It calls the "*readADC()*" routine to read the temperature sensor value and then calculates the temperature using Equation 2-5. The "*readADC()*" routine takes 16 ADC samples and provides an averaged value in the global variable "adcAvg".

The *Tref[]*, *ADCref[]*, and *M[]* arrays used for the temperature calculation (see Equation 2-5 and Table 2-4) are initialized to default values based on measured data on a single test unit. The default values can be changed to other values based on better calibration data.

4.2.2 "tcomp.c"

This module contains routines to perform temperature compensation for OSC1, OSC3, and IOSC. It includes the interrupt handler for T16 CH0 (*intT16Ch0()*), which is used for OSC3 software-trimming based on OSC1 as time reference, and the following public routines:

HWTrimIOSC() – performs hardware auto-trim of IOSC oscillator
SWTrimOSC3() – performs software trim of OSC3 oscillator using OSC1 as time reference
tcompSetRtcAlarm() – programs the RTC alarm to interrupt *alarmsec* seconds from now
initRTCATrim() - initializes variables for RTC Theoretical Regulation trimming
RTCATrim() – performs RTC Theoretical Regulation trimming

The "*HWTrimOSC()*" and "*SWTrimOSC3()*" routines can be called as required to auto-trim IOSC and OSC3, respectively. A typical application would call these two routines whenever it detects a significant change in temperature.

SWTrimOSC3() Routine

The "*targetOSC1OSC3[6][2]*" array is used for the OSC3 trimming using OSC1 as reference clock and its initialization values are taken from Table 3-2.

The "*SWTrim OSC3()*" routine performs the following tasks to adjust the OSC3 trim register value with OSC1 as reference clock to minimize frequency error of OSC3:

1) Read the current trim value and stored in "*trim_ref*".
2) Call "*countT16()*" function to measure the OSC3 frequency. The "*countT16()*" function (and "*intT16Ch0()*" interrupt handler) saves the value of the timer counter which is clocked by OSC3 in the global variable "*measuredOSC3*".
3) Calculate the difference between the "*measuredOSC3*" value and the expected value read from the "*targetOSC1OSC3[][1]*" table and store it in "*delta_ref*".
4) If "*delta_ref*" is positive, reduce the OSC3 trim value by 1; otherwise, increase the OSC3 trim value by 1. Save the new trim value in "*trim_new*".
5) Call "*countT16()*" to measure the next OSC3 frequency after trim adjustment.
6) Calculate the difference between the "*measuredOSC3*" value and the expected value read from the "*targetOSC1OSC3[][1]*" table and store it in "*delta_new*".
7) If the absolute value of "*delta_new*" is greater than the absolute value of "*delta_ref*", set the OSC3 trim register value to "*trim_ref*" and terminate; otherwise, make/set "*trim_ref*" equal to "*trim_new*" and "*delta_ref*" equal to "*delta_new*".
8) Go back to step 4).

RTC Compensation Related Routines

The "*tcompSetRtcAlarm()*" routine is used for programming the RTC alarm to schedule an interrupt. The parameter passed is the number of seconds from now for the alarm interrupt to occur. This routine is typically used to schedule the sampling interval for Theoretical Regulation trimming of the RTC.

The "*initRTCATrim()*" routine should be called to initialize variables needed by the "*RTCATrim()*" routine. The parameter passed to "*initRTCATrim()*" is the current frequency error in milliHertz ("*curr_freqerr*"). The routine initializes global variables "*freqerrsum_mHz*" to 0 and "*prev_osc1FreqErr_mHz*" to the value of "*curr_freqerr*".

The "*RTCATrim()*" routine is called to perform Theoretical Regulation trimming of the RTC. Typically, when the RTC alarm interrupt occurs, "*RTCATrim()*" is called. The parameters passed to "*RTCATrim()*" are the sampling period ("*tsamp*") and current frequency error in milliHertz ("*curr_freqerr_mHz*"). The value of the "*tsamp*" corresponds to the T_s value in Equation 3-6. The "*freqerrsum_mHz*" global variable corresponds to the summation of ***ferr(n)*** values in Equation 3-6 which is the cumulative sum of frequency errors.

The "*RTCATrim()*" routine performs the following tasks:

1) Add the current frequency error ("*curr_freqerr_mHz*") and the previous frequency error ("*prev_osc1FreqErr_mHz*") and divided by 2 to get an average frequency error.
2) Add the average frequency error to the cumulative error sum "*freqerrsum_mHz*".
3) Calculate the trim value ("*rtcTrim*" variable) needed to correct the latest calculated cumulative error "*freqerrsum_mHz*" using Equation 3-6.
4) Initiate a Theoretical Regulation adjustment and reduce the "*freqerrsum_mHz*" by the trim value amount (Equation 3-7).
5) Program the RTC alarm (call "*tcompSetRtcAlarm()*" routine) to interrupt in "*tsamp*" seconds.

4.2.3 “main.c”

This module contains the “main()” application and routines to interact with the user over the UART interface. It includes the following routines:

initClocks() – initializes clocks used by the application.

setTref() – this is called by the ‘c’ or ‘C’ command to set the current reference temperature and ADC value for the current selected VREF for the temperature sensor.

setOSC3trim() – sets the OSC3 trim value. OSC3 is turned off first, trim value set, and restarted.

getOSC3freq() – returns the current nominal frequency of OSC3 in Hertz.

getClgSystemClkFreq() – returns the current nominal system clock frequency in Hertz.

autoOSC3trimTemp() – sets the OSC3 trim register using a lookup table of trim value versus temperature whenever the temperature changes. This routine is called if OSC3 trim mode is 1 (“a 1” command) and the temperature changes.

sprintfcomma() – print a comma-separated (every 3 digits) decimal value to a string

clrMsgArea() – clears the message area

updateTemp() – updates the temperature status line

updateOSC3() - updates the OSC3 status line

updateVref() – updates the VREF status line

updateFOUT() – updates the FOUT status line

updateCmdPrompt() – prepares/restores the command prompt

displayHeader() – displays the header

updateDisplay() – full display update

displayHelp() – displays the “help” text

dumpOSC3TrimTable() – displays the table of OSC3 trim values.

nextarg() – finds the next token/argument in a string

process() – process keystrokes from user

The “*osc1FreqErr_mHz[]*” array is the lookup table of OSC1 frequency error (in milliHertz) versus temperature (-40 to 85C, in 1-degree steps) which is used for RTC trimming. It is initialized to default values which are based on calibration of one test unit of an SVTmini17W18 board. The initialization values for this table can be changed to a different set of calibration values depending on the OSC1 crystal circuit design.

Revision History

Rev. No.	Date	Page	Category	Contents
Rev 1.0	2016/10/27	All	New	
				-
				-

EPSON

International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.
214 Devcon Drive,
San Jose, CA 95112, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

EUROPE

EPSON EUROPE ELECTRONICS GmbH
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA

EPSON (CHINA) CO., LTD.
4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

SHANGHAI BRANCH
7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5577 FAX: +86-21-5423-4677

SHENZHEN BRANCH
Room 804-805, 8 Floor, Tower 2, Ali Center,No.3331
Keyuan South RD(Shenzhen bay), Nanshan District, Shenzhen 518054, CHINA
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

EPSON TAIWAN TECHNOLOGY & TRADING LTD.
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

EPSON SINGAPORE PTE., LTD.
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORP.
KOREA OFFICE
19F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

SEIKO EPSON CORP.
MICRODEVICES OPERATIONS DIVISION

Device Sales & Marketing Department
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 413349100
Issue October 2016 in JAPAN