Temperature-Compensated Oscillator Example

1. Introduction

All Silicon Labs C8051F5xx MCU devices have an internal oscillator frequency tolerance of ±0.5%, which is rated at the oscillator's average frequency. In certain applications where timing is of utmost importance, an oscillator with higher precision is required. This application note describes a technique for improving the internal oscillator accuracy. This is accomplished by comparing the device temperature to the known characteristic behavior of the oscillator, computing an oscillator frequency offset for the temperature, and compensating for that offset using oscillator calibration registers on the device.

2. Changing the Oscillator Calibration Bits

The following characterization chart of the internal oscillator's frequency versus temperature shows that the frequency of the oscillator is inversely parabolic over its operating temperature range from –40° to 125° C.

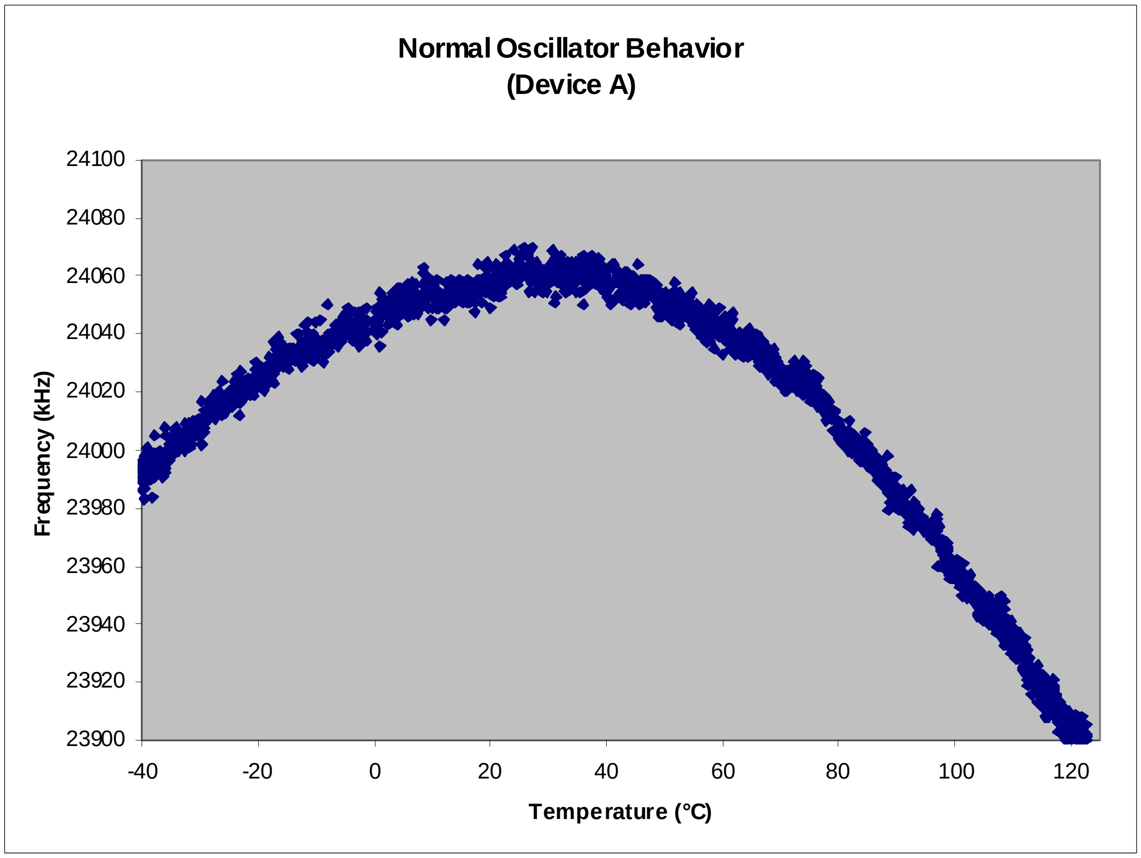


Figure 1. Normal Oscillator Behavior (Device A)

For the F50x devices, the curve in Figure 1 can be described by the following equation.

$$f(T) = f0 \times (1 + TC1 \times (T - T0) + TC2 \times (T - T0)^2)$$

In the above equation, f0 is the internal oscillator frequency at 25 °C, and T0 is 25 °C. The temperature coefficients, TC1 and TC2, are 5.0 and –0.65 ppm/°C respectively. More information about the internal oscillator can be found in Table 5.6 Internal High-Frequency Oscillator Electrical Characteristics of the C8051F50x data sheet. Using the information from Figure 1 as a guide, the device can detect if the oscillator frequency should be increased when running at a given temperature. Increasing the oscillator frequency is done by adjusting the OSCIFIN register.

The OSCIFIN register is the Internal Oscillator Fine Calibration register. During the production process of the chip, this register is calibrated to the appropriate value to produce a nominal frequency of 24 MHz. Changing the value of this register will adjust the capacitance within the oscillator circuit, thus affecting the frequency of the oscillator. Raising the value in this register by one will shorten the oscillator's period by approximately 31 ps. This means that the oscillator frequency increases linearly with changes to OSCIFIN. It is important to note that changes to this register will not be permanent. The value of the OSCIFIN register will be reset to its factory calibrated value whenever the device is reset.

By detecting the temperature of the device, software can derive the value to which the OSCIFIN register must be set in order to properly compensate for the oscillator's parabolic frequency shift. Figure 2 shows the characterization chart of the internal oscillator's speed versus temperature after such compensations are implemented.

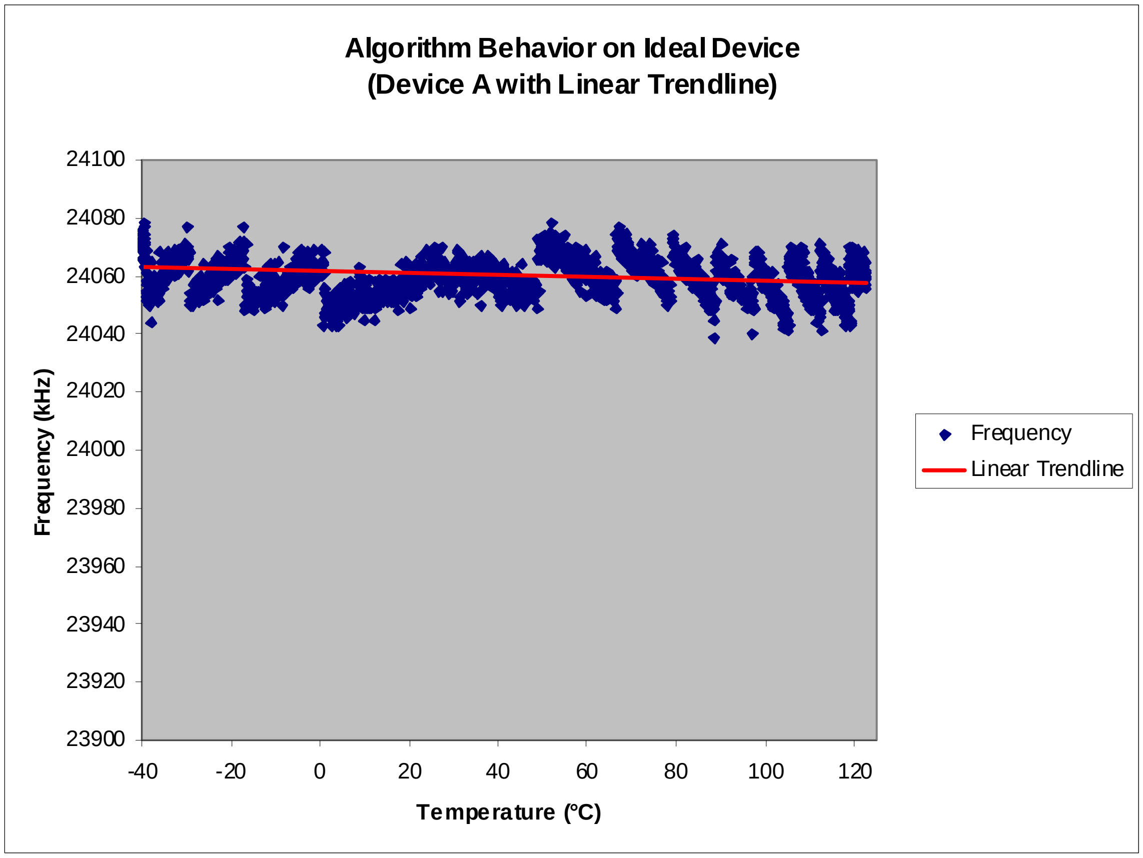


Figure 2. Algorithm Behavior on Ideal Device (Device A with Linear Trendline)

The dark blue data points in Figure 2 indicate recorded frequencies over the full operating temperature range. The red line is a linear trend line that is very near a slope of zero. The behavior displayed by the dark blue data has sharp steps in it because those are the points at which the example code algorithm is changing the value of OSCIFIN, which causes a shift in the oscillator frequency. Note that it is possible to recreate Figure 2 from Figure 1 or vice versa by applying frequency offsets to the data at known temperature points. The temperature points can be derived from the example code included with this application note.

3. Variability of Results

The calculated tolerance of the results shown in Figure 2 is 24.06 MHz ±0.083%. The example code was very successful, and the reason for this success can be seen more clearly in Figure 1. The apex of the parabola is very close to 25° C. The example code expects an oscillator with the apex at exactly 25° C because the apex point on all the C8051F50x devices is, on average, 25° C. The actual value of the apex point can be several degrees away from 25° C in either direction. The next two figures are derived from worst-case corner devices in order to exaggerate and demonstrate the behavior of the algorithm under worst-case conditions.

Figure 3 shows what happens when the device's apex temperature is below 25° C. This causes overcompensation at temperatures below 25° C and too little compensation at temperatures above 25° C. In Figure 3, the tolerance for this device is about 24.09 MHz ±0.188%.

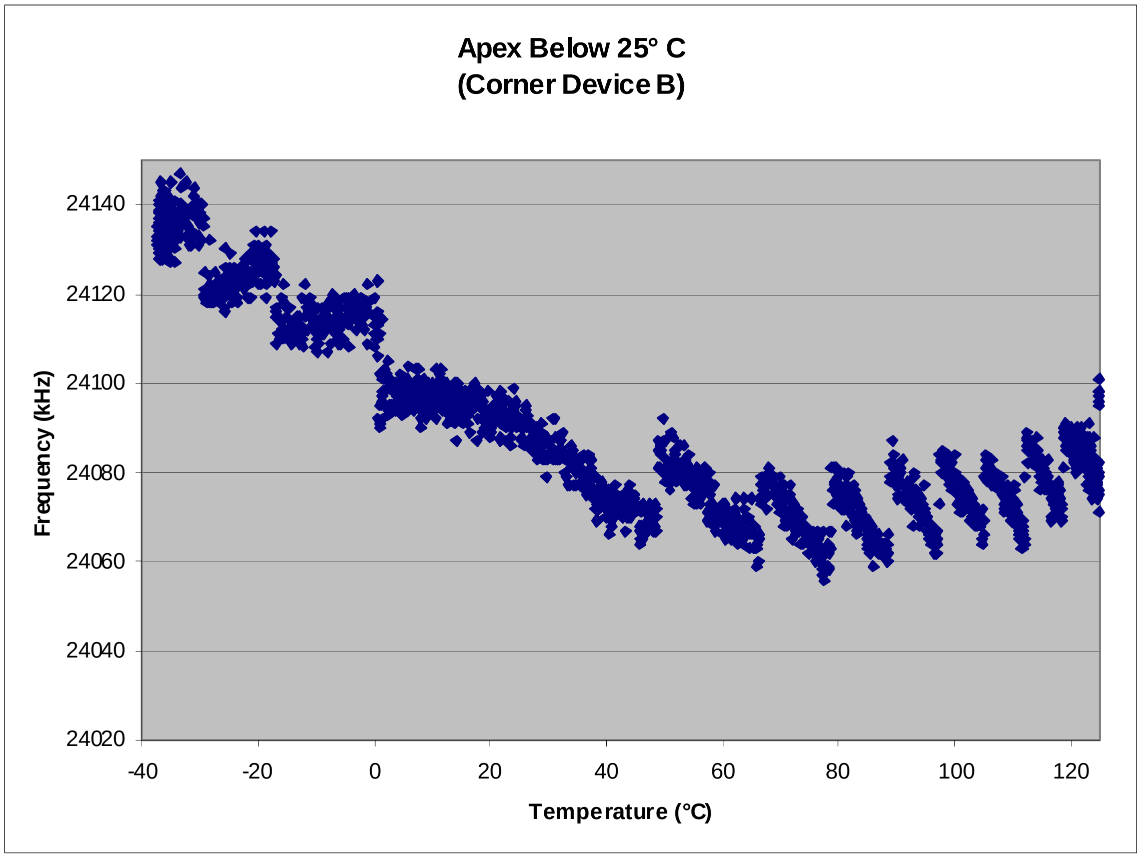


Figure 3. Apex below 25 °C (Corner Device B)

Figure 4 displays what happens if the apex of the internal oscillator versus temperature behavior is above the expected 25° C. The algorithm overcompensates above 25° C and does not compensate enough below 25° C. This results in a tolerance of about 24.06 MHz ±0.168% for this particular device.

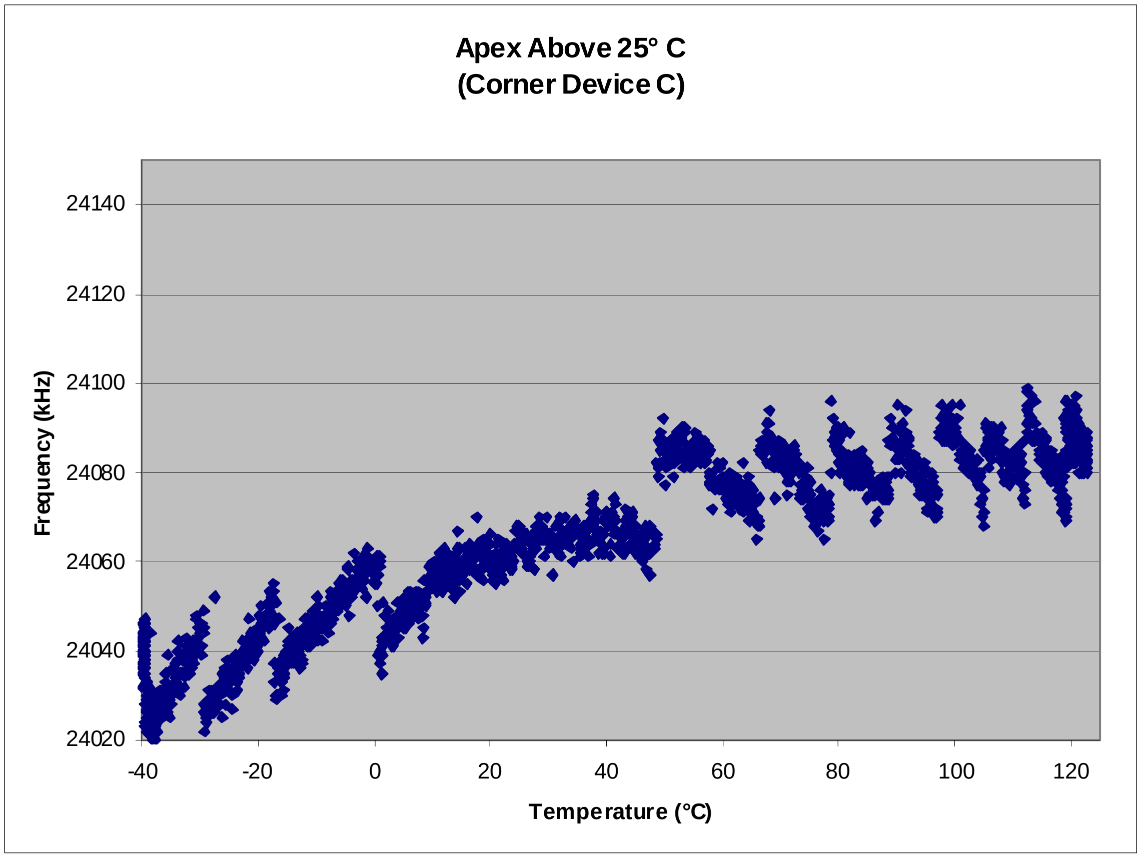


Figure 4. Apex above 25 °C (Corner Device C)

Using this algorithm for adjusting the OSCIFIN register according to what the temperature sensor reads, the internal oscillator now has a tolerance maximum of its average frequency ±0.25%. The algorithm has the potential to reduce the tolerance to below ±0.1% on ideal parts, but variation from part to part causes the tolerance to be higher on average and ±0.25% in the worst cases. This is significantly better than the standard ±0.5% tolerance on parts not running this algorithm.

4. Temperature Sensing

The temperature compensated oscillator algorithm requires using the on-chip internal temperature sensor of the device. In order to properly estimate the temperature of the device, the software will need to be given a one-point calibration value for the temperature. Through UART communication, the example software takes user input of what the current ambient temperature is. Using this input and the current value that the temperature conversion is returning, the software can calculate and store an offset value for all future temperature conversions. An incorrect temperature calibration value will give the same results as described in "3. Variability of Results" on page 3. This is because the temperature calibration affects the temperature the algorithm uses to process changes to OSCIFIN.

This application note will not go into the specifics of using the internal temperature sensor of the devices. For more information about using the temperature sensor, refer to the Temperature Sensor section of the data sheet for the particular device being used.

5. Temperature Sensitivity

In order to convert the tolerances discussed in Section 3 into terms of temperature sensitivity, the percentage values must be changed into units of ppm/°C. The following steps provide an example of how to convert ±0.25% into 25 ppm/°C on the first order.

1. Convert the percentage into a decimal number.
   0.25% = 0.0025
2. Determine the maximum temperature delta and divide by this value. The max delta for this application is from 25°C to 125°C so the delta value is 100°C.
   0.0025/100°C = .000025
3. Multiply by 1 million in order to get the value into parts per million.
   .000025 x 1000000 = 25 ppm/°C on the first order

6. Considerations for Improvements

The example code for this application note is not optimized for minimum CPU utilization, memory efficiency, or speed. The code can be improved in a number of ways, such as characterizing each device to find its true apex value, modifying the starting value of OSCIFIN to adjust the average frequency of the oscillator, and removing the temperature to Celsius conversions to free up more memory and lower CPU utilization.

As indicated by the Variability of Results section, the results of this approach are directly related to the accuracy of the apex approximation. A good way of improving the results is to characterize each device's oscillator across temperature; derive its true apex value from the results, and adjust the software to expect the measured apex value instead of the standard 25 °C. There are two different methods that can be used to characterize the device's oscillator.

One way is to have the device output its system clock to a port pin. Using a temperature-controlled environment and an oscilloscope or frequency counter, the device's frequency can be recorded across the full temperature range of the device. The second way of characterizing the oscillator is by feeding a known, accurate 1 kHz square wave into the device from a signal generator. The device's PCA input capture can be used to detect rising edges of this signal. The software then counts system clocks between rising edges and converts the counts into its frequency using the known period of the 1 kHz square wave. Using either of these methods to record the oscillator frequency over a temperature sweep will give a good estimation of the apex value for that device.

When oscillator tolerance is of utmost importance, it is good to note the operating temperatures to which the device will be exposed. Referring back to Figures 1 and 2, note that the points where the oscillator is the furthest away from its rated frequency are at the extreme ends of the temperature range. If the device will not be exposed to this full temperature range, the tolerance range can be constricted significantly.

Only the lower six bits of the OSCIFIN register are used for the oscillator calibration values. This means that a value of 0x3F (63d) is the highest value to which the OSCIFIN register can be set. The algorithm is programmed to increment OSCIFIN by up to nine units at the most extreme temperatures away from the apex value. It is important to consider what happens if the factory calibrated OSCIFIN value is greater than 54d. If the program is allowed to keep incrementing past 63d, the OSCIFIN register will loop back to zero, and the oscillator will suddenly slow down by a great amount. If the program creates a max value of 63d in the OSCIFIN register, the program will no longer properly compensate the oscillator at higher temperature deltas away from the apex.

The best solution to this problem is to read the OSCIFIN register before allowing the temperature compensation algorithm to run. If the OSCIFIN register is greater than 54d, it should be set to 54d. This will still result in a waveform that looks like Figure 2, but the results will be offset to lower values. The tolerance of the new waveform will be the same as if OSCIFIN register's starting value was not modified and the register was allowed to increase past 63d, but the average frequency of the oscillator will be slightly slower because of the offset. This is still acceptable because the algorithm will keep the oscillator's frequency from shifting too far away from the desired frequency. Note that the starting value of OSCIFIN can also be purposely modified to change the average frequency of the oscillator.

When looking at ways of including the temperature-calibrated oscillator algorithm in another application, the CPU utilization and memory usage of this algorithm can be improved. Temperature sensing is implemented by taking ADC readings from the temperature sensor and converting those ADC codes to temperature values in Celsius. By removing the conversion to Celsius temperature values and just using the raw ADC readings, the response speed of the algorithm will improve, and many long integer variables can be removed to increase memory efficiency. The main drawback of doing the temperature sensing without the temperature conversion is that debugging is more difficult because none of the temperature ADC readings will be in Celsius anymore. It is also important to note that the software will still need a one-point temperature calibration value. The best way to do this is to still do some temperature to Celsius conversions exclusively in the temperature calibration function. The temperature calibration value can also be derived using another set of software and then hard-coded into the temperature-compensated oscillator algorithm.

7. Example Code Input/Output

The remainder of this document describes the example code that accompanies this application note which is included in AN365SW.zip. The program inside the zip is called F50x_Temp_Compensated_Oscillator_Example.c. AN365SW.zip is also available for download from the following web page:
https://www.silabs.com/products/mcu/Pages/ApplicationNotes.aspx.

This section discusses the various inputs and outputs of the program. SYSCLK is output to a port pin using the PCA on P0.2 so that its frequency is available to external equipment. An oscilloscope connected to P0.2 of the MCU can measure the change in frequency resulting from modifying the OSCIFIN register.

The software uses UART at 115200 baud rate. When setting up a terminal program to interpret data from the device, make sure to set the baud rate for 115200, data bits to 8, parity to none, stop bits to 1, and flow control to none. Through UART, the code will transmit three important debugging values. The output will look like the following line:

```
T = 59.55 (C)   OSCIFIN = 21   temp_distance = 34550
```

The first value is the temperature monitored by the device. The second value is the current setting (in decimal) of the OSCIFIN register. The final value is the temperature delta from 25 °C. This unsigned value is scaled up by a factor of 1000; so, a temp_distance of 34550 means that the current temperature is 34.55 degrees away from 25 °C.

To calibrate the temperature offset, the ambient temperature of the room can be entered via the UART at any time. The input temperature value is expected to be the desired temperature multiplied by 10. If the desired temperature is 25.1 °C, an input of "251" followed by a carriage return will result in the device's temperature output changing to 25.1 °C. This feature will help calibrate out the self-heating of the device.

Note: The offset value is stored in flash; so, the most recent calibration will persist through reset.

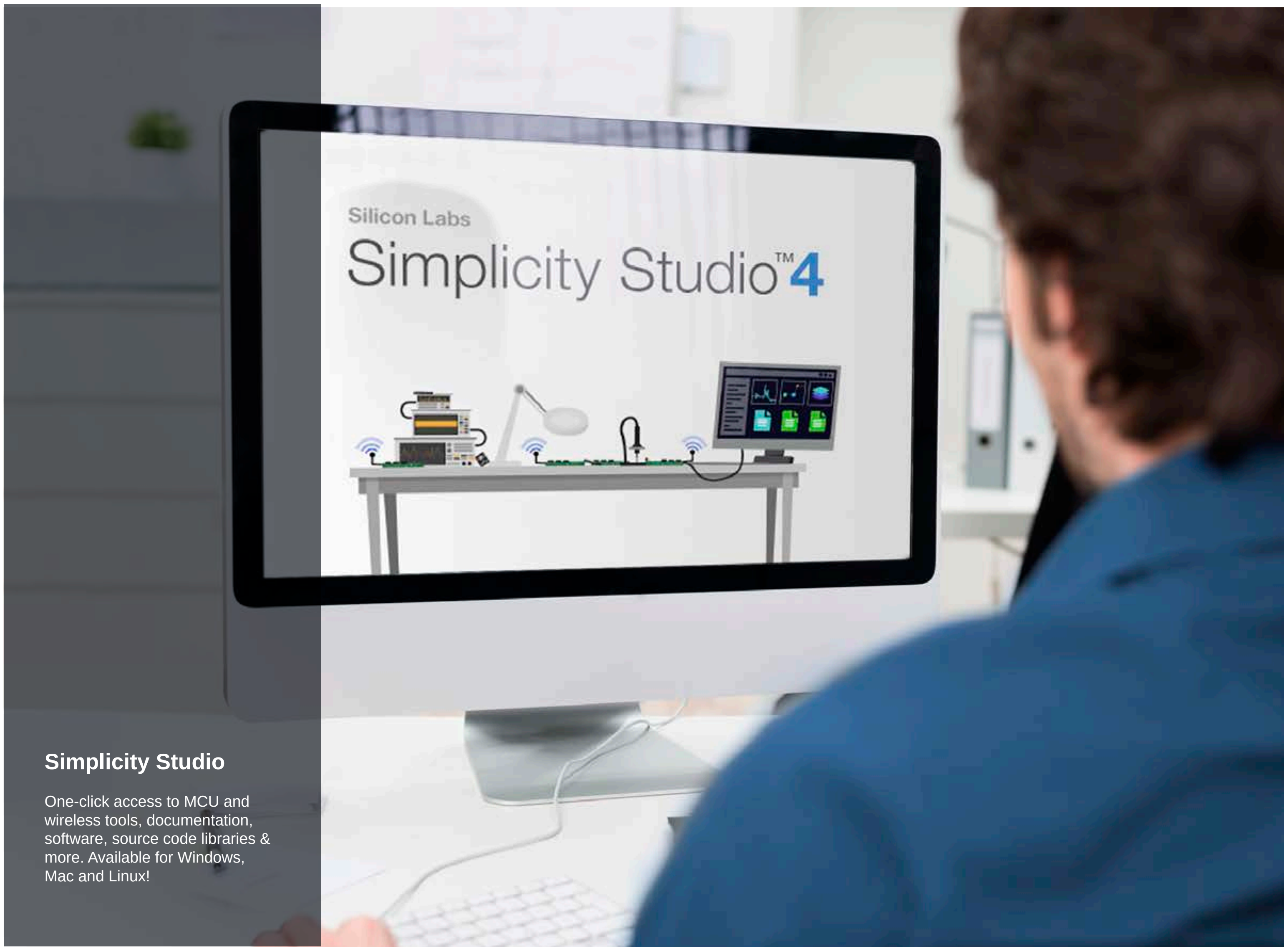
8. Example Code Algorithm

The example program included with this software has an array containing the following delta values:

{24296, 42097, 54367, 64352, 72993, 80729, 87794, 94341, 100470}

The program also has a matching array index that points to any element of this array. The current array element is compared to the value of temp_distance. If temp_distance is larger than the current array element, the program will increment the OSCIFIN register and also increment the array index.

Unless the array index points at the first element of the array, the program will also compare the value of temp_distance to the element that precedes the current array element. If temp_distance + 500 is less than the previous array element, the program will decrement OSCIFIN and decrement the array index. In this comparison, half a degree (500) is added to temp_distance to add a hysteresis effect.



Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com