Zephyr Bluetooth Low Energy Controller

nWP-029

White Paper

v1.1

NORDIC® SEMICONDUCTOR

Contents

Revision history

Date	Version	Description
December 2019	1.1	Fixed broken links
November 2017	1.0	First release

1 Introduction

The Zephyr™ open source project provides a *Bluetooth*® Low Energy (LE) Controller that runs on all nRF5 ICs. The Controller is compatible with any Bluetooth-compliant Host.

This white paper gives a short overview of the Zephyr project, describes in what cases you should consider using the Zephyr Bluetooth LE Controller instead of one of Nordic Semiconductor's SoftDevices, and explains how to build and program this Controller on nRF5 ICs.

2 The Zephyr Project

The Zephyr Project is a Linux Foundation open source project that focuses on the development of a Real Time Operating System (RTOS) for small embedded devices.

As of version 1.9, the project includes the following features:

- Support for the ARC, Arm® (Cortex®-M), NIOS2, RISCV-32, x86, and Xtensa architectures
- Tickless kernel with support for cooperative and preemptible threads
- Driver and sensor framework with power management
- Native TCP/IP protocol stack over Bluetooth Low Energy, 802.15.4, and Ethernet
- HTTP, CoAP, MQTT, LWM2M, and DNS
- Native Bluetooth Low Energy 5.0 Host and Controller protocol stack
- Cryptographic framework with underlying mbed TLS and TinyCrypt

See What is the Zephyr Project? on the Zephyr Project website for additional information about the Zephyr project's statement and goals.

Zephyr follows a continuous development process with the latest changes available in the project's master branch. All changes come in the form of GitHub pull requests, which are reviewed by the relevant subsystem code owners, and anyone can submit contributions provided they do so in accordance with the Zephyr contribution guidelines. About every three months, a major release is tagged after being tested to ensure that the release is stable, reliable, and ready for deployment. Minor (point) releases are prepared and tagged ad-hoc if needed.

Nordic Semiconductor actively contributes to the Zephyr open source project, both to improve and extend it, and to ensure that the RTOS runs as reliably and efficiently as possible on nRF5 ICs. Additionally, Nordic Semiconductor ran the v1.9 release of the Zephyr RTOS through the Bluetooth qualification process (Bluetooth LE Controller only) to enable product qualification based on Zephyr.

Note: The core of the codebase in the Zephyr Project is licensed under the Apache License, Version 2.0, with the exception of third-party code that is included in a special folder named "`ext/`".

2.1 Maintenance and support

The Zephyr Project's development model is different from the one typically used in proprietary software, because the Zephyr RTOS is an open source project under the umbrella of the Linux Foundation. As such, it is developed independently of Nordic Semiconductor by multiple contributors, and it is governed by the Technical Steering Committee and the Zephyr Board.

Zephyr development happens publicly in the main Zephyr GitHub repository. Continuous integration is used to ensure that the daily contributions do not break the build for the set of architectures and boards that are already supported. This means that the "latest and greatest" is always available to everyone in the project's master branch, and this codebase can be used to preview and test new features that will later be included in a release.

Continuous integration means that the latest state of development might not pass Bluetooth qualification. However, Nordic Semiconductor intends to re-test the Bluetooth LE Controller builds on every major Zephyr release. QDIDs of the latest Zephyr qualification are available in the Listings search (search for "Nordic Semiconductor" and check the latest Controller Subsystem with Zephyr).

Zephyr has a set of community tools that are available to all users. They can be used to report issues with the Bluetooth LE Controller, to request additional features, or in general to give feedback or ask questions

about the RTOS. Use these communication channels to contact the developers if you have questions about building or running the Bluetooth LE Controller. The most useful tools are the following:

- For reporting issues, see Zephyr GitHub repository issues.
- For asking questions or providing feedback, see Zephyr mailing lists.
- For interacting with users and developers directly, use the #zephyrproject channel on freenode.net.

3 Bluetooth LE Controller

A Bluetooth Low Energy stack is split into two different layers: the Bluetooth Low Energy Host and the Bluetooth Low Energy Controller. These software subsystems can be run either on the same IC or on two separate ICs that are connected through a physical serial interface and the standard Host Controller Interface (HCI) protocol.

For details about the configurations and layers of a Bluetooth Low Energy stack, see the blog post Building a qualified BLE Controller with Zephyr OS 1.9++.

Nordic Semiconductor SoftDevice

Nordic Semiconductor's SoftDevices include a full Bluetooth Low Energy stack including both Host and Controller. This solution is ideal for the following use cases:

- A single-chip configuration where the application, the Bluetooth LE Host, and the Bluetooth LE Controller run on the same IC
- A dual-chip configuration where the application runs on one IC and the SoftDevice including Host and Controller on another (with the serialized GAP/GATT API of the SoftDevice over a physical serial interface linking the two together)

Nordic Semiconductor does not currently offer a stand-alone Bluetooth LE Controller variant of its SoftDevice protocol stack.

Zephyr Bluetooth LE Controller

If you require a stand-alone Bluetooth LE Controller, the Zephyr Project Bluetooth LE Controller provides a solution on Nordic Semiconductor devices. Such a stand-alone Controller is required, for example, for the following use cases:

- Designs that rely on an existing (home-grown, commercial, or open source) Bluetooth LE Host implementation
- Designs that benefit from the functionality being split among two different ICs with one (MCU or CPU) running the application and the Bluetooth LE Host, and the second (MCU) running the Bluetooth LE Controller
- Designs that require a Nordic Semiconductor IC to expose the standard Host Controller Interface (HCI) protocol that can communicate with any existing Bluetooth LE Host

Since the Zephyr Bluetooth LE Controller is distributed in source code form, you can easily add any required functionality in the form of Vendor Specific (VS) HCI commands and events. This might be useful if you need to make additional use of the hardware and peripherals included in the Nordic Semiconductor IC, or even to add support in the Controller for proprietary protocols outside the scope of the Bluetooth specification.

Note: Changing the design of the standard controller implementation may invalidate the Bluetooth qualification listing obtained by Nordic Semiconductor. Configuration and additions, on the other hand, should not impact the qualification listing.

The responsibility of evaluating the extent of design changes when listing a product lies with you as the end product maker.

3.1 Compatibility

The Zephyr Bluetooth LE Controller runs on all nRF5 ICs, and it is compatible with any Bluetooth compliant Host.

LE Controller support on Nordic Semiconductor devices

The Zephyr Bluetooth LE Controller has been verified to run on all of Nordic Semiconductor's nRF5 ICs with Bluetooth Low Energy functionality. The following ICs have been tested:

- nRF51822 and nRF51422 (note that some Bluetooth features are not available on the nRF51 Series)
- nRF52810
- nRF52832
- nRF52840

Zephyr currently supports the following HCI physical transports for the Bluetooth LE Controller:

UART
: This transport complies with the UART transport layer as defined in the Bluetooth Core Specification, Vol. 4. The Three-Wire UART transport layer is not yet supported.

  UART speed is configurable. If not specified, it defaults to 1 Mbit/s.

SPI
: This transport is not a standard HCI transport in the Bluetooth specification, but it has been implemented in Zephyr. If this transport is used, the Bluetooth LE Host must be the SPI master and the Bluetooth LE Controller the SPI slave.

The Controller has Bluetooth Low Energy Direct Test Mode (DTM) built in, which makes production testing and RF PHY re-evaluation easy. It supports all mandatory and optional DTM features over an HCI transport.

Nordic Semiconductor provides the required QDIDs (Qualification Design IDs) for the Controller subsystem. This simplifies the Bluetooth qualification process for designs based on nRF5 ICs running the Zephyr Bluetooth LE Controller. You can combine the Controller QDIDs with the relevant Host QDIDs for the Bluetooth LE Host that you are using. The following functionality has been qualified for the nRF5 Series:

- nRF51: All Bluetooth 5 features except for Advertising Extensions, Data Length Extension, 2 Mbps, coded PHYs, and Controller-based privacy
- nRF52: All Bluetooth 5 features except for Advertising Extensions

The following link contains the QDID for Zephyr v1.9.x, which is currently the latest qualified version:

- Zephyr Bluetooth LE Controller subsystem QDID

LE Host compatibility

The Zephyr Project's Bluetooth LE Controller is compatible with any Bluetooth compliant Host with version support 4.0 and above.

The Controller has been tested against the following Hosts:

- The Zephyr Bluetooth LE Host, both in single-chip and dual-chip configurations (Zephyr v1.9.0 or later)
- The Linux Bluetooth stack BlueZ (starting with kernel 4.10 and BlueZ 4.45)
- BlueKitchen's BTstack (through the posix-h4-zephyr port)

The following Zephyr supported boards use the Zephyr Host and Controller dual-chip configuration:

- HCI over SPI: The 96Boards Carbon combines an STM32 MCU running both the application and the Zephyr Bluetooth LE Host with an nRF51 IC acting as a LE Controller.

- HCI over UART: The Arduino/Genuino 101 contains a Curie module, which combines an Intel Quark x86 MCU with an nRF51 IC.

The Zephyr Bluetooth LE Controller implements the standard HCI Bluetooth Low Energy commands and events to ensure interoperability with a wide range of Bluetooth LE Hosts. In addition, it includes support for a series of Zephyr HCI extensions. These HCI extensions are defined by the Zephyr Project and add functionality that is not defined by the Bluetooth specification, for example:

- Reading the factory-generated Random Static Address over HCI
- Reading the factory-generated security key roots (ER, IR) over HCI

4 Using the Zephyr Bluetooth LE Controller

Nordic Semiconductor does not provide precompiled images of the Zephyr Bluetooth LE Controller because of the large amount of build-time configuration options that Zephyr supports. Therefore, the Controller must be built from the source and programmed onto an nRF5 IC before it can be used.

The following instructions are based on the Zephyr Getting Started Guide.

1. Set up your development environment using the instructions from the official Zephyr documentation.

   The following operating systems are supported:

   - GNU/Linux - The required Arm toolchain is part of the Zephyr SDK.
   - macOS - The GNU ARM Embedded toolchain must be installed separately.
   - Microsoft Windows - The GNU ARM Embedded toolchain must be installed separately.
2. Get a copy of the Zephyr GitHub repository, if you do not have one already.

   There are different ways of obtaining the repository:

   - Go to the Zephyr Downloads, select and download the version that you want to use, and extract it.
   - Clone the repository. To clone the repository anonymously, enter the following command in a command prompt:

     ```
     git clone https://github.com/zephyrproject-rtos/zephyr.git
     ```

     Then enter the repository and switch to the branch of the version that you want to use. For example, for version 1.9.2, enter the following command:

     ```
     cd zephyr
     git checkout v1.9.2
     ```
3. In the command prompt, navigate to the folder that contains the application for the HCI physical transport that you want to use:

   - UART transport layer: samples/bluetooth/hci_uart
   - SPI transport layer (non-standard): samples/bluetooth/hci_spi
4. Build the application, specifying the board that the build will target and the configuration of the application.

   The command for building depends on the Zephyr version that you use. For Zephyr 1.9, enter the following command:

   ```
   make BOARD=board CONF_FILE=config_file
   ```

   For Zephyr 1.10 or later, enter the following commands:

   ```
   mkdir build && cd build
   cmake -DBOARD=board -DCONF_FILE=config_file ..
   make
   ```

   Use the following parameters:

board

Selects the board that the build will target. Boards are defined in the boards folder, and they typically describe a single PCB. If you are using a custom board, you must add all necessary files in a folder inside `boards`. See the Zephyr Board Porting Guide for more information.

Specify the identifier of the board that you plan to use, for example, nrf52840_pca10056. If the BOARD parameter is omitted, `BOARD=nrf52_pca10040` is used for the UART sample and `BOARD=96b_carbon_nrf51` is used for the SPI sample.

config_file

Specifies the default configuration file for the build that is used to set or override any configuration options of the Zephyr kernel or the application itself. See the Zephyr application configuration documentation for more information.

If the CONF_FILE parameter is omitted, `CONF_FILE=nrf5.conf` is used for the UART sample and `CONF_FILE=prj.conf` is used for the SPI sample. The `nrf5.conf` file that is used for the UART sample configures the RS232 settings to the settings from the Bluetooth Core Specification:

Setting	Value
Baud rate	Extracted from Device Tree (DT) `.dts` file for the board in `dts/arm/` or from the `.overlay` file if present for the board in `samples/bluetooth/hci_uart`.
Number of data bits	8
Parity bit	No parity
Stop bit	1
Flow control	RTS/CTS

5. Program the nRF5 IC with the resulting HEX file to make it a Bluetooth LE Controller.

   See the Zephyr Nordic nRF5x Segger J-Link documentation for instructions on how to program the file. If enabled, you can capture debug output from the Bluetooth LE Controller build via Segger RTT if the target board has a Segger-based debugging chip.

Legal notices

By using this documentation you agree to our terms and conditions of use. Nordic Semiconductor may change these terms and conditions at any time without notice.

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function, or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Nordic Semiconductor ASA does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. If there are any discrepancies, ambiguities or conflicts in Nordic Semiconductor's documentation, the Product Specification prevails.

Nordic Semiconductor ASA reserves the right to make corrections, enhancements, and other changes to this document without notice.

Life support applications

Nordic Semiconductor products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury.

Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

RoHS and REACH statement

Complete hazardous substance reports, material composition reports and latest version of Nordic's REACH statement can be found on our website www.nordicsemi.com.

Trademarks

All trademarks, service marks, trade names, product names, and logos appearing in this documentation are the property of their respective owners.

Copyright notice

COMPANY WITH
QUALITY SYSTEM
CERTIFIED BY DNV GL
= ISO 9001 =