AN1297: Custom Direction-Finding Solutions using the Silicon Labs Bluetooth Stack

Bluetooth 5.1 introduced support for Direction Finding by adding the option to send and receive Constant Tone Extensions (CTEs) after Bluetooth packets. This makes it possible to measure the phase of the incoming signal on different antennas. However, calculating the direction of the incoming signal from the measured phases is the responsibility of the application, and not of the Bluetooth stack. This document explains the interface between the stack and the application regarding phase measurements to enable implementing custom direction-finding solutions using Silicon Labs' Bluetooth stack. The information in this document provides an alternative to using the Silicon Labs customizable reference implementation, and is intended for developers who have a deep understanding of direction-finding algorithms and prefer to develop their own solution.

Developers who prefer to start with the Silicon Labs customizable direction-finding solution should refer to *AN1296: Application Development with the Silicon Labs RTL Library* instead of this document.

KEY POINTS

- Constant Tone Extension
- Antenna Switching
- IQ samples
- Phase compensation

 Copyright © 2021 by Silicon Laboratories

1 Introduction

This document explains how The Silicon Labs Bluetooth stack supports Direction Finding. Before going forward, the reader must have a basic knowledge of Direction Finding. To learn the basics, refer to *UG103.18: Bluetooth® Direction Finding Fundamentals*. Reading the Technical Overview of Bluetooth direction Finding published by the Bluetooth SIG: https://www.bluetooth.com/wp-content/uploads/Files/developer/1903_RDF_Technical_Overview_FINAL.pdf is also recommended.

Direction Finding, which is estimating the angle of the incoming signal, is based on the concept of antenna arrays, where multiple antennas specially arranged in space sample the same reference signal. The reference signal is a continuous wave where both frequency and phase are maintained over a time interval long enough to be sampled on all antennas. The samples are then turned into phase differences, and phase differences are turned into angle estimation.

Since Bluetooth 5.1 introduced Constant Tone Extensions (CTEs) that can be applied after regular Bluetooth packets, the Silicon Labs Bluetooth stack makes it possible to sample these extensions on different antennas and provide the antenna samples to the application. This document explains the antenna sample format. The samples can be used by anyone who wants to implement their own direction-finding application that turns antenna samples into angle estimation.

If you are unfamiliar with direction finding, or if you are familiar with the concept but you do not have a deep understanding of angle estimator algorithms, Silicon Labs strongly recommends using the RTL library in your application. The RTL library provides a full solution to calculate the angle of the incoming signal from antenna samples in a real world environment. To learn more about the Silicon Labs solution, refer to *AN1296: Application Development with the Silicon Labs RTL Library.* If you decide to continue development using the Silicon Labs RTL library, you can skip this document.

2 Constant Tone Extension

2.1 Concept

To determine the angle of the incoming signal it is essential to transmit a signal with continuous phase, constant amplitude, and constant frequency over a time period long enough to be sampled by all receiver antennas. Transmitting a CW (continuous wave) signal for a long time is not recommended outside of test environments, because it has a very sharp spectrum and can cause serious interference with other devices working in the 2.4 GHz frequency range. Therefore, a short CW must be used, and the transmitter and the receiver must be synchronized so that both know when the CW signal is sent.

Many solutions can be found to overcome this simple problem, and indeed there are many indoor positioning solutions on the market already using direction-finding algorithms. None of them is based on a well-known standard, however. The Bluetooth standard, in contrast, is widespread, and it solves the synchronization problem by its nature: Bluetooth packets are sent with very strict timing, and the peer devices resynchronize their clocks on each reception.

Bluetooth 5.1 introduced a new method to request and send short CW signals as an extension of a normal package. This extension is called Constant Tone Extension (CTE), and it is sent after the CRC of the package when requested.

CTEs can be sent both through a connection (in LL_CTE_RSP packets after an LL_CTE_REQ packet was received), and in periodic advertisements (in AUX_SYNC_IND packets). Additionally, the Silicon Labs Bluetooth stack provides a non-standard solution where CTEs can be sent in extended advertisements (AUX_ADV_IND packets), which makes Direction Finding much more scalable regarding the number of assets to be located. For more info on CTEs see the *Bluetooth Core Specification*, v5.1 or later.

2.2 Sending and Receiving CTEs

To enable CTE responses on a connection use the API `sl_bt_cte_transmitter_enable_connection_cte()` on the CTE transmitter side. To send a CTE request from the receiver to the transmitter, use the API `sl_bt_cte_receiver_enable_connection_cte()` on the receiver side. If CTE responses are enabled on the transmitter side, the Bluetooth stack automatically responds to each CTE request with a CTE response.

To transmit CTEs in periodic advertisements, first start a periodic advertiser on the transmitter side and then use the API `sl_bt_cte_transmitter_enable_connectionless_cte()`. To start listening to CTEs attached to periodic advertisements, first establish a periodic synchronization on the receiver side and then use the API `sl_bt_cte_receiver_enable_connectionless_cte()`.

To transmit CTEs in Silicon Labs proprietary extended advertisements, first start a (regular) extended advertisement on the transmitter side, and then use the API `sl_bt_cte_transmitter_enable_silabs_cte()`. To start listening to CTEs attached to Silicon Labs proprietary extended advertisements, first start scanning on the receiver side and then use the API sl_bt_cte_receiver_enable_silabs_cte().

Any time a CTE is received by the Bluetooth stack (either via connection or in advertisements), the Bluetooth stack raises an iq_report event. Depending on the mode of transmission, this means either an sl_bt_evt_cte_receiver_connection_iq_report event, an sl_bt_evt_cte_receiver_connectionless_iq_report event, or an sl_bt_evt_cte_receiver_silabs_iq_report event. Each event includes an array of IQ samples. The following sections detail what these samples mean and how to interpret them.

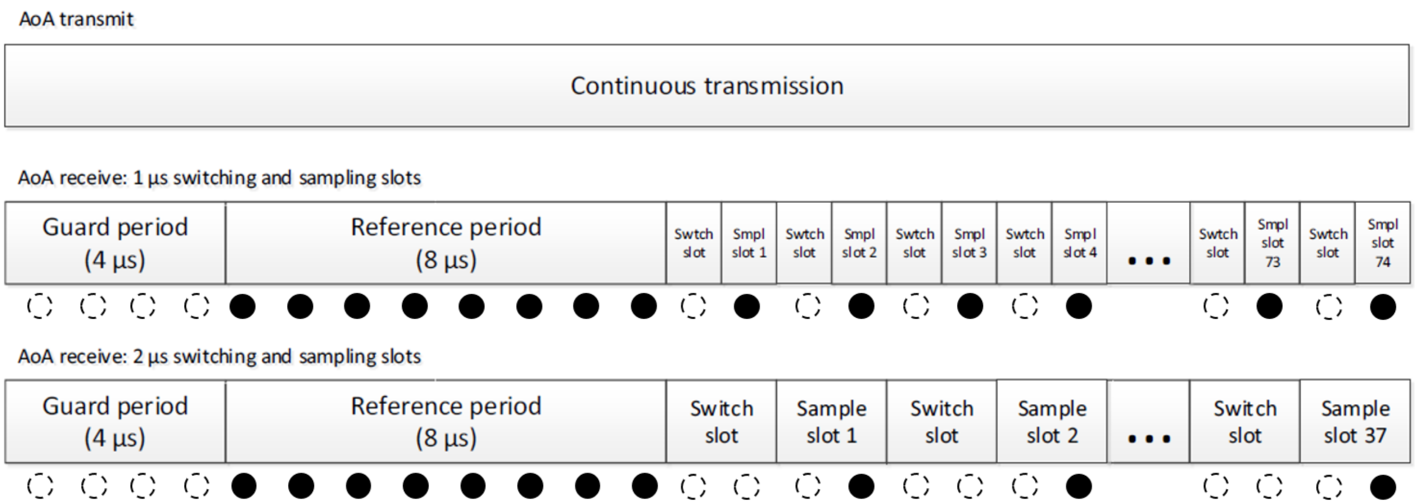
Note: To get the APIs listed above working, CTE Transmitter and CTE Receiver Software components need to be installed in your project. For more information on the API see the Bluetooth API Reference on http://docs.silabs.com.

3 Interpreting IQ Samples

3.1 Sampling

Although the Constant Tone Extension is a simple continuous wave – with a variable length of 16 µs to 160 µs – it is divided into periods as defined in the Bluetooth Core Specification (Vol. 6, Part B, Section 2.5).

The first 4 µs of the Constant Tone Extension is called the guard period and the next 8 µs is called the reference period. After the reference period, the Constant Tone Extension consists of a sequence of alternating switch slots and sample slots, each either 1 µs or 2 µs long, as specified by the application using the CTE Transmitter and CTE Receiver APIs. 2-µs slots make it possible to use a cheaper RF switch between the antennas that has a longer transition time. 1-µs slots make it possible to sample each antenna multiple times, which can help reduce the effect of noise and result in better accuracy.



Once a CTE has started, the radio starts sampling the In-phase (I) and Quadrature (Q) components of the baseband signal with its native sample rate. The samples are then downsampled to 1 sample/µs sample rate. The first 4 samples (taken in the guard period) are discarded, then 8 samples (taken in the reference period) are stored in the sample buffer. Finally, every sample taken in switching slots is discarded and every sample taken in sample slots is stored in the sample buffer. In case of 2-µs slots, only one sample is kept for each sample slot. Stored samples are marked with filled circles on the previous figure.

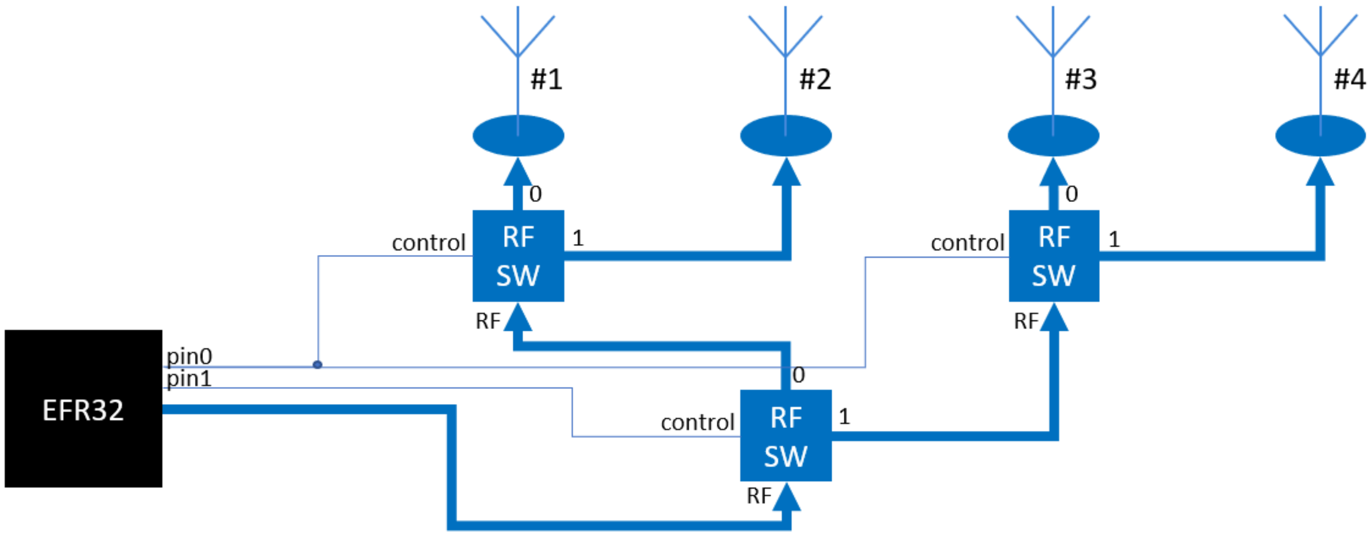
At this point, $N = 8 + (L - 12) / 2 / S$ number of samples are in both the In-phase (I) and in the Quadrature (Q) sample buffer, where L is the length of the CTE in microseconds, and S is the length of the slots in microseconds. The I and Q samples are merged into a common IQ sample buffer with 2N elements so that I and Q samples follow each other alternating:

I(0), Q(0), I(1), Q(1), I(2), Q(2), I(3), Q(3), … I(N), Q(N).

This IQ sample buffer is passed to the application for further processing.

3.2 Antenna switching

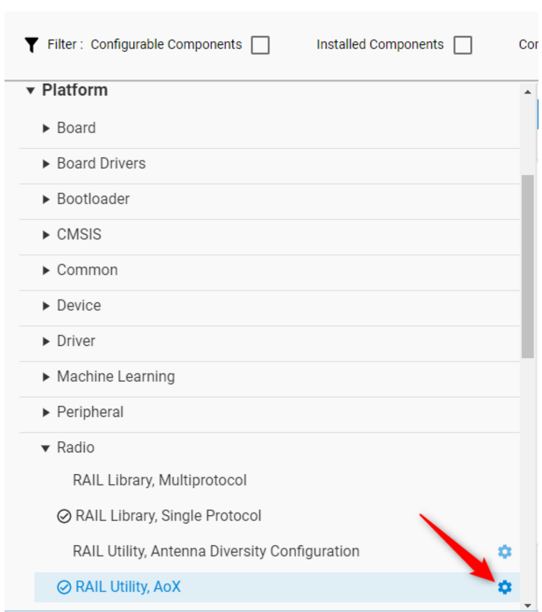
The goal of CTE is, to be able to sample the same continuous wave on different antennas. Since the radio can only sample on one antenna at a time, an RF path switching network must be created among the antennas, as shown in the following figure.



This way, any antenna can be connected with the EFR32 through control pins. The Bluetooth stack is designed so that it can drive up to six output pins to control the RF switches during CTE reception. With this solution up to 64 antennas can be addressed. As an example, on the previous image the four antennas can be addressed with the following addresses:

address	pin1	pin0	antenna
0x00	0	0	#1
0x01	0	1	#2
0x02	1	0	#3
0x03	1	1	#4

The six control pins can be selected in the configuration file of the **Platform > Radio > RAIL Utility, AoX** software component.



The antenna switching pattern, which is the order of addressing the different antennas, can be set with the CTE Receiver and CTE Transmitter Bluetooth APIs as described in the Bluetooth API Reference. For example, to set the antenna switching pattern for connectionless AoA mode, use the following code:

```
// Antenna switching pattern
static const uint8_t antenna_array[16] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };

sl_bt_cte_receiver_enable_connectionless_cte(sync_handle,
                                             CTE_SLOT_DURATION,
                                             CTE_COUNT,
                                             sizeof(antenna_array),
                                             antenna_array);
```

The Bluetooth stack automatically controls the antenna switch pins in the switch slots of the CTE according to the defined switching pattern. The switching pattern can have any length between 1 and 35. It is repeated over the CTE time period, in other words once the end of the pattern is reached it starts over, and this is repeated until the end of the CTE signal. Note that the first switch happens after the reference period, which means that the first antenna in the pattern will be sampled 8 times.

Considering all of this, a switching pattern of [4 3 2 1] results in the following IQ samples:

I4	Q4	I4	Q4	I4	Q4	I4	Q4	I4	Q4	I4	Q4	I4	Q4	I4	Q4	I3	Q3	I2	Q2	I1	Q1	I4	Q4	I3	Q3	…

where the indices mean the address of the antenna. Note, that the addresses do not necessarily have to be consecutive values starting from 0. Addresses can be any value between 0 and 63 and they should always be derived from the RF switching network.

3.3 Phase Retrieval

IQ samples are passed to the application in a form of a uint8 array, with each I and Q sample taking 1-1 byte. Note, however, that I and Q samples are signed integers, which means that the uint8 values must be casted to int8 values immediately after the application receives them from the Bluetooth stack. Using uint8 values is simply a limitation of the BGAPI.

I and Q samples represent the in-phase and quadrature components of the signal, that is

$s(t) = I(t) \cos(2 \pi f t) + Q(t) \sin(2 \pi f t)$,

where $s(t)$ is the incoming signal, f is the carrier frequency and t is time. If the signal is phase-modulated (or frequency-modulated, which can be derived from phase modulation), it can be written in the following form:

$s(t) = A \cos(2 \pi f t + \varphi(t)) = A \cos(2 \pi f t) \cos(\varphi(t)) – A \sin(2 \pi f t) \sin(\varphi(t))$,

where A is the amplitude of the signal and $\varphi(t)$ is the instantaneous phase. From these equations the I and Q components of a phase modulated signal correspond to

$I(t) = A \cos(\varphi(t))$ and $Q(t) = - A \sin(\varphi(t))$

from which it is easy to retrieve the amplitude and the instantaneous phase of the signal:

$$A = \sqrt{I^2 + Q^2} \qquad \phi = \operatorname{atan2}(-Q, I)$$

Note that the amplitude does not represent the absolute amplitude of the signal, since it is compensated with AGC (automatic gain control). To learn the absolute amplitude of the signal, RSSI must be accounted for as well. Direction-finding algorithms do not usually need to do this, since all that matters is the phase difference and amplitude difference between antennas.

3.4 Phase Compensation

Ideally, all antennas should be sampled at the same time to easily calculate the phase difference between them. However, this is not possible with a single radio. The only way to sample more than one antenna is time division, which also means that the sampling is shifted in time.

Note that, if antennas are not sampled at the same time, then even if $\varphi(t)$ is constant and same for all antennas, there will be a phase increase of $\phi_d = 2\pi f \Delta t$ between two samples, where f is the carrier frequency and Δt is the time difference between two samples.

Although the incoming signal is mixed and downsampled by the radio such that the central frequency of the channel corresponds to a DC signal in the baseband, the CTE will not have zero frequency in the baseband for two reasons:

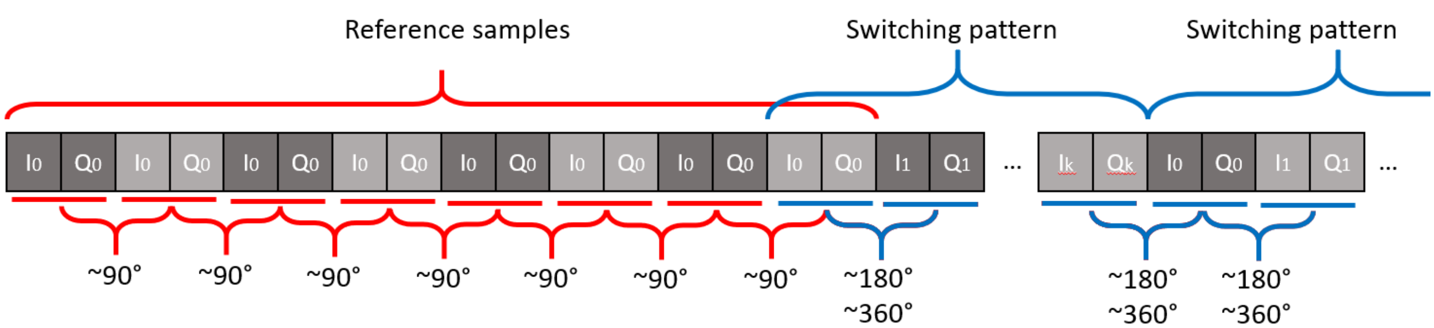
- CTE is not using the central frequency. Instead it is generated as a sequence of continuous non-whitened 1's, which corresponds to a continuous wave with the frequency of $f_c + f_\Delta$, where f_c is the central frequency of the channel and f_Δ is the frequency deviation. This corresponds to a frequency of f_Δ in the baseband.
- Oscillators are not always perfectly tuned, which results in an offset between the ideal carrier frequency and the real carrier frequency. This again introduces a frequency shift f_{offset} in the CTE frequency.

Considering this, the frequency of the CTE signal in the baseband is $f_\Delta + f_{offset}$, where f_Δ = 250kHz according to the Bluetooth specification, and f_{offset} is in the order of 10kHz. Consequently, the phase shift between two consecutive samples with a 1-µs difference is around $\phi = 2\pi(f_\Delta + f_{\text{offset}})t \cong 2\pi \cdot 0.25 \cdot 1 = 0.5\pi = 90°$. Taking the offset frequency into account as well, it might vary between 80°-100°.

Since the reference samples are taken on the same antenna, they can be used to measure the actual phase shift over a 1-µs time period. It is strongly recommended to calculate the phase shift for each packet, as it may vary between packets. That is, every time an iq_report event is received, the first 8 IQ samples should be used to determine the phase shift for the actual packet. This can be done either by averaging the phase differences between the first 8 samples or by applying a median filter on those phase differences.

Note, however, that once a phase shift for 1 µs is calculated, this value should be multiplied and used to compensate for the phase shift between samples after the reference period due to the non-sampled switching slots, as illustrated in section 3.1 Sampling. For 1- µs slots, 2 times the value should be used, and for 2- µs slots, 4 times the value should be used.

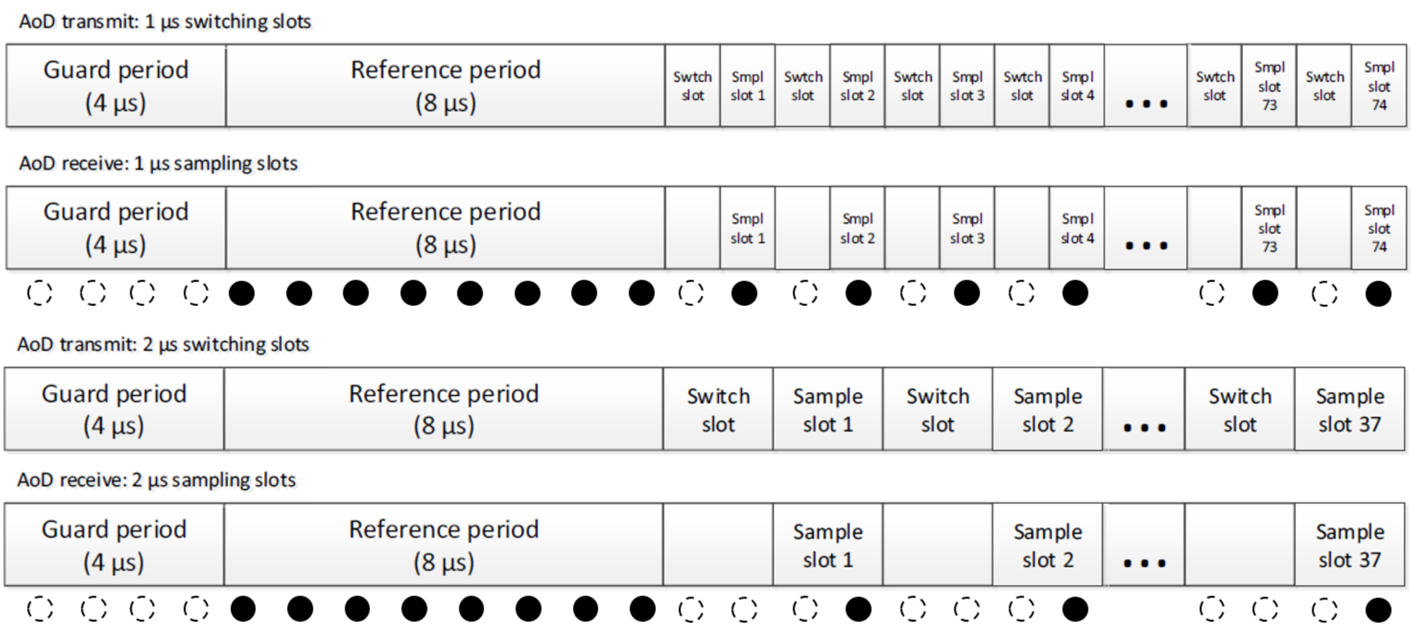
The content of the IQ sample buffer for a *k*-long antenna switching pattern is illustrated in the following figure along with the expected phase differences between the consecutive IQ samples, assuming that there is no phase shift between the antennas due to oblique angle of arrival.



4 Angle of Arrival / Angle of Departure

The previous sections primarily discuss the case of Angle of Arrival (AoA) method, where the transmitter sends a continuous wave on a single antenna, and the receiver samples the incoming signal on multiple antennas.

In the Angle of Departure (AoD) method, the antenna switching happens on the transmitter side, and the receiver samples the incoming signal on a single antenna. This, however, does not change anything on the structure and interpretation of the IQ samples. The fact that antenna switching happens on the transmitter side does not affect the result at all. The only difference is in the configuration. In the case of AoD the switching pins must be set up on the transmitter, and both the transmitter and the receiver must know the switching pattern. This contrasts with AoA, where the switching pattern need only be configured on the receiver side.



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 & Community
www.silabs.com/community

Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications. **Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project**

Trademark Information
Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals®, WiSeConnect , n-Link, ThreadArch®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress® , Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

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

www.silabs.com