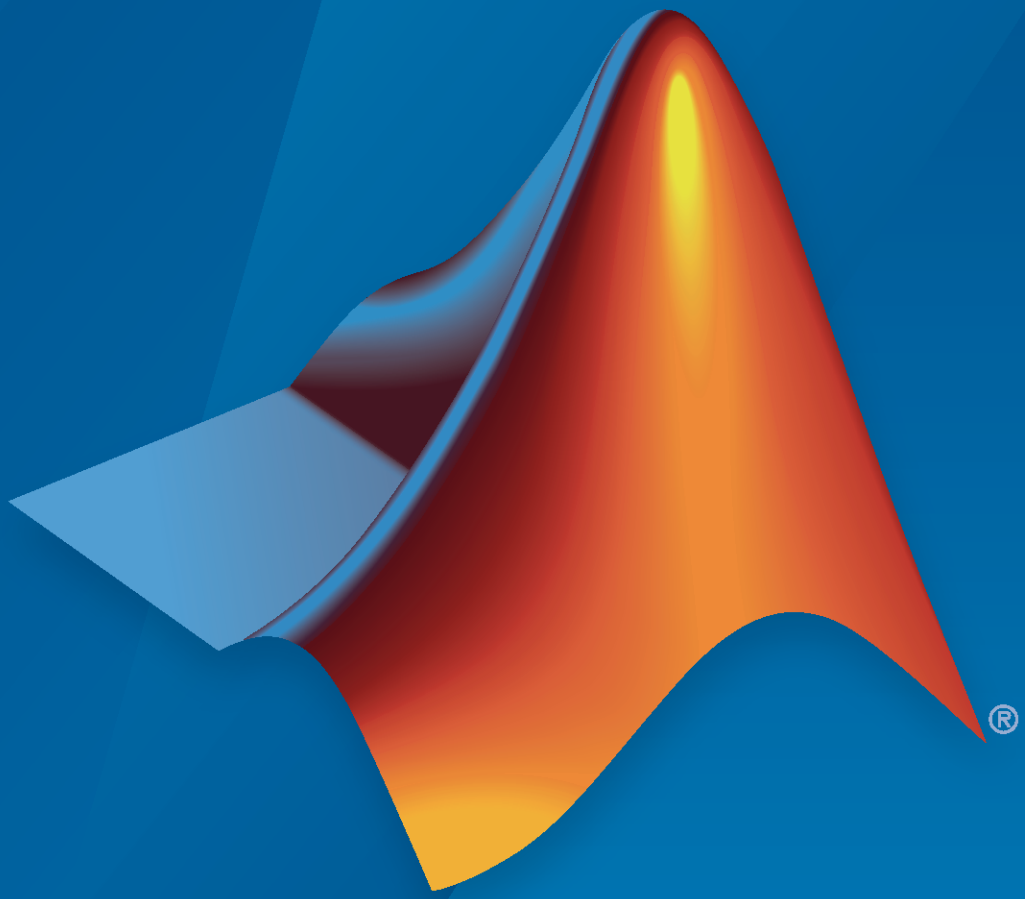
Bluetooth® Toolbox

Getting Started Guide



MATLAB®

R2023a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Bluetooth® Toolbox Getting Started Guide

Revision History

March 2022	Online only	New for Version 1.0 (Release 2022a)
September 2022	Online only	Revised for Version 1.1 (Release 2022b)
March 2023	Online only	Revised for Version 1.2 (Release 2023a)

Contents

1 Introduction to Bluetooth Toolbox

2 About Bluetooth

3 Tutorials

Introduction to Bluetooth Toolbox

Bluetooth Toolbox Product Description

Simulate, analyze, and test Bluetooth communications systems

MathWorks Bluetooth Toolbox provides standard-based tools to design, simulate, and verify Bluetooth communications systems. It supports test waveform generation, golden reference verification, and Bluetooth network modeling.

With the toolbox, you can configure, simulate, and analyze end-to-end Bluetooth communication links. You can create and reuse test benches to verify that your designs, prototypes, and implementations comply with the Bluetooth standard, including Bluetooth Low Energy (LE) and Bluetooth Classic. You can also assess coexistence, interference, localization, and LE Audio scenarios by modeling multiple layers of the Bluetooth protocol stack.

About Bluetooth

Bluetooth Technology Overview

Bluetooth [1] wireless technology is the air interface intended to replace the cables connecting portable and fixed electronic equipment. Bluetooth device manufacturers have the flexibility to include optional core specification features to optimize and differentiate product offers.

Bluetooth is equated with the implementation specified by the Bluetooth Core Specification [3] group of standards maintained by the Bluetooth Special Interest Group (SIG) industry consortium. The Bluetooth Toolbox functionalities enables you to model Bluetooth low energy (LE) and Bluetooth basic rate/enhanced data rate (BR/EDR) communications system links, as specified in the Core System Package [Low Energy Controller volume], Specification Volume 6. It also enables you to explore variations on implementations for future evolution of the standard. Bluetooth BR/EDR and Bluetooth LE devices operate in the same unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) frequency band as Wi-Fi®.

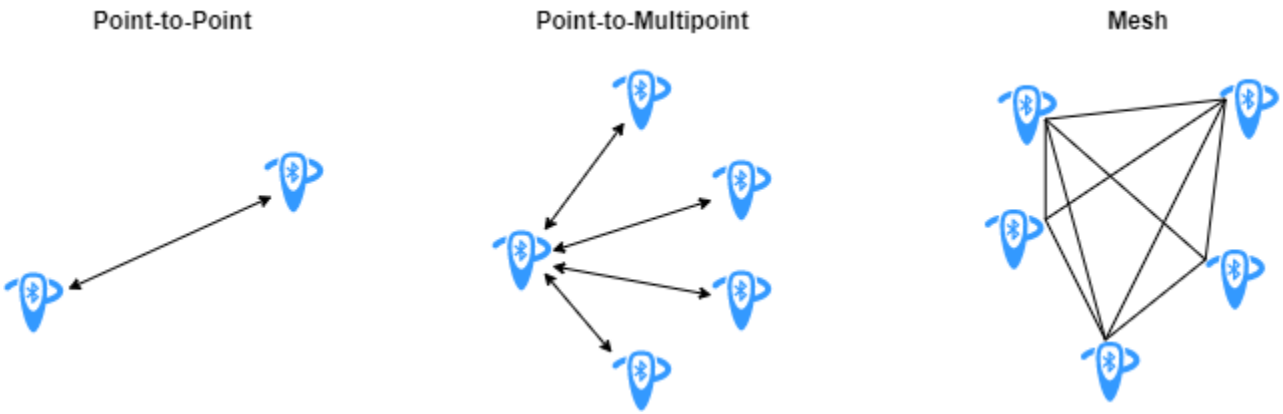
In Bluetooth BR/EDR, the radio hops in a pseudo-random way on 79 designated Bluetooth channels. Each Bluetooth BR/EDR channel has a bandwidth of 1 MHz. Each frequency is located at (2402 + k) MHz, where k = 0,1,...78.

In Bluetooth LE, the operating radio frequency is in the range 2.4000 GHz to 2.4835 GHz, inclusive. The channel bandwidth is 2 MHz and the operating band is divided into 40 channels, k = 0, ..., 39. The center frequency of the k^{th} channel is located at 2402 + k × 2 MHz.

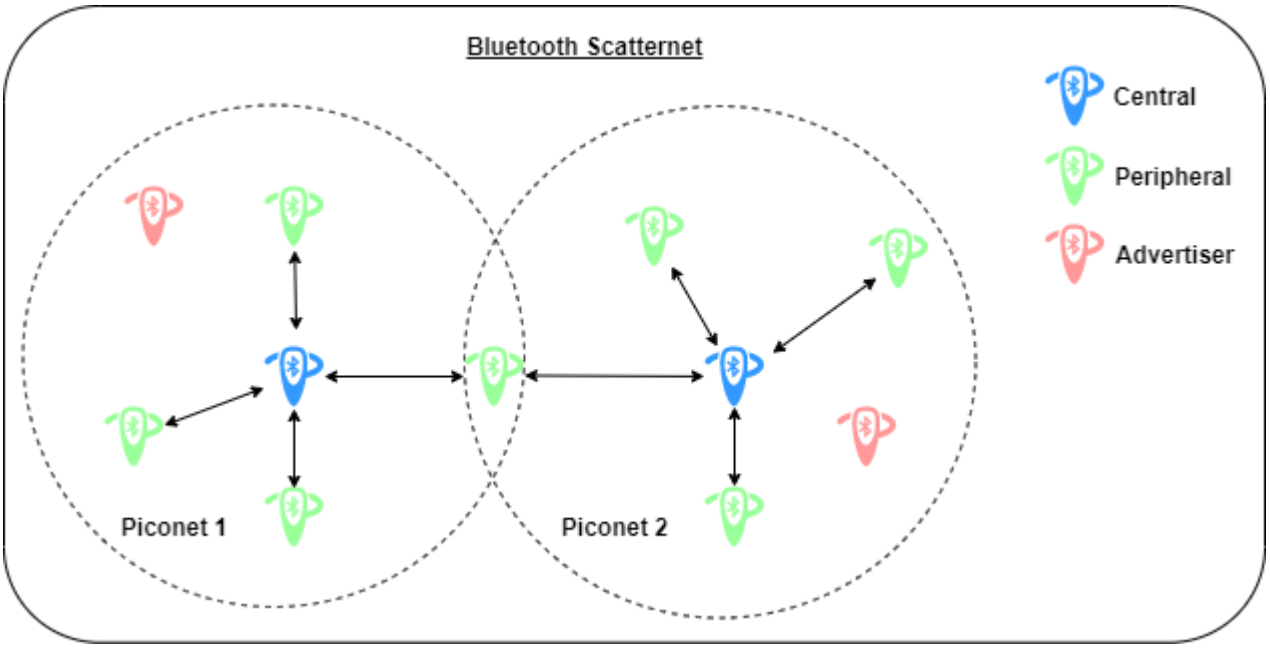
For more information about the specifications of Bluetooth BR/EDR and LE, see “Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications”.

Bluetooth Connection Topologies

Most Bluetooth LE devices communicate with each other using a simple point-to-point (one-to-one communication) or point-to-multipoint (one-to-many communication) topology as shown in this figure.



Devices using one-to-one communication operate in a Bluetooth *piconet*. As shown in this figure, each piconet consists of a device in the role of *Central*, with other devices in the *Peripheral* or *Advertiser* roles. Before joining the piconet, each *Peripheral* node is in an advertiser role. Multiple piconets connect to each other, forming a Bluetooth *scatternet*.



For example, a smartphone with an established one-to-one connection to a heart rate monitor over which it can transfer data is an example of point-to-point connection.

On the contrary, the Bluetooth mesh enables you to set up many-to-many communication links between Bluetooth devices. In a Bluetooth mesh, devices can relay data to remote devices that are not in the direct communication range of the source device. This enables a Bluetooth mesh network to extend its radio range and encompass a large geographical area containing a large number of devices. Another advantage of the Bluetooth mesh over point-to-point and point-to-multipoint topologies is the capability of self healing. The self-healing capability of the Bluetooth mesh implies that the network does not have any single point of failure. If a Bluetooth device disconnects from the mesh network, other devices can still send and receive messages from each other, which keeps the network functioning. For more information about Bluetooth mesh networking, see “Bluetooth Mesh Networking”.

Solution Areas

This table summarizes prominent solution areas of Bluetooth BR/EDR and Bluetooth LE.

Application	Bluetooth BR/EDR	Bluetooth LE
Audio streaming applications such as: • Bluetooth headphones or earbuds • Bluetooth speakers • Bluetooth watches	Supported	Supported

Application	Bluetooth BR/EDR	Bluetooth LE
Location and direction finding applications such as: • Asset tracking • Indoor navigation services • Beacon-based services	Not supported	Supported
Data transmission applications such as: • Medical and health equipments • Sports and fitness equipments • Peripherals and accessories	Not supported	Supported
Device network applications such as: • Monitoring systems and services • Automation systems • Control systems	Not supported	Supported

New Use Cases and Enhancements

These are some of the prominent new Bluetooth use cases and capabilities introduced by the SIG.

- COVID-19 pandemic response solutions — To tackle the challenges of COVID-19 pandemic, many private and Government institutions have turned towards the Bluetooth technology for innovative solutions that can realize and accelerate reopening efforts across the world, and enable safer and faster treatment of patients during the COVID-19 pandemic and future disease outbreaks. To achieve these solutions, these three use cases leverage Bluetooth technology.
  - Exposure notification systems (ENS): Public ENS use the Bluetooth technology present in the smartphones to apprise people when they have been in close proximity with a person who was later diagnosed with COVID-19.
  - Safe return solutions: Public venues such as stadiums, offices, and universities etc. are looking to Bluetooth technology to provide safe return solutions to help them take necessary steps to reopen or continue to operate safely during the pandemic times. Some of the prominent solutions in these use case include occupancy management, exposure management, hygiene management, and touchless access and control.
  - Safe treatment solutions: Medical institutions are leveraging Bluetooth technology to improve the quality and efficiency of diagnosis and treatment. Some of these solutions include safe facility management, safe patient diagnosis and monitoring, remote patient care and monitoring.
- Networked lighting control — Networked lighting control systems feature an intelligent network of individually addressable and sensor-rich luminaries and control elements that enable each device of the system to send and receive data. Bluetooth networked lighting control systems are

deployed in offices, retail, healthcare, factories, and other commercial places to provide a combination of significant energy savings, improved occupant well-being and productivity, and efficient building operations and predictive maintenance. The key advantages of shifting from wired to wireless solutions for networked lighting control are reduced operation and maintenance cost, greater design and configuration flexibility, and future extensibility.

- Bluetooth LE audio — The Bluetooth Core Specification 5.2 [2] introduced the next generation of Bluetooth audio called the LE audio. LE audio operates on the Bluetooth LE standard. LE audio is the next generation of Bluetooth audio, which supports development of the same audio products and use cases as the classic audio. It also enables creation of new products and use cases and presents additional features and capabilities to help improve the performance of classic audio products. Some of the key features and use cases of LE audio include enabling audio sharing, providing multistream audio, and supporting hearing aids. For more information about LE audio, see “Bluetooth LE Audio”.

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed September 14, 2020. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.2. https://www.bluetooth.com/.

[3] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

More About

- “Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications”
- “Bluetooth Protocol Stack”
- “Bluetooth Packet Structure”
- “Bluetooth Location and Direction Finding”
- “Bluetooth LE Audio”
- “Bluetooth-WLAN Coexistence”
- “Bluetooth Mesh Networking”

Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications

Bluetooth technology [1], operating on the 2.4 GHz unlicensed industrial, scientific, and medical (ISM) frequency band, uses low-power radio frequency to enable short-range communication at a low cost. The two variants of the Bluetooth technology are –

- Bluetooth basic rate/enhanced data rate (BR/EDR) or classic Bluetooth
- Bluetooth low energy (LE) or Bluetooth Smart

The Bluetooth Core Specification [2], specified by the Special Interest Group (SIG) consortium, defines the technologies required to create interoperable Bluetooth BR/EDR and Bluetooth LE devices.

Bluetooth BR/EDR radio is primarily designed for low power, high data throughput operations. In Bluetooth BR/EDR, the radio hops in a pseudo-random way on 79 designated Bluetooth channels. Each Bluetooth BR/EDR channel has a bandwidth of 1 MHz. Each frequency is located at (2402 + k) MHz, where k = 0,1, ..., 78.

In 2010, the SIG introduced Bluetooth LE with the Bluetooth 4.0 version. The Bluetooth LE radio is designed and optimized to support applications and use cases that have a relatively low duty cycle. For example, suppose a person wears a heart rate monitoring device for several hours. Because this device transmits only a few bytes of data every second, its radio is in the 'on' state for a very short period of time. In Bluetooth LE, the operating radio frequency is in the range from 2.4000 GHz to 2.4835 GHz. The channel bandwidth is 2 MHz, and the operating band is divided into 40 channels, (k = 0, 1, ..., 39). The center frequency of the kth channel is located at (2402 + k × 2) MHz.

This table summarizes and compares different features of Bluetooth BR/EDR and Bluetooth LE.

Feature	Bluetooth BR/EDR	Bluetooth LE
Frequency band	Operates on a 2.4 GHz Industrial, Scientific, and Medical (ISM) band, with the values in the range from 2.4000 GHz to 2.4835 GHz	Operates on 2.4 GHz ISM band, with the values in the range from 2.4000 GHz to 2.4835 GHz
Channels	79 channels	40 channels (37 data channels and 3 advertising channels)
Channel bandwidth	1 MHz	2 MHz
Spread spectrum technique	1600 hops/sec frequency-hopping spread spectrum (FHSS)	FHSS
Modulation scheme	• Gaussian frequency shift keying (GFSK) • π/4 differential quadrature phase shift keying (DQPSK) • 8 differential phase shift keying (DPSK)	GFSK

Feature	Bluetooth BR/EDR	Bluetooth LE
Power usage	1 W (reference value)	~0.01x W to 0.5x W of reference (depending on the use case scenario)
Maximum transmission power	• Class 1: 100 mW (20 dBm) • Class 2: 2.5 mW (4 dBm) • Class 3: 1 mW (0 dBm)	• Class 1: 100 mW (20 dBm) • Class 1.5: 10 mW (10 dBm) • Class 2: 2.5 mW (4 dBm) • Class 3: 1 mW (0 dBm)
Data rate	• BR PHY (GFSK): 1 Mb/s • EDR PHY (π/4 DQPSK): 2 Mb/s • EDR PHY (8 DPSK): 3 Mb/s	• LE Coded PHY (S = 8): 125 Kb/s • LE Coded PHY (S = 2): 500 Kb/s • LE 1M PHY: 1 Mb/s • LE 2M PHY: 2 Mb/s
Device discovery	Inquiry or paging	Advertising
Device address privacy	None	Private device addressing supported
Encryption algorithm	E0/SAFER+	AES-CCM
Audio capable	Yes	Yes (Bluetooth LE audio is introduced in Bluetooth Core Specification 5.2)
Network topology	Point-to-point (including piconet)	• Point-to-point (including piconet) • Broadcast • Mesh

This table summarizes prominent applications of Bluetooth BR/EDR and Bluetooth LE.

Application	Bluetooth BR/EDR	Bluetooth LE
Audio streaming applications such as: • Bluetooth headphones or earbuds • Bluetooth speakers • Bluetooth watches	Supported	Supported
Location and direction finding applications such as: • Asset tracking • Indoor navigation services • Beacon-based services	Not supported	Supported

Application	Bluetooth BR/EDR	Bluetooth LE
Data transmission applications such as: • Medical and health equipments • Sports and fitness equipments • Peripherals and accessories	Not supported	Supported
Device network applications such as: • Monitoring systems and services • Automation systems • Control systems	Not supported	Supported

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed September 14, 2020. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

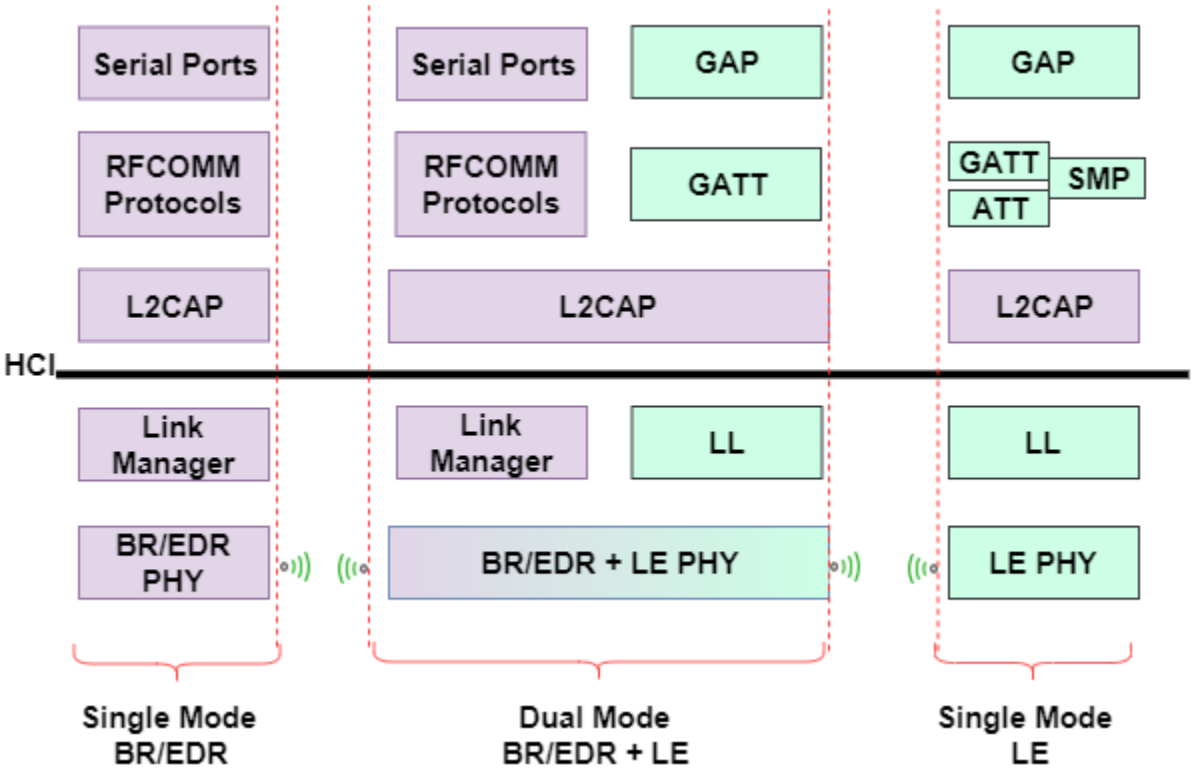
More About

- “Bluetooth Technology Overview”
- “Bluetooth Protocol Stack”
- “Bluetooth Packet Structure”

Bluetooth Protocol Stack

The Bluetooth Special Interest Group (SIG) [1] and [2] defines the protocol stack for Bluetooth low energy (LE) and Bluetooth basic rate/enhanced data rate (BR/EDR) technology. The fundamental objectives of these specifications are to develop interactive services and applications over interoperable radio components and data communication protocols.

This figure shows the architecture of the Bluetooth stack.



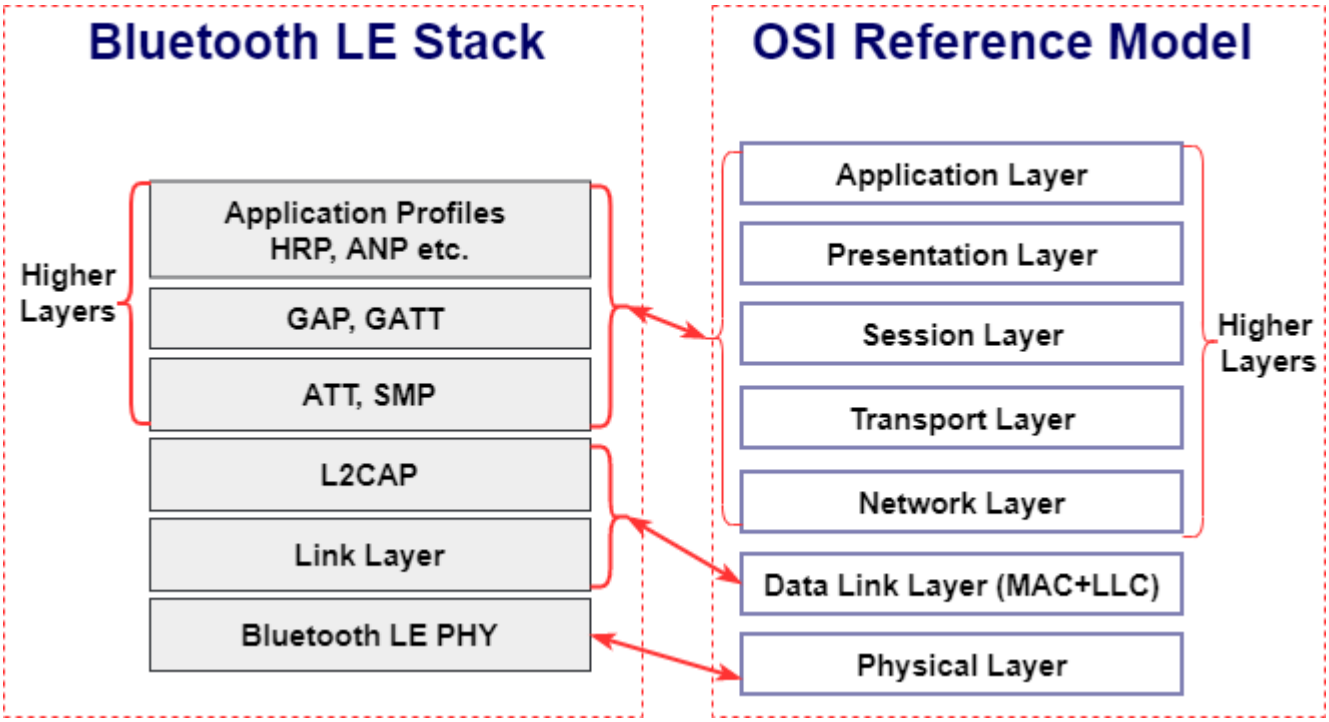
Bluetooth devices can be one of these two types:

- Single mode – Supports a BR/EDR or LE profile
- Dual mode – Supports BR/EDR and LE profiles

The subsequent sections provide details about the architecture of “Bluetooth LE Protocol Stack” and “Bluetooth BR/EDR Protocol Stack”.

Bluetooth LE Protocol Stack

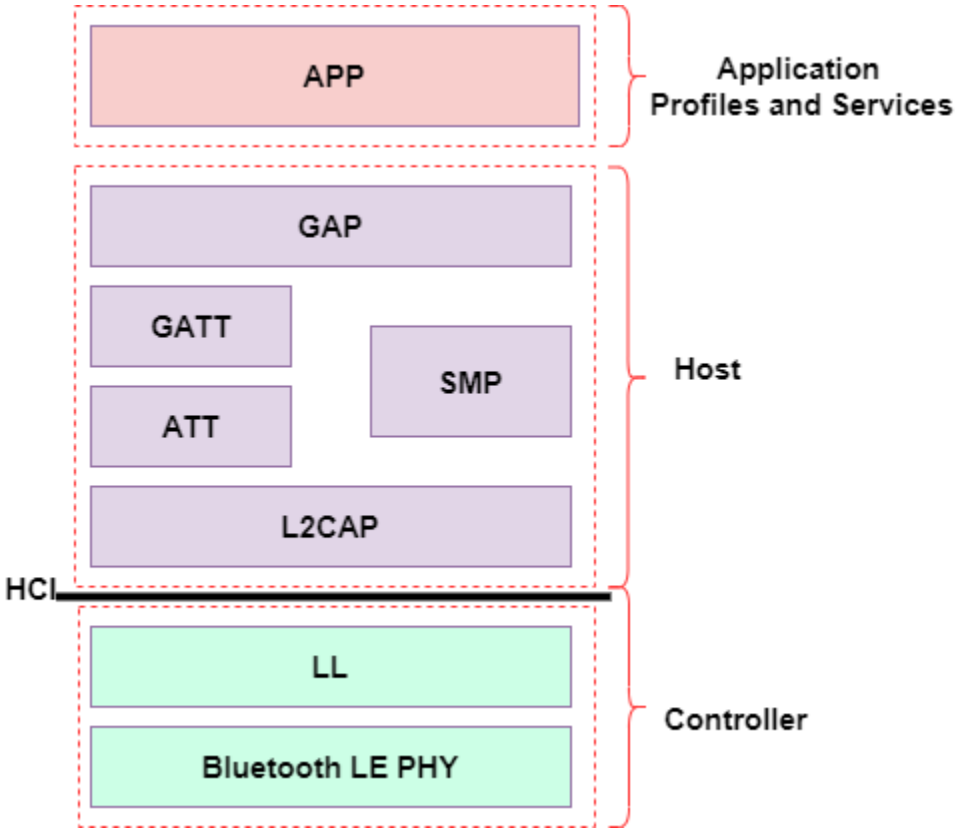
This figure compares the Bluetooth LE protocol stack to the Open System Interconnection (OSI) reference model.



In the preceding figure, the Bluetooth LE protocol stack is shown along with the OSI reference model.

- There is one-to-one mapping at the physical layer (PHY)
- The OSI data link layer (DLL) maps to the Bluetooth LE logical link control and adaptation protocol (L2CAP) and link layer (LL)
- In the Bluetooth LE stack, the higher layers provide application layer services, device roles and modes, connection management, and security protocol

The functionality of the Bluetooth LE protocol stack is divided between three main layers: the Controller, the Host, and Application Profiles and Services.



Controller

The controller layer includes the Bluetooth LE PHY, the LL, and the controller-side host controller interface (HCI).

Bluetooth LE PHY

The Bluetooth LE PHY air interface operates in the same unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) frequency band as Wi-Fi. The Bluetooth LE PHY air interface also includes these characteristics:

- Operating radio frequency (RF) is in the range 2.4000 GHz to 2.4835 GHz, inclusive.
- The channel bandwidth is 2 MHz. The operating band is divided into 40 channels, $k = 0, \ldots, 39$. The center frequency of the kth channel is $2402 + k \times 2$ MHz.
  - User data packets are transmitted using channels in the range [0, 36].
  - Advertising data packets are transmitted in channels 37, 38, and 39.
- Gaussian frequency shift-keying (GFSK) modulation scheme is implemented.
- The Bluetooth LE PHY uses frequency-hopping spread spectrum (FHSS) to reduce interference and to counter the impact of fading channels. The time between frequency hops can vary from 7.5 ms to 4 s and is set at the connection time for each Peripheral.
- Support for throughput at 1 Mbps is mandatory for specification version 4.x compliant devices. At a data rate of 1 Mbps, the transmission is uncoded.
- Optionally, devices compliant with the Bluetooth Core Specification version 5.1 support these additional data rates:

- Coded transmission at bit rates of 500 kbps or 125 kbps
- Uncoded transmission at a bit rate of 2 Mbps

LL

The LL performs tasks similar to the medium access control (MAC) layer of the OSI model. In Bluetooth, the LL interfaces directly with the Bluetooth LE PHY and manages the link state of the radio to define the role of a device as Central, Peripheral, Advertiser, or Scanner.

Controller-Side HCI

The HCI on the controller side handles the interface between the host and the controller. The HCI defines a set of commands and events for transmission and reception of packet data. When receiving packets from the controller, the HCI extracts raw data at the controller to send to the host.

Host

The host includes the host-side HCI, L2CAP, attribute protocol (ATT), generic attribute profile (GATT), security manager protocol (SMP), and generic access profile (GAP).

Host-Side HCI

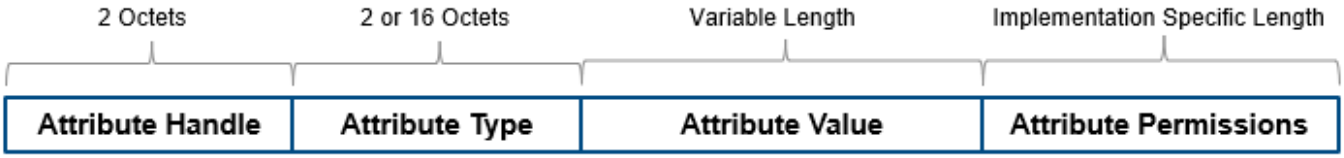
The HCI on the host side handles the interface between the host and the controller. The HCI defines a set of commands and events for transmission and reception of packet data. When transmitting data, the HCI translates raw data into packets to send them from the host to the controller.

L2CAP

The L2CAP encapsulates data from the Bluetooth LE higher layers into the standard Bluetooth LE packet format for transmission or extracts data from the standard Bluetooth LE LL packet on reception according to the link configuration specified at the ATT and SMP layers.

ATT

The ATT transfers attribute data between clients and servers in GATT-based profiles. The ATT defines the roles of the client-server architecture. The roles typically correspond to the Central and the Peripheral as defined in the link layer. In general, a device could be a client, a server, or both, irrespective of whether it is a Central or a Peripheral. The ATT also performs data organization into attributes as shown in this figure.



Device attributes are represented as:

- The attribute handle is a 16-bit identifier value assigned by the server to enable a client to reference those attributes.
- The attribute type is a universally unique identifier (UUID) defined by Bluetooth SIG. For example, UUID 0x2A37 represents a heart-rate measurement.
- The attribute value is a variable length field. The UUID associated with and the service class of the service record containing the attribute value, determine the length of the attribute value field.

- Attribute permissions are sets of permission values associated with each attribute. These permissions specify read and write privileges for an attribute, and the security level required for read and write permission.

GATT

The GATT provides a reference framework for all GATT-based profiles. The GATT encapsulates the ATT and is responsible for coordinating the exchange of profiles in a Bluetooth LE link. Profiles include information and data such as handle assignment, a UUID, and a set of permissions.

For devices that implement the GATT profile,

- The client is the device that initiates commands and requests toward the server. The client can receive responses, indications, and notifications.
- The server is the device that accepts incoming commands and requests from the client. The server sends responses, indications, and notifications to the client.

The GATT uses client-server architecture. The roles are not fixed and are determined when a device initiates a defined procedure. Roles are released when the procedure ends.

The terminology used in the GATT includes:

- Service — A collection of data and associated behaviors used to accomplish a particular function or feature
- Characteristic — A value used in a service along with appropriate permissions
- Characteristic descriptor — A description of the associated characteristic behavior
- GATT-Client — A GATT-Client initiates commands and requests towards the server and can receive responses, indications, and notifications sent by the server
- GATT-Server — A GATT-Server accepts incoming commands and requests from a client and sends responses, indications, and notifications to the client

SMP

The SMP applies security algorithms to encrypt and decrypt data packets. This layer defines the initiator and the responder, corresponding to the Central and the Peripheral, once the connection is established.

GAP

The GAP specifies roles, modes, and procedures of a device. It also manages the connection establishment and security. The GAP interfaces directly with the Application Profiles and Services (App) layer.

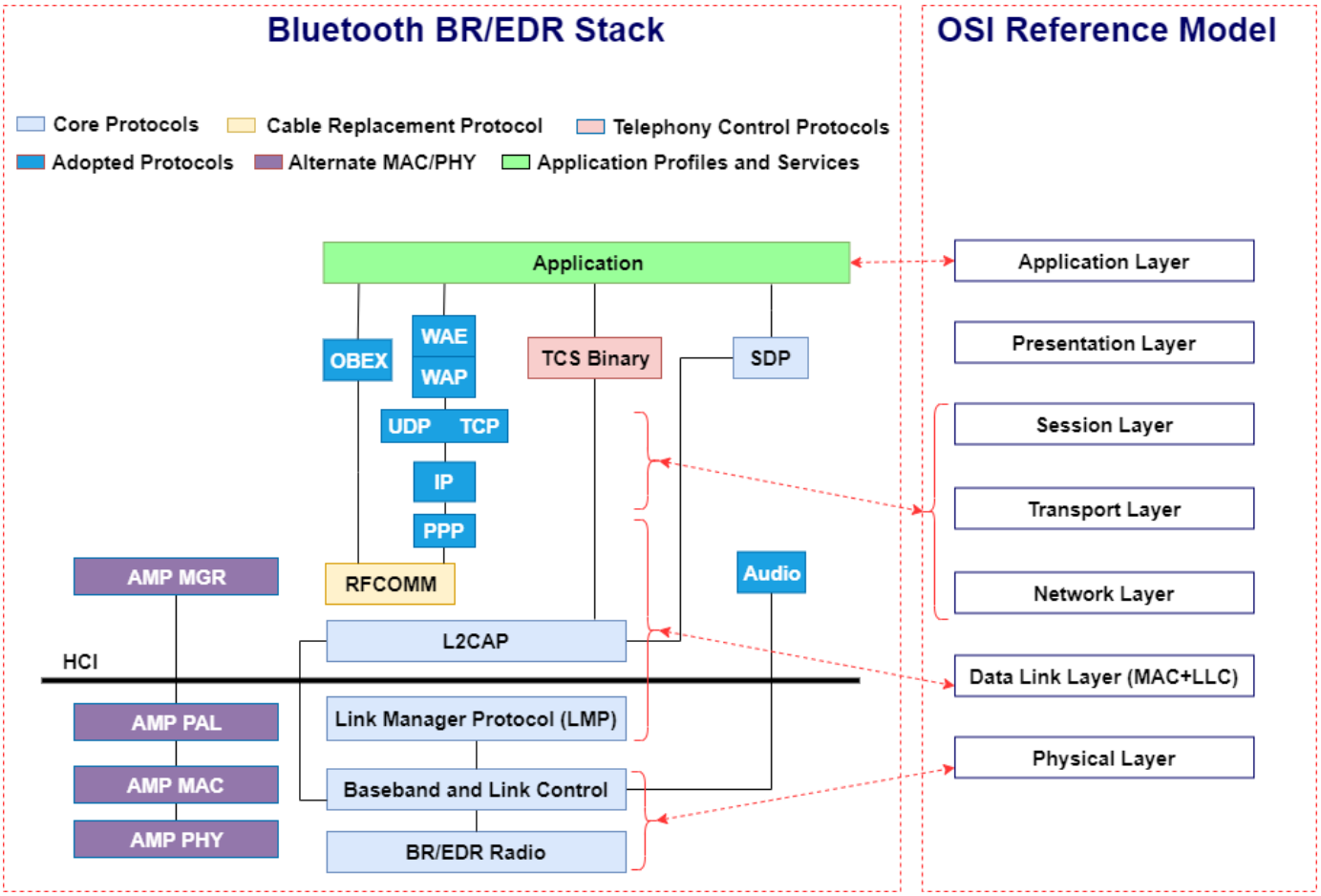
APP Layer

The App layer is the direct user interface defining profiles that afford interoperability between various applications. The Bluetooth core specification enables vendors to define proprietary profiles for use cases not defined by SIG profiles.

Note For more information about the Bluetooth LE protocol stack architecture, see Volume 3, Part C, Sections 2 and 2.1 of the Bluetooth Core Specification [1].

Bluetooth BR/EDR Protocol Stack

This figure compares the block diagram of the Bluetooth BR/EDR protocol stack and with the OSI reference model.



The mapping of BR/EDR stack to the OSI reference model is as shown below:

- The “BR/EDR Radio” and “Baseband and Link Control” layers of the Bluetooth BR/EDR stack map to the OSI PHY layer.
- The “Link Manager Protocol (LMP)”, “L2CAP”, “Cable Replacement Protocol” (RFCOMM), and “PPP” layers of the Bluetooth BR/EDR stack map to the OSI data link layer.
- The user datagram protocol (UDP), transmission control protocol (TCP), and internet protocol (IP) layers of Bluetooth BR/EDR stack map to a combined, network, transport and session layers of the OSI reference model.
- There is one-to-one mapping at the application layer.

Core Protocols

The Bluetooth core protocols and the Bluetooth radio are required by most of the Bluetooth devices. The core protocols include these layers.

BR/EDR Radio

The BR/EDR radio is the lowest defined layer of the Bluetooth specification. The BR mode is mandatory, whereas the EDR mode is optional. This layer defines the requirements of the Bluetooth transceiver device operating in the 2.4 GHz ISM frequency band. It implements a 1600 hops/sec FHSS technique. The radio hops in a pseudo-random way on 79 designated Bluetooth channels. Each Bluetooth channel has a bandwidth of 1 MHz. Each frequency is located at (2402 + k) MHz, where k = 0,1,...78. The modulation technique for BR and EDR mode is GFSK and differential phase shift keying (DPSK), respectively. The baud rate is 1 Msymbols/s. The Bluetooth BR/EDR radio uses the time division duplex (TDD) topology in which data transmission occurs in one direction at one time. The transmission alternates in two directions, one after the other.

Baseband and Link Control

The baseband and link control layer enables the PHY RF link between different Bluetooth devices, forming a piconet. The baseband handles the channel processing and timing, and the link control handles the channel access control. This layer provides these two different types of PHY RF links with their corresponding baseband packets:

- Synchronous connection-oriented (SCO) – Supports real-time audio traffic
- Asynchronous connection-oriented (ACL) – Supports data packet transmission

Link Manager Protocol (LMP)

The LMP layer is primarily responsible for link setup and link configuration between different Bluetooth devices. These processes include establishing security functions such as authentication and encryption by generating, exchanging, and checking link and encryption keys. Furthermore, this layer controls the power modes and duty cycles of the Bluetooth radio device and the connection states of a Bluetooth unit in a piconet.

L2CAP

The L2CAP adapts higher-layer protocols over the baseband. It shields the higher-layer protocols from the details of the lower-layer protocols. The L2CAP provides connection-oriented and connectionless services to the higher-layer protocols. This includes protocol multiplexing capability, segmentation and reassembly operations, and group abstractions.

SDP

Discovery services are an important aspect of the Bluetooth framework. The service discovery protocol (SDP) provides a means for applications to query services and the characteristics of services, following which a connection can be established between two or more Bluetooth devices. The SDP is quite different from service discovery in traditional network-based environments. The SDP is built on top of the L2CAP.

Cable Replacement Protocol

The cable replacement protocol in the Bluetooth BR/EDR stack uses RFCOMM to provide emulation of serial ports over L2CAP. RFCOMM emulates RS-232 control and data signals over the Bluetooth baseband and provides transport capabilities for higher-layer services that use a serial interface as a transport mechanism. RFCOMM also provides multiple simultaneous connections to one device and enables connections to multiple devices.

Telephony Control Protocols

The telephony control protocol specification, binary (TCS binary), defines the call control signaling to establish data and voice calls between Bluetooth devices. It is built on top of the L2CAP. Moreover, TCS binary defines mobility management procedures for handling Bluetooth devices.

Adopted Protocols

In addition to the core protocols, the Bluetooth BR/EDR stack includes protocols adopted from other standard bodies. These adopted protocols are defined in specifications issued by other standard-making organizations and are incorporated into the Bluetooth framework.

PPP

The point-to-point protocol (PPP) is an Internet Engineering Task Force (IETF) [3] standard protocol for transporting IP datagrams over a point-to-point link. The PPP runs over the RFCOMM to realize point-to-point connections.

TCP, UDP, and IP

These layers are the IETF-defined foundation protocols of the TCP/IP protocol suite.

- TCP – This protocol provides a reliable virtual connection between devices to realize data communication. The TCP treats the data as a stream of bytes and transmits them without any errors or duplication.
- UDP – This protocol is an alternative to the TCP and provides an unreliable datagram connection between devices. As there is no end-to-end connection in UDP, data is transmitted link-by-link without any guarantee of service.
- IP – This layer is a network layer protocol that enables a datagram service between devices, supporting both the TCP and UDP.

The use of the TCP, UDP, and IP in the Bluetooth BR/EDR stack enables communication with any other device connected to the Internet.

OBEX

The object exchange (OBEX) protocol is a session-level protocol developed by the Infrared Data Association (IrDA) to exchange objects. The OBEX protocol provides functionality similar to that of HTTP, but in a simpler manner. HTTP is an application layer protocol and layered above the TCP/IP. The OBEX protocol provides the client with a reliable transport for connecting to a server. It also provides a model for representing objects and operations.

WAE and WAP

Bluetooth BR/EDR stack incorporates the wireless application environment (WAE) and wireless application protocol (WAP) into its architecture. The advantages of using WAE/WAP features in the Bluetooth stack are:

- Build application gateways that act as an interface between WAP servers and some other application on the PC
- Provide functions such as remote control and data fetching from the PC to the Bluetooth handset
- Reuse the upper software application developed for the WAP application environment

Application Profiles and Services

For more information, see “APP Layer”.

Alternate MAC/PHY

The alternate MAC/PHY (AMP) manager is a secondary controller in the Bluetooth core system. After an L2CAP connection is established between two devices over the BR/EDR radio, the AMP manager can discover the AMPs that are available on the other device. If an AMP is common between two devices, the Bluetooth core system provides mechanisms for moving data traffic from the BR/EDR controller to an AMP controller.

Each AMP manager consists of a protocol adaptation layer (PAL) on top of a MAC and PHY. The PAL maps the Bluetooth protocols to the specific protocols of the underlying MAC and PHY.

L2CAP channels can be created on, or moved to, an AMP. If an AMP physical link has a link supervision timeout, then L2CAP channels can be moved back to BR/EDR radio. To minimize power consumption in the device, AMPs are enabled or disabled as required.

HCI

The HCI provides a command interface to the BR/EDR radio, baseband controller, and the link manager. It is a single standard interface for accessing the Bluetooth baseband capabilities, the hardware status, and the control registers.

Note For more information about the Bluetooth BR/EDR protocol stack architecture, see Volume 1, Part A, Sections 2 and 2.1 of the Bluetooth Core Specification [1].

References

[1] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

[2] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed November 6, 2019. https://www.bluetooth.com/.

[3] IETF. “Internet Standards.” Accessed November 6, 2019. https://www.ietf.org/.

[4] Bluetooth Protocol Stack - an Overview | ScienceDirect Topics. Accessed November 15, 2019. https://www.sciencedirect.com/.

See Also

More About

- “Bluetooth Technology Overview”
- “Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications”
- “Bluetooth Packet Structure”

Bluetooth Packet Structure

The Bluetooth Special Interest Group (SIG) [1] and [2] defines different packet structures for Bluetooth low energy (LE) and Bluetooth basic rate/enhanced data rate (BR/EDR) devices.

Bluetooth LE Packet Structure

Bit Ordering in Bluetooth LE Packets

When defining packets and messages in the baseband specification, the bit ordering follows the little-endian format. In this format, these rules apply:

- The least significant bit (LSB) corresponds to b_0.
- LSB is the first bit sent over the air.
- When illustrating the packet structure, the LSB is shown on the left side.

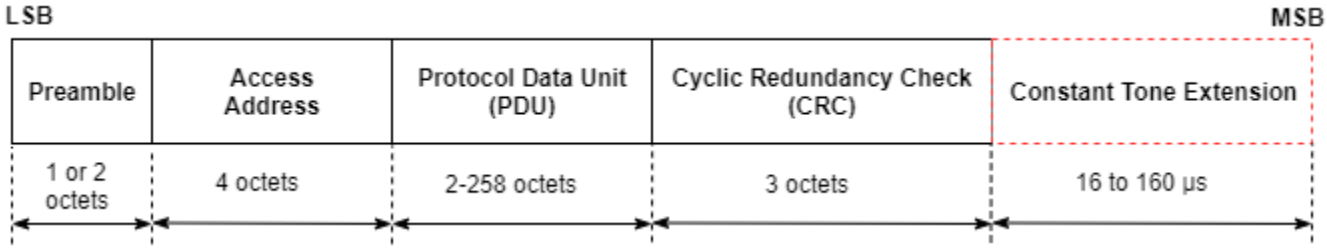
Moreover, data fields generated internally at the baseband level (packet header and payload header length), must be transmitted with the LSB first. For example, a 3-bit parameter is sent as: $b_0b_1b_2$ = 110 over the air, where 1 is sent first and 0 is sent last.

The Bluetooth LE devices use packet formats for: “Bluetooth LE Uncoded Physical Layer (PHY)”, “Bluetooth LE Coded PHY”, “Advertising Physical Channel PDU”, “Data Physical Channel PDU”, and “Constant Tone Extension and In-Phase Quadrature (IQ) Sampling”.

Note For more information about Bluetooth LE packet structure, see Volume 6, Part B, Section 2 of the Bluetooth Core Specification [2].

Bluetooth LE Uncoded Physical Layer (PHY)

The Bluetooth Core Specification [2] defines two physical layer (PHY) transmission modes (LE 1M and LE 2M) for uncoded PHY. This figure shows the packet structure for the Bluetooth LE uncoded PHY operating on LE 1M and LE 2M.



Each packet contains four mandatory fields (preamble, access-address, protocol data unit (PDU), and cyclic redundancy check (CRC)) and one optional field (constant tone extension (CTE)). The preamble is transmitted first, followed by the access address, PDU, CRC, and CTE (if present) in that order. The entire packet is transmitted at the same symbol rate of 1 Msym/s or 2 Msym/s.

Preamble

All link layer (LL) packets contain a preamble, which is used in the receiver to perform frequency synchronization, automatic gain control (AGC) training, and symbol timing estimation. The preamble

is a fixed sequence of alternating 0 and 1 bits. For the Bluetooth LE packets transmitted on the LE 1M PHY and LE 2M PHY, the preamble size is 1 octet and 2 octets, respectively.

Access address

The access address is a 4-octet value. Each LL connection between any two devices and each periodic advertising train has a distinct access address. Each time the Bluetooth LE device needs a new access address, the LL generates a new random value adhering to these requirements:

- The value must not be the access address for any existing LL connection on this device.
- The value must not be the access address for any enabled periodic advertising train.
- The value must have no more than six successive 1s or 0s.
- The value must not be the access address for any advertising channel packets.
- The value must not be a sequence that differs from the access address of advertising physical channel packets by only 1 bit.
- All four octets for the value must not be equal.
- The value must have a minimum of two transitions in the most significant 6 bits.

If the random value does not satisfy the above requirements, a new random value is generated until it meets all of the requirements.

PDU

When a Bluetooth LE packet is transmitted on either the primary or secondary advertising physical channel or the periodic physical channel, the PDU is defined as the “Advertising Physical Channel PDU”. When a packet is transmitted on the data physical channel, the PDU is defined as the “Data Physical Channel PDU”.

CRC

The size of the CRC is 3 octets and is calculated on the PDU of all LL packets. If the PDU is encrypted, then the CRC is calculated after encryption of the PDU is complete. The CRC polynomial has the form $x^{24}+x^{10}+x^9+x^6+x^4+x^3+x+1$.

For more information about CRC generation, see Volume 6, Part B, Section 3.1.1 of the Bluetooth Core Specification [2].

CTE

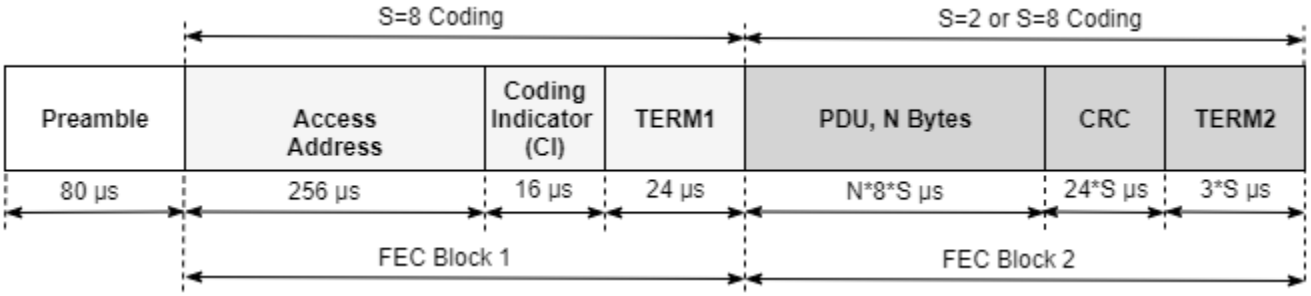
The CTE consists of a constantly modulated series of unwhitened 1s. This field has a variable length that ranges from 16 µs to 160 µs.

For more information about the CTE, see Volume 6, Part B, Section 2.5.1 of the Bluetooth Core Specification [2].

Note For more information about Bluetooth LE uncoded PHY packet structure, refer to Volume 6, Part B, Section 2.1 of Bluetooth Core Specification [2].

Bluetooth LE Coded PHY

This figure shows the packet structure for the Bluetooth LE coded PHY and is implemented for Bluetooth LE packets on all the physical channels.



Each Bluetooth LE packet consists of a preamble and these two forward error correcting (FEC) blocks:

- FEC block 1— This block contains three fields: access address, coding indicator (CI), and TERM1. This block implements an *S*=8 coding scheme, where each bit represents eight symbols. This gives a data rate of 125 Kbps.
- FEC block 2— This block contains these three fields: PDU, CRC, and TERM2. This block implements an *S*=8 or *S*=2 coding scheme. In the *S*=2 coding scheme, each bit represents two symbols. Therefore, the data rate is 500 Kbps.

The Bluetooth LE coded PHY does not contain the CTE.

Preamble

The Bluetooth LE coded PHY preamble is 80 symbols in length and contains 10 repetitions of the symbol pattern '00111100' (in the transmission order).

Access address

The length of Bluetooth LE coded PHY access address is 256 symbols. For more information, see “Access address”. In addition to the requirements listed in the access address subsection of the “Bluetooth LE Uncoded Physical Layer (PHY)” section, the new value for the access address of the Bluetooth LE coded PHY must also meet these requirements:

- The value must have at least three 1s in the last significant bits.
- The value must have no more than 11 transitions in the least significant 16 bits.

CI

The CI field consists of two bits as shown in this table:

Bits in CI	Description
`00b`	FEC block 2 coded using *S*=8
`01b`	FEC block 2 coded using *S*=2
All other values	Reserved for future use

PDU

The PDU in the Bluetooth LE coded PHY packet structure has the same formatting as the “PDU” in the Bluetooth LE uncoded PHY packet.

CRC

The CRC in the Bluetooth LE coded PHY packet structure has the same formatting as the “CRC” in the Bluetooth LE uncoded PHY packet.

TERM1 and TERM2

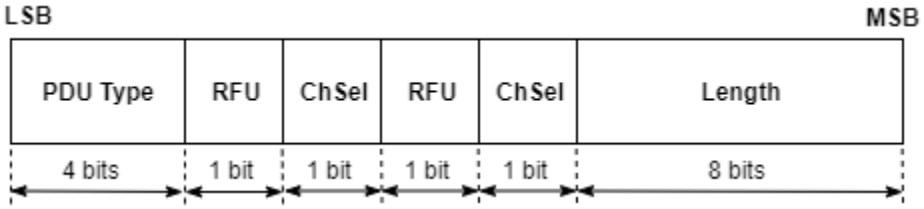
Each FEC block contains a terminator at the end of the block. That terminator is referred to as TERM1 and TERM2. Each terminator is 3 bits long and forms the termination sequence during the FEC encoding process.

Note For more information about Bluetooth LE coded PHY packet structure, see Volume 6, Part B, Section 2.2 of the Bluetooth Core Specification [2].

Advertising Physical Channel PDU

The packet structure format of the advertising physical channel PDU is shown in this figure.

The advertising physical channel PDU has a 16-bit header and a variable-size payload. The 16-bit header field of the advertising physical channel PDU is shown in this figure.



The PDU type field in the advertising channel PDU header defines different types of PDUs that can be transmitted on the Bluetooth LE coded PHY. This table maps different types of PDUs with the physical channels and the PHYs on which the Bluetooth LE packet might appear. The table also indicates the PHY transmission modes supported for each type of advertising physical channel PDU.

PDU Type	PDU Name	Physical Channel	LE 1M Support	LE 2M Support	LE Coded Support
0000b	ADV_IND	Primary Advertising	Yes		
0001b	ADV_DIRECT_IND	Primary Advertising	Yes		
0010b	ADV_NONCONN_IND	Primary Advertising	Yes		
0011b	SCAN_REQ	Primary Advertising	Yes		
	AUX_SCAN_REQ	Secondary Advertising	Yes	Yes	Yes

PDU Type	PDU Name	Physical Channel	LE 1M Support	LE 2M Support	LE Coded Support
0100b	SCAN_RSP	Primary Advertising	Yes		
0101b	CONNECT_IND	Primary Advertising	Yes		
	AUX_CONNECT_REQ	Secondary Advertising	Yes	Yes	Yes
0110b	ADV_SCAN_IND	Primary Advertising	Yes		
0111b	ADV_EXT_IND	Primary Advertising	Yes		Yes
	AUX_ADV_IND	Secondary Advertising	Yes	Yes	Yes
	AUX_SCAN_RSP	Secondary Advertising	Yes	Yes	Yes
	AUX_SYNC_IND	Periodic	Yes	Yes	Yes
	AUX_CHAIN_IND	Secondary Advertising and Periodic	Yes	Yes	Yes
1000b	AUX_CONNECT_RSP	Secondary Advertising	Yes	Yes	Yes
All other values	Reserved for future use				

The RFU field is reserved for future use. The `ChSel`, `TxAdd`, and `RxAdd` fields of the advertising physical channel PDU header contain information specific to the type of PDU defined for each advertising physical channel PDU separately. If the `ChSel`, `TxAdd`, or `RxAdd` fields are not defined as used in a given PDU, then they are considered as reserved for future use.

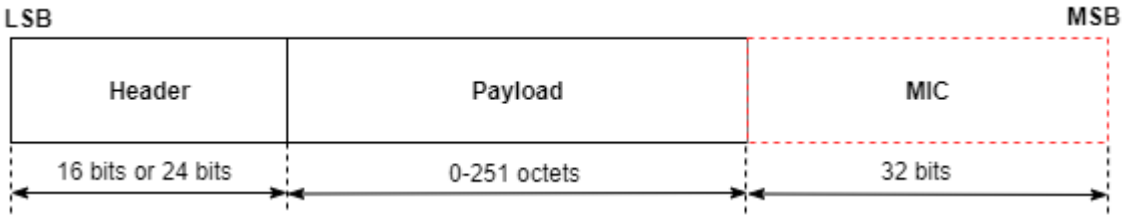
The `Length` field of the advertising physical channel PDU header denotes the length of the payload in octets. The valid range of the Length field is 1 to 255 octets.

The `Payload` field in the advertising physical channel PDU packet structure is specific to the type of PDUs listed in the preceding table.

Note For more information about advertising physical channel PDUs, see Volume 6, Part B, Section 2.3 of the Bluetooth Core Specification [2].

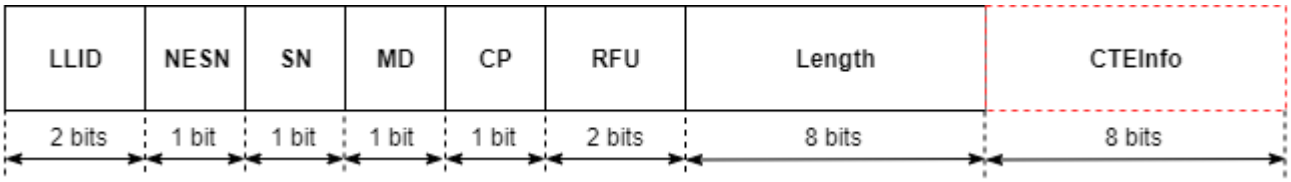
Data Physical Channel PDU

The packet structure format of the data physical channel PDU is shown in this figure.



The data physical channel PDU has a 16-bit or 24-bit header, a variable length payload in the range [0, 251] octets, and can include a 32-bit message integrity check (MIC) field. The MIC is not included in an unencrypted LL connection or in an encrypted LL connection with a data channel PDU containing an empty payload. The MIC is included in an encrypted LL connection with a data channel PDU containing a nonzero length payload. In this case, the MIC is calculated as specified in Volume 6, Part E, Section 1 of the Bluetooth Core Specification [2].

The header field of the data physical channel PDU is shown in this figure.



The data physical channel PDU header includes these fields:

- Link layer identifier (LLID) — This field indicates whether the packet is an LL data PDU or LL control PDU.
  - `00b` — Reserved for future use
  - `01b` — LL Data PDU, which can be a continuation fragment of an logical link control and adaptation (L2CAP) message or an empty PDU
  - `10b` — LL Data PDU, which can be a start of an L2CAP message or a complete L2CAP message with no fragmentation
  - `11b` — LL control PDU
- Next expected sequence number (NESN): The LL uses this field to either acknowledge the last data physical channel PDU sent by the peer or to request the peer to resend the last data physical channel PDU. For more information about NESN, see Volume 6, Part B, Section 4.5.9 of the Bluetooth Core Specification [2].
- Sequence number (SN): The LL uses this field to identify the Bluetooth LE packets sent by it. For more information about the SN, see Volume 6, Part B, Section 4.5.9 of the Bluetooth Core Specification [2].
- More data (MD): This field indicates that the Bluetooth LE device has more data to send. If neither of Central and Peripheral Bluetooth LE device has set the MD bit in their packets, the packet from the Peripheral closes the connection event. If the Central and Peripheral devices have set the MD bit, the Central can continue the connection event by sending another packet, and the Peripheral must listen after sending its packet. For more information about MD, see volume 6, Part B, Section 4.5.6 of the Bluetooth Core Specification [2].
- CTEInfo present (CP): This field indicates whether the data physical channel PDU header has a CTEInfo field and, subsequently whether the data physical channel packet has a CTE. For more information about the packet structure of the CTEInfo field, see Volume 6, Part B, Section 2.5.2 of the Bluetooth Core Specification [2].
- Length: This field indicates the size, in octets, of the payload and MIC, if present. The size of this field is in the range [0, 255] octets.
- CTEInfo: This field indicates the type and length of the CTE.

The two types of data physical channel PDUs are: “LL Data PDU” and “LL Control PDU”.

LL Data PDU

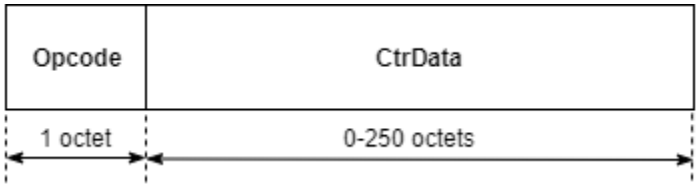
The LL uses the LL data PDU to send L2CAP data. The LLID field in the LL data channel PDU header is set to either `01b` or `10b`. An LL data PDU is referred to as an empty PDU if

- The LLID field of the LL data channel PDU header is set to `01b`.
- The Length field of the LL data channel PDU header is set to `00000000b`.

An LL data PDU with the LLID field in the header set to `10b` does not have the Length field set to `00000000b`.

LL Control PDU

The LL uses the LL data PDU to control the LL connection. If the LLID field of data physical channel PDU header is set to `11b`, the data physical channel PDU contains an LL control PDU. This figure shows the LL control PDU payload.



The Opcode field defines different types of LL control PDUs as shown in this table.

Opcode	LL Control PDU
0x00	LL_CONNECTION_UPDATE_IND
0x01	LL_CHANNEL_MAP_IND
0x02	LL_TERMINATE_IND
0x03	LL_ENC_REQ
0x04	LL_ENC_RSP
0x05	LL_START_ENC_REQ
0x06	LL_START_ENC_RSP
0x07	LL_UNKNOWN_RSP
0x08	LL_FEATURE_REQ
0x09	LL_FEATURE_RSP
0x0A	LL_PAUSE_ENC_REQ
0x0B	LL_PAUSE_ENC_RSP
0x0C	LL_VERSION_IND
0x0D	LL_REJECT_IND
0x0E	LL_SLAVE_FEATURE_REQ
0x0F	LL_CONNECTION_PARAM_REQ
0x10	LL_CONNECTION_PARAM_RSP
0x11	LL_REJECT_EXT_IND
0x12	LL_PING_REQ
0x13	LL_PING_RSP
0x14	LL_LENGTH_REQ
0x15	LL_LENGTH_RSP
0x16	LL_PHY_REQ
0x17	LL_PHY_RSP

Opcode	LL Control PDU
0x18	LL_PHY_UPDATE_IND
0x19	LL_MIN_USED_CHANNELS_IND
0x1A	LL_CTE_REQ
0x1B	LL_CTE_RSP
0x1C	LL_PERIODIC_SYNC_IND
0x1D	LL_CLOCK_ACCURACY_REQ
0x1E	LL_CLOCK_ACCURACY_RSP
All other values	Reserved for future use

The CtrData field in the LL control PDU is specific to the value of the Opcode field. For more information about different LL control PDUs and their corresponding CtrData field structure, see Volume 6, Part B, Sections 2.4.2.1 to 2.4.2.28 of the Bluetooth Core Specification [2].

Constant Tone Extension and In-Phase Quadrature (IQ) Sampling

The length of the CTE is variable and in the range [16, 160] µs. This field contains a constantly modulated series of 1s with no whitening applied. The CTE is of two types: antenna switching during CTE transmission (AoD) and antenna switching during CTE reception (AoA). When receiving a packet containing an AoD CTE, the receiver does not need to switch antennae. When receiving a packet containing an AoA CTE, the receiver performs antenna switching according to the switching pattern configured by the host. In both cases, the receiver takes an IQ sample at each microsecond during the reference period and an IQ sample each sample slot. The controller reports the IQ samples to the host. The receiver samples the entire CTE regardless of its length, unless this conflicts with other activities. For more information about CTE, see Volume 6, Part B, Sections 2.5.1 to 2.5.3 of the Bluetooth Core Specification [2].

When requested by the host, the receiver performs IQ sampling when receiving a valid Bluetooth LE packet with a CTE. However, when receiving a Bluetooth LE packet with a CTE and an incorrect CRC, the receiver might perform IQ sampling. For more information about IQ sampling, see Volume 6, Part B, Section 2.5.4 of the Bluetooth Core Specification [2].

Note For more information about data physical channel PDUs, see Volume 6, Part B, Section 2.4 of the Bluetooth Core Specification [2].

Bluetooth BR/EDR Packet Structure

Bit Ordering in Bluetooth BR/EDR Packets

The bit ordering in Bluetooth BR/EDR packets follows the same format as the “Bit Ordering in Bluetooth LE Packets”.

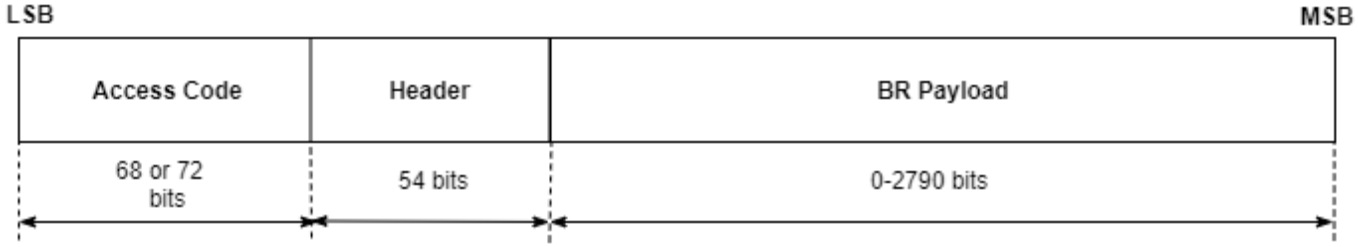
Bluetooth BR/EDR devices use packet formats for: “BR Mode”, “EDR Mode”, “Access Code”, “Packet Header”, “Packet Types”, and “Payload Format”.

Note For more information about Bluetooth BR/EDR packet structure, see Volume 2, Part B, Section 6 of the Bluetooth Core Specification [2].

General Format

BR Mode

The general format of Bluetooth BR packets is shown in this figure. Each packet consists of these fields: the access code (68 or 72 bits), header (54 bits), and payload in the range [0, 2790] bits.

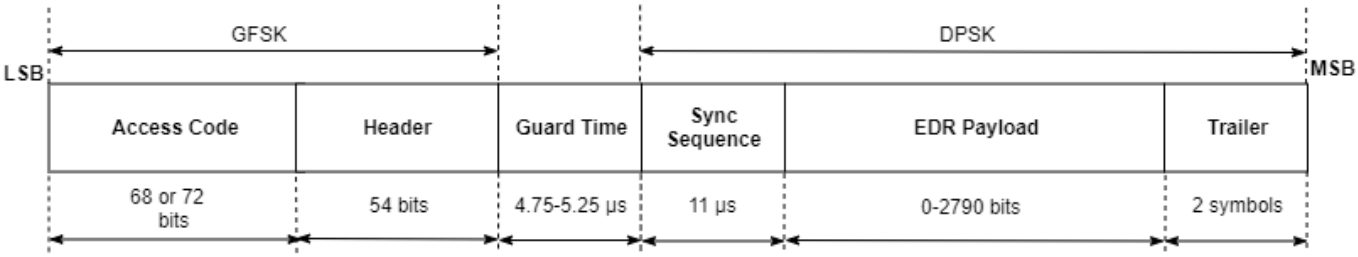


The Bluetooth Core Specification [2] defines different types of packets. A packet can consist of:

- The shortened access code only
- The access code and the packet header
- The access code, the packet header, and the payload

EDR Mode

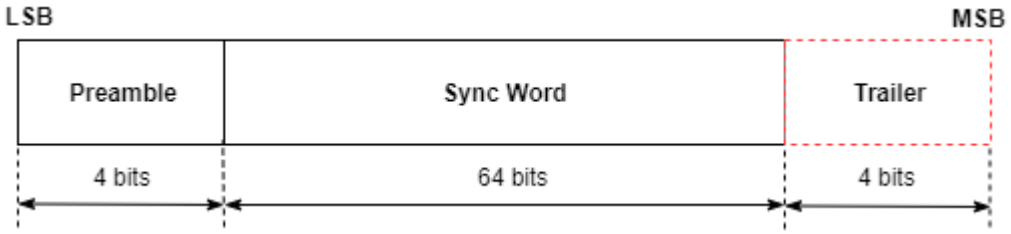
The general format of Bluetooth EDR packets is shown in this figure.



The format and modulation of the access code and the packet header fields are similar to that of BR packets. Following the header field, the EDR packets have a guard time in the range [4.75, 5.25] µs, a sync sequence (11 µs), payload in the range [0, 2790] bits, and trailer (two symbols) fields.

Access Code

Each packet starts with an access code. If a packet header follows, the access code is 72 bits long. Otherwise, the length of the access code is 68 bits. In this case, the access code is referred to as a shortened access code. The shortened access code does not contain a trailer. The access code is used for synchronization, DC offset compensation, and identification of all packets exchanged on the physical channel. The shortened access code is used in paging and inquiry. In this case, the access code itself is used as a signaling message, and neither a header nor a payload is present. This figure shows the packet structure of the access code.



Different access code types use different lower address parts (LAPs) to construct the sync word. A summary of different access code types is shown in this table.

Access Code Type	LAP	Access Code Length (Bits)	Description
Channel access code (CAC)	Central	72	This access code is used in the connection state, synchronization train substate, and synchronization scan substate. It is derived from the LAP of the Central's `BD_ADDR` .
Device access code (DAC)	Paged device	68 or 72	This access code is used during page, page scan, and page response substates. It is derived from the paged devices's `BD_ADDR`.
Dedicated inquiry access code (DIAC)	Dedicated	68 or 72	This access code is used in the inquiry substate for dedicated inquiry operations.
General inquiry access code (GIAC)	Reserved	68 or 72	This access code is used in the inquiry substate for general inquiry operations.

For DAC, DIAC, and GIAC access code types, the access code length of 72 bits is used only in combination with frequency hopping sequence (FHS) packets. When used as self-contained messages without a header, the DAC, DIAC and GIAC do not include trailer bits and are of length 68 bits.

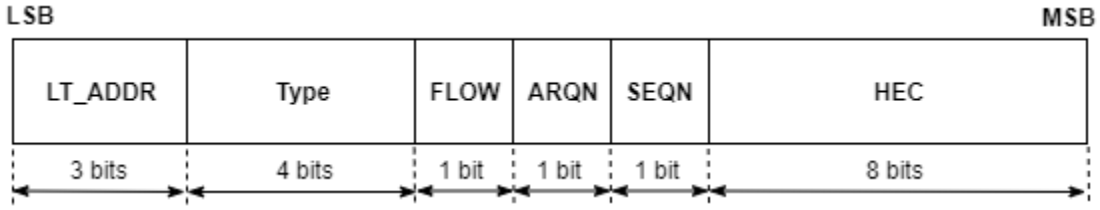
The CAC consists of a preamble, sync word, and trailer.

- Preamble: It is a fixed 4-symbol pattern of 1s and 0s that facilitates DC compensation. If the LSB of the following sync word is `1` or `0`, the preamble sequence is `1010` or `0101` (in transmission order), respectively.
- Sync word: It is a 64-bit code word derived from a 24-bit LAP address. The construction guarantees a large Hamming distance between sync words based on different LAPs. The autocorrelation properties of the sync word improve timing acquisition.
- Trailer: It is appended to the sync word as soon as the packet header follows the access code. The trailer is a fixed 4-symbol pattern of 1s and 0s. The trailer together with the three MSBs of the sync word form a 7-bit pattern of alternating 1s and 0s which is used for extended DC compensation. The trailer sequence is either `1010` or `0101` (in transmission order) depending on whether the MSB of the sync word is `0` or `1`, respectively.

Note For more information about access code in Bluetooth BR/EDR, see Volume 2, Part B, Section 6.3 of the Bluetooth Core Specification [2].

Packet Header

The structure of the Bluetooth BR/EDR packet header is shown in this figure.



This table provides a brief description about the packet header fields.

Packet Header Field	Size of the Field (Bits)	Description
Logical transport address (LT_ADDR)	3	This field indicates the destination Peripheral(s) for a packet in a Central-to-Peripheral transmission slot and indicates the source Peripheral for a Peripheral-to-Central transmission slot.
Type	4	This field specifies the type of packet used. The Bluetooth Core Specification [2] defines 16 different types of BR/EDR packets. The value in this field depends on the value of LT_ADDR field in the packet. This field determines the number of slots occupied by the current packet.
Flow control (FLOW)	1	This field implements the flow control of BR/EDR packets over the asynchronous connection-oriented logical (ACL) transport. When the receive buffer for the ACL logical transport is full, a 'STOP' indication (FLOW = 0) is returned to stop the other device from transmitting data temporarily. When the receive buffer can accept data, a 'GO' indication (FLOW = 1) is returned.
Automatic repeat request number (ARQN)	1	This field informs the source of a successful transfer of payload data with the CRC. This field is reserved for future use on the connectionless Peripheral broadcast (CSB) logical transport.
Sequence number (SEQN)	1	This field provides a sequential numbering scheme to order the data packet stream. This field is reserved for future use on the CSB logical transport.

Packet Header Field	Size of the Field (Bits)	Description
Header error check (HEC)	8	This field checks the packet header integrity. Before generating the HEC, the HEC generator is initialized with an 8-bit value. These 8 bits correspond to the upper address part (UAP). After the initialization, the HEC generator calculates the HEC value for the 10 header bits. Before checking the HEC, the receiver initializes the HEC check circuitry with the appropriate 8-bit UAP. If the HEC does not check the packet header integrity, the entire packet is discarded.

Note For more information about packet header used in Bluetooth BR/EDR, see Volume 2, Part B, Section 6.4 of the Bluetooth Core Specification [2].

Packet Types

The packets used in the piconet are related to these logical transports on which they are used.

- Synchronous connection-oriented (SCO): It is a circuit-switched connection that reserved slots between the Central and a specific Peripheral.
- Extended SCO (eSCO): Similar to SCO, it reserves slots between the Central and a specific Peripheral. eSCO supports a retransmission window following the reserved slots. Together, the reserved slots and the retransmission window form the complete eSCO window.
- ACL: It provides a packet-switched connection between the Central and all active Peripherals participating in the piconet. ACL supports asynchronous and isochronous services. Between a Central and a Peripheral, only a single ACL logical transport must exist.
- CSB: It is used to transport profile broadcast data from a Central to multiple Peripherals. A CSB logical transport is unreliable.

This table summarizes the packets defined for the SCO, eSCO, ACL, and CSB logical transport types.

Note The column entries followed by "D" means data field only. "C.1" implies that the MIC value is mandatory when encryption with AES-CCM is enabled. Otherwise, MIC is excluded. For more information about different packet types used in Bluetooth BR/EDR, see Volume 2, Part B, Sections 6.5 and 6.7 of the Bluetooth Core Specification [2].

Packet Type	TYPE Code	Slot Occupancy	Payload Header (Bytes)	User Payload (Bytes)	FEC	MIC	CRC	Logical Transport Types Supported
ID	N/A	1	N/A	N/A	N/A	N/A	N/A	N/A
NULL	0000	1	N/A	N/A	N/A	N/A	N/A	SCO, eSCO, ACL, CSB

Packet Type	TYPE Code	Slot Occupancy	Payload Header (Bytes)	User Payload (Bytes)	FEC	MIC	CRC	Logical Transport Types Supported
POLL	0001	1	N/A	N/A	N/A	N/A	N/A	SCO, eSCO, ACL
FHS	0010	1	N/A	18	2/3	N/A	Yes	SCO, ACL
DM1	0011	1	1	0-17	2/3	C.1	Yes	SCO, ACL, CSB
DH1	0100	1	1	0-27	No	C.1	Yes	ACL, CSB
DM3	1010	3	2	0-121	2/3	C.1	Yes	ACL, CSB
DH3	1011	3	2	0-183	No	C.1	Yes	ACL, CSB
DM5	1110	5	2	0-224	2/3	C.1	Yes	ACL, CSB
DH5	1111	5	2	0-339	No	C.1	Yes	ACL, CSB
2-DH1	0100	1	2	0-54	No	C.1	Yes	ACL, CSB
2-DH3	1010	3	2	0-367	No	C.1	Yes	ACL, CSB
2-DH5	1110	5	2	0-679	No	C.1	Yes	ACL, CSB
3-DH1	1000	1	2	0-83	No	C.1	Yes	ACL, CSB
3-DH3	1011	3	2	0-552	No	C.1	Yes	ACL, CSB
3-DH5	1111	5	2	0-1021	No	C.1	Yes	ACL, CSB
HV1	0101	1	N/A	10	1/3	No	No	SCO
HV2	0110	1	N/A	20	2/3	No	No	SCO
HV3	0111	1	N/A	30	No	No	No	SCO
DV	1000	1	1 D	10+(0-9) D	2/3 D	No	Yes D	SCO
EV3	0111	1	N/A	1-30	No	No	Yes	eSCO
EV4	1100	3	N/A	1-120	2/3	No	Yes	eSCO
EV5	1101	3	N/A	1-180	No	No	Yes	eSCO
2-EV3	0110	1	N/A	1-60	No	No	Yes	eSCO
2-EV5	1100	3	N/A	1-360	No	No	Yes	eSCO
3-EV3	0111	1	N/A	1-90	No	No	Yes	eSCO
3-EV5	1101	3	N/A	1-540	No	No	Yes	eSCO

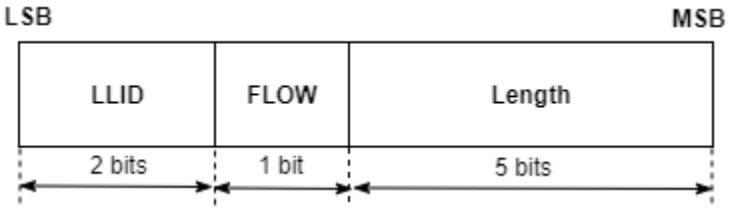
Payload Format

The Bluetooth Core Specification [2] defines two types of payload field formats: synchronous data field (for ACL packets) and asynchronous data field (for SCO and eSCO packets). However, the DV packets contain both the synchronous and asynchronous data fields.

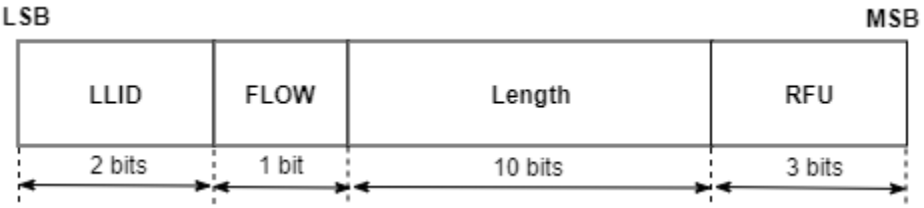
- Synchronous data field: In SCO, which supports only the BR mode, the length of the synchronous data field is fixed. The synchronous data field contains only the synchronous data body portion and does not have a payload header. In BR eSCO, the synchronous data field consists of these two segments: a synchronous data body and a CRC code. In this case, no payload header is present. In

EDR eSCO, the synchronous data field consists of a guard time, synchronization sequence, synchronous data body, CRC code, and trailer. In this case, no payload header is present.

- Asynchronous data field: The BR ACL packets have an asynchronous data field consisting of payload header, payload body, MIC (if applicable), and CRC (if applicable). This figure shows the 8-bit payload header format for BR single-slot ACL packets.



EDR ACL packets have an asynchronous data field consisting of guard time, synchronization sequence, payload header, payload body, MIC (if applicable), CRC (if applicable), and trailer. This figure shows the 16-bit payload header format for EDR multislot ACL packets.



Note For more information about the payload format, see Volume 2, Part B, Sections 6.6.1 and 6.6.2 of the Bluetooth Core Specification [2].

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed November 22, 2019. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

More About

- “Bluetooth Technology Overview”
- “Comparison of Bluetooth BR/EDR and Bluetooth LE Specifications”
- “Bluetooth Protocol Stack”
- “Bluetooth Packet Structure”
- “Generate and Decode Bluetooth Protocol Data Units”

Tutorials

Generate Wireless Waveform in Simulink Using App-Generated Block

This example shows how to configure and use the block that is generated using the **Export to Simulink** capability that is available in the **Wireless Waveform Generator** app.

Introduction

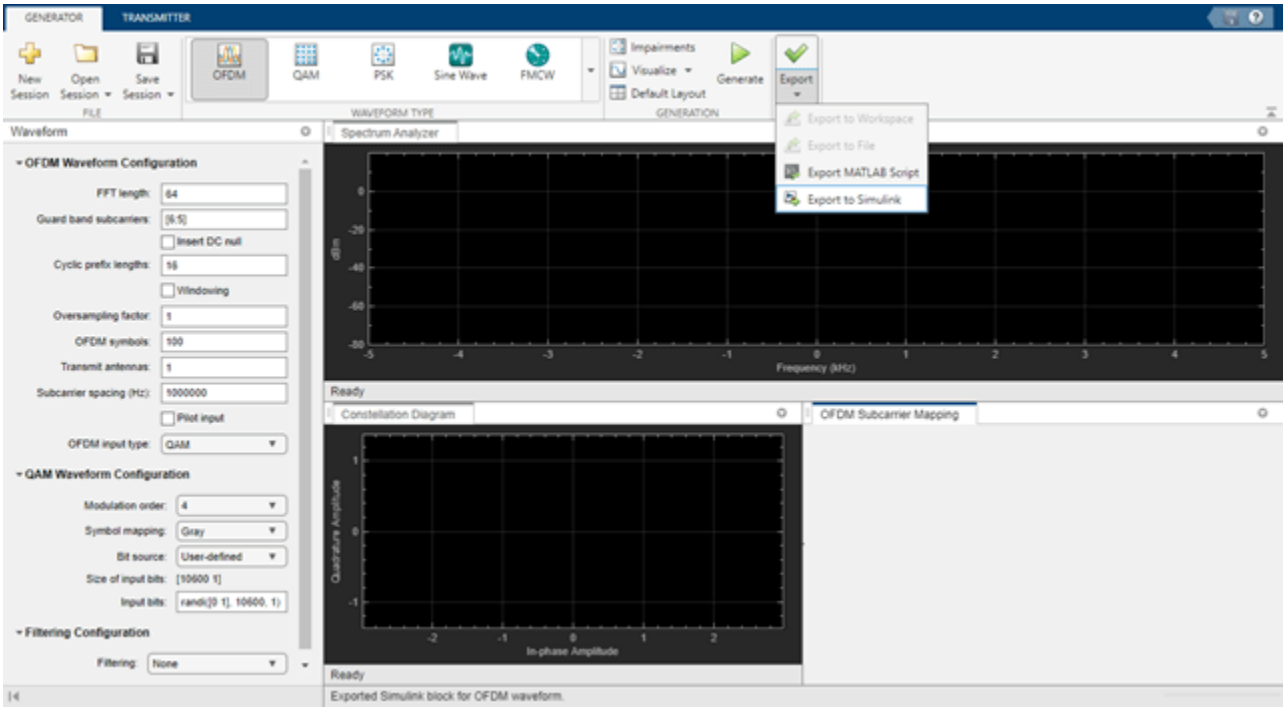
The **Wireless Waveform Generator** app is an interactive tool for creating, impairing, visualizing, and exporting waveforms. You can export the waveform to your workspace or to a `.mat` or `.bb` file. You can also export the waveform generation parameters to a runnable MATLAB® script or a Simulink® block. You can use the exported Simulink block to reproduce your waveform in Simulink. This example shows how to use the **Export to Simulink** capability of the app and how to configure the exported block to generate waveforms in Simulink.

Although this example focuses on exporting an OFDM waveform, the same process applies for all of the supported waveform types.

Export Wireless Waveform Configuration to Simulink

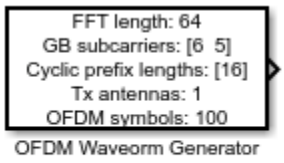
Open the **Wireless Waveform Generator** app by clicking the app icon on the **Apps** tab, under **Signal Processing and Communications**. Alternatively, enter `wirelessWaveformGenerator` at the MATLAB command prompt.

In the **Waveform Type** section, select an OFDM waveform by clicking **OFDM**. In the left-most pane of the app, adjust any configuration parameters for the selected waveform. Then export the configuration by clicking **Export** in the app toolstrip and selecting **Export to Simulink**.



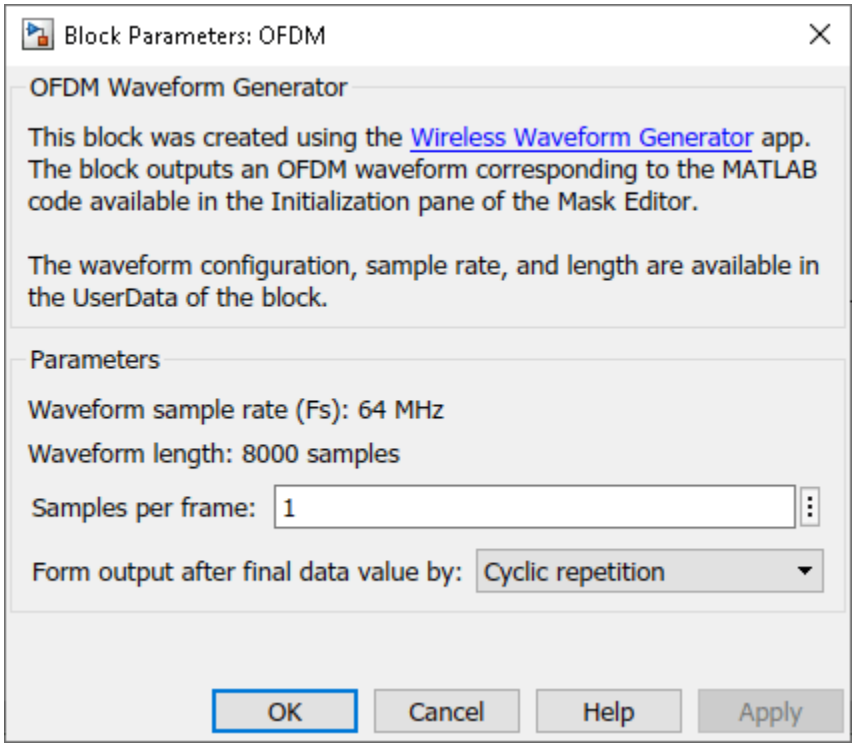
The **Export to Simulink** option creates a Simulink block, which outputs the selected waveform when you run the Simulink model. The block is exported to a new model if no open models exist.

```
modelName = 'WWGExport2SimulinkBlock';
open_system(modelName);
```




```
% Copyright 2021-2023 The MathWorks, Inc.
```

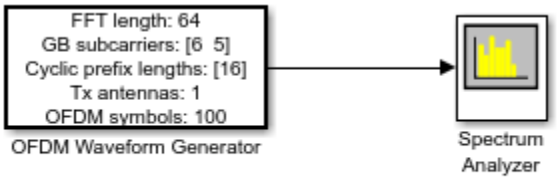
The **Form output after final data value by** block parameter specifies the output after all of the specified signal samples are generated. The value options for this parameter are `Cyclic repetition` and `Setting to zero`. The `Cyclic repetition` option repeats the signal from the beginning after it reaches the last sample in the signal. The `Setting to zero` option generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal. The **Waveform sample rate (Fs)** and **Waveform length** block parameters are derived from the waveform configuration that is available in the **Code** tab of the Mask Editor dialog box. For further information about the block parameters, see Waveform From Wireless Waveform Generator App. This figure shows the parameters of the exported block.




```
close_system(modelName);
```

Connect a Spectrum Analyzer block to the exported block.

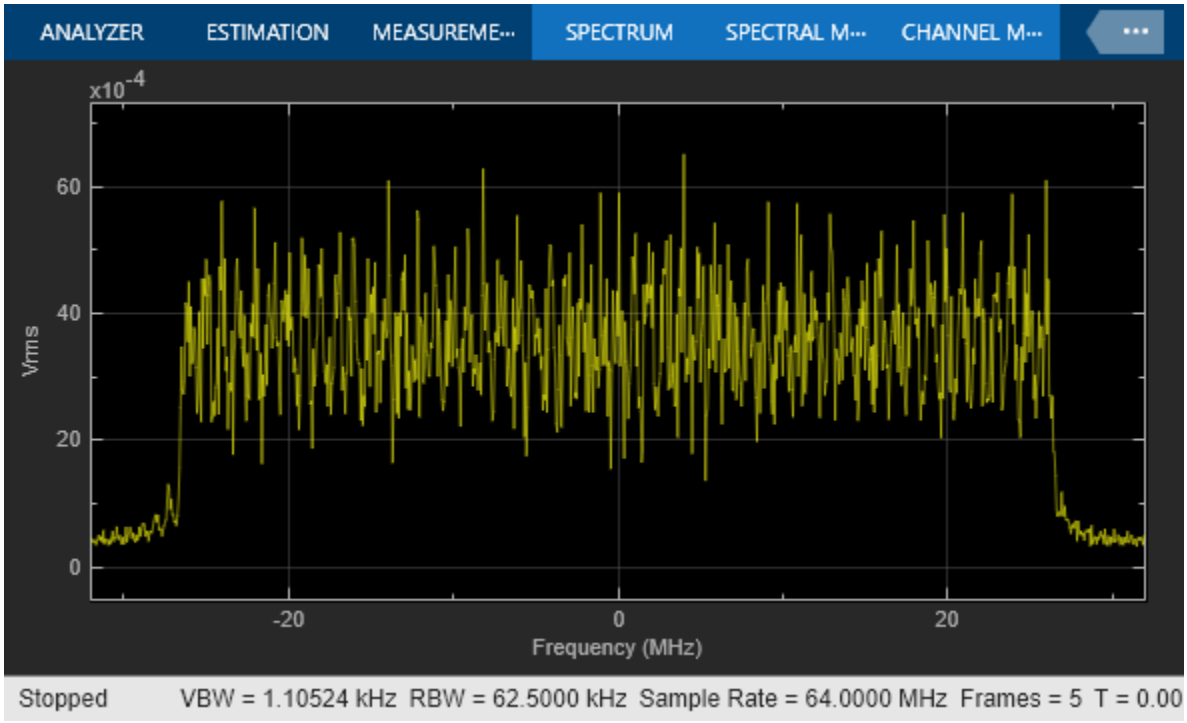
```
modelName = 'WWGExport2SimulinkModel';
open_system(modelName);
```




```
% Copyright 2021-2023 The MathWorks, Inc.
```

Simulate the model to visualize the waveform using the current configuration.

```
sim(modelName);
```



The Spectrum Analyzer block inherits the **Waveform sample rate (Fs)** parameter, which is 64 MHz.

```
close_system(modelName);
```

Modify Wireless Waveform Configuration

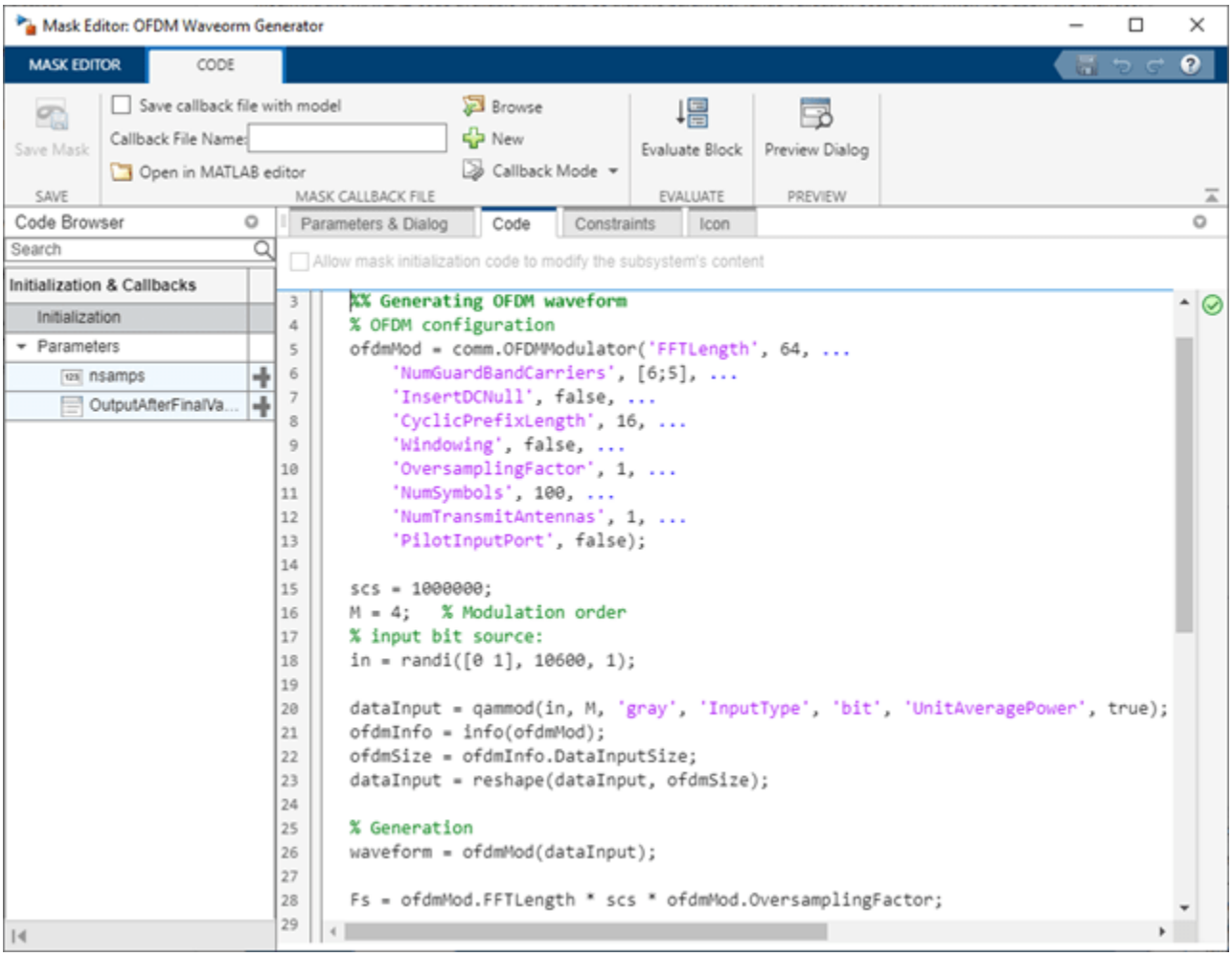
When you run the Simulink model, the exported block outputs the waveform generated in the **Code** tab of the Mask Editor dialog box for the block. The MATLAB code that initializes the waveform in this tab corresponds to the configuration that you selected in the **Wireless Waveform Generator** app before exporting the block. To modify the configuration of the waveform, choose one of these options:

- Open the **Wireless Waveform Generator** app, select the configuration of your choice, and export a new block. This option provides interaction with an app interface instead of MATLAB code,

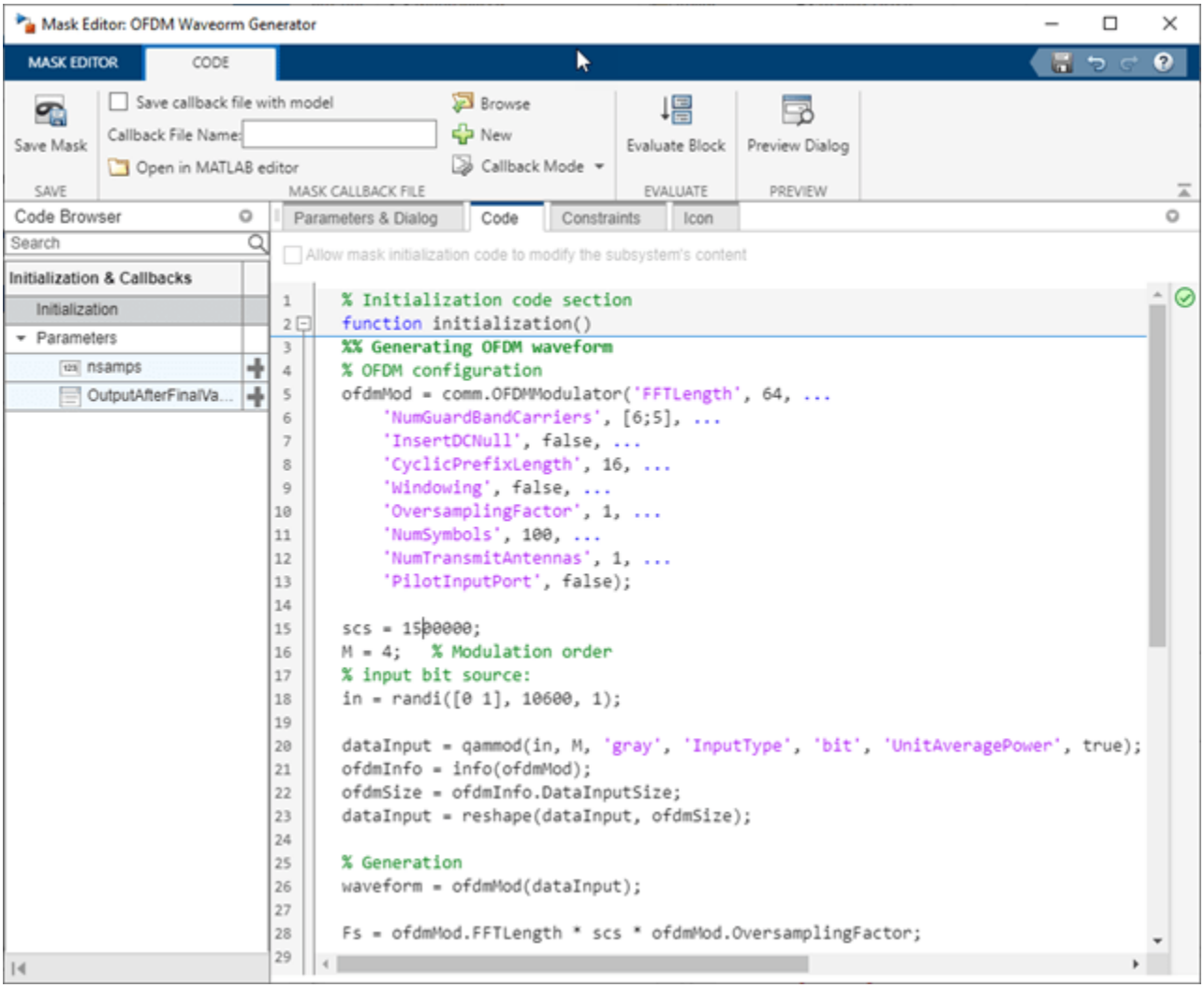
parameter range validation during the parameterization process, and visualization of the waveform before running the Simulink model.

- Update the configuration parameters that are available in the **Code** tab of the Mask Editor dialog box of the exported block. This option requires modifying the MATLAB code available in this tab so that the parameter range validation occurs only when you apply the changes. This option does not provide visualization of the waveform before running the Simulink model. Modifying the waveform parameters using this option is not recommended if you are not familiar with the MATLAB code that generates the selected waveform.

You can update the configuration in the **Code** tab of the Mask Editor. To open the Mask Editor, click the exported block and press **Ctrl+M**.

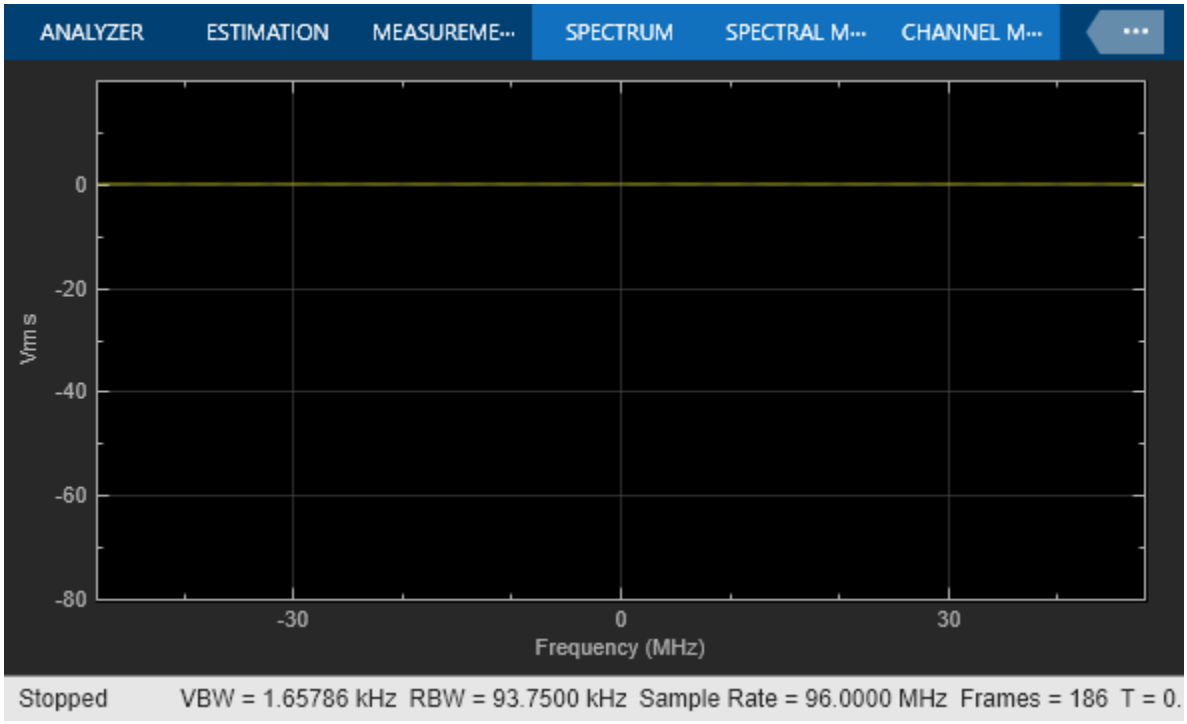


Use the MATLAB code that is available in the **Code** tab to update the parameters of your choice. For example, set the subcarrier spacing, `scs`, to 1,500,000 Hz.



Click **OK** to apply the changes and close the Mask Editor dialog box. Simulate the model to visualize the updated waveform.

```
modelName = 'WWGExport2SimulinkModelSCSModified';
sim(modelName);
```



The Spectrum Analyzer block now shows a sample rate of 96 MHz, which is 1.5 times the previous sample rate, as expected.

Share Wireless Waveform Configuration with Other Blocks in the Model

To access read-only block parameters and waveform configuration parameters, use the `UserData` common block property, which is a structure with these fields.

- `WaveformConfig`: Waveform configuration
- `WaveformLength`: Waveform length
- `Fs`: Waveform sample rate

You can access the user data of the exported block by using the `get_param` function.

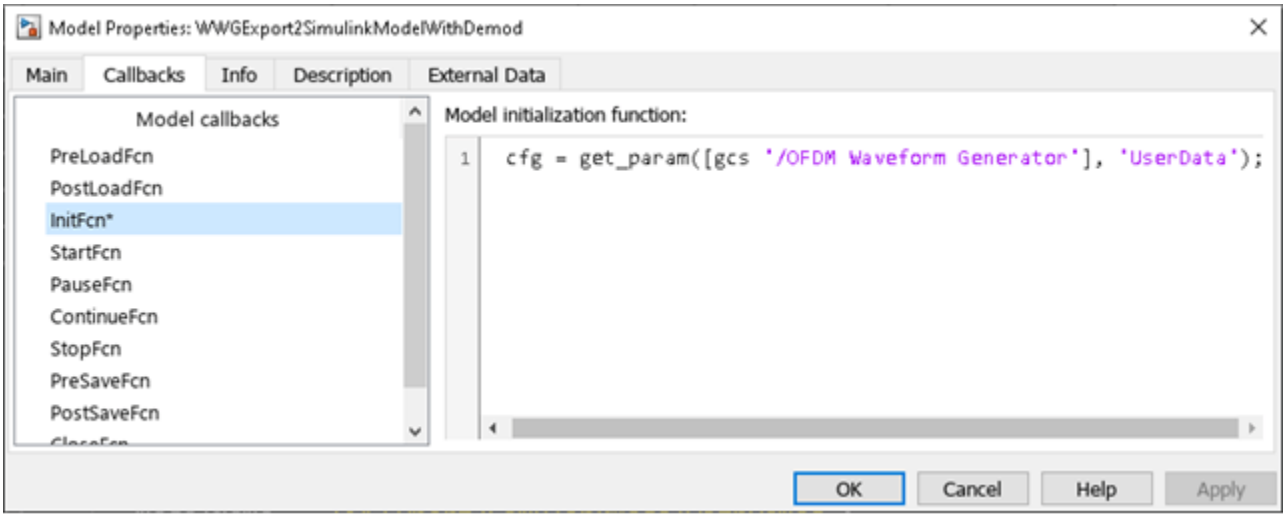
```
get_param([gcs '/OFDM Waveform Generator'],'UserData')
```

```
ans =

  struct with fields:

    WaveformConfig: [1x1 comm.OFDMModulator]
    WaveformLength: 8000
                Fs: 96000000
```

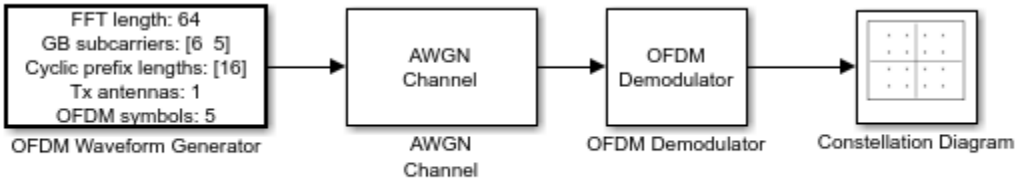
Store the structure available in the user data in a base workspace variable by using the `InitFcn` in the callback. The `InitFcn` callback is executed during a model update and simulation. To use this callback, click the **MODELING** tab, then click the **Model Settings** dropdown, and click the **Model Properties** option. In the **Callbacks** pane, select the `InitFcn` callback. Assign the user data to a new base workspace variable (for example, `cfg`).



The parameters that are available in the user data of the exported block are updated every time you apply configuration changes in the **Code** tab.

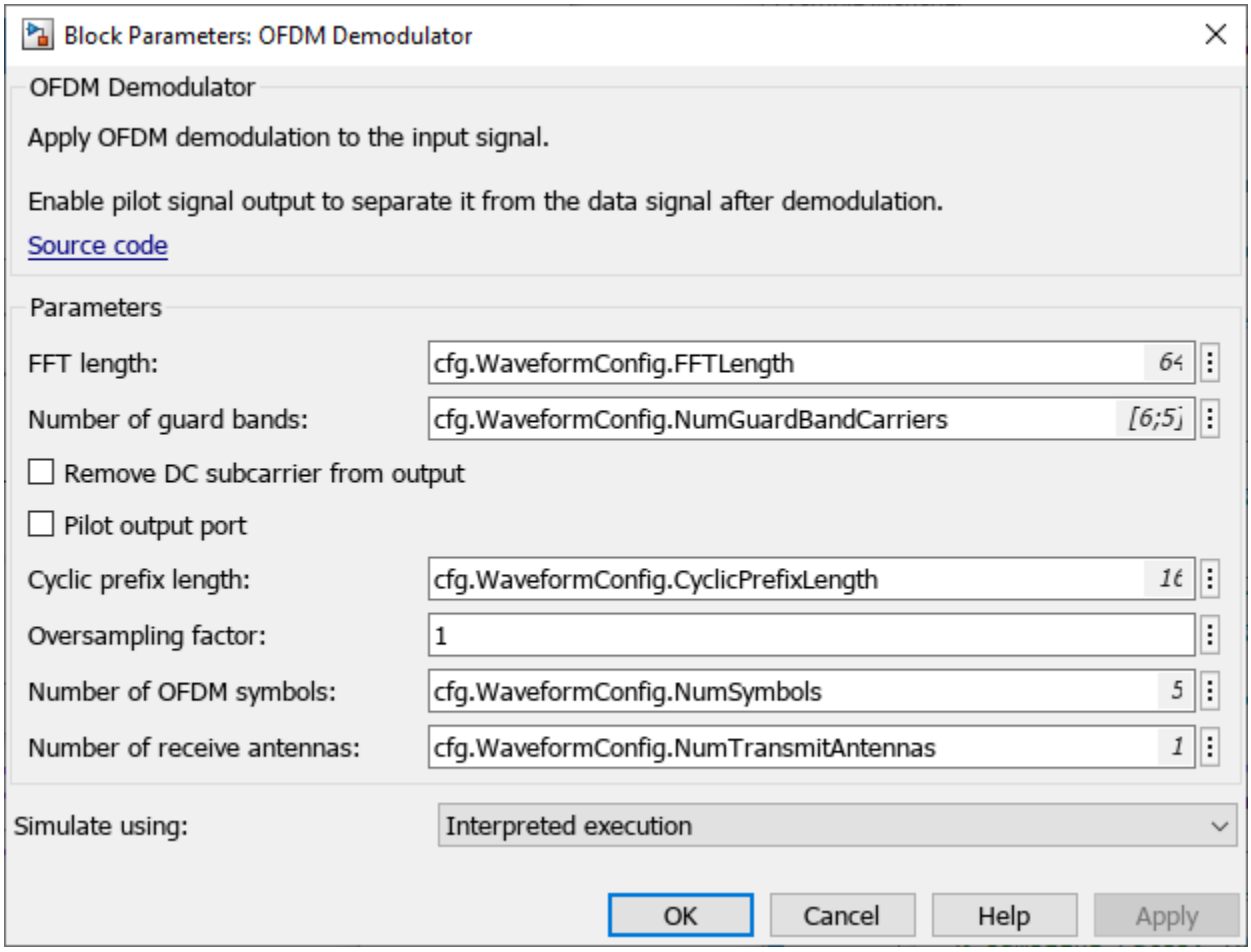
To demodulate the OFDM waveform, add an OFDM Demodulator block to the model. Connect an AWGN Channel block between the OFDM Waveform Generator and OFDM Demodulator blocks to add white Gaussian noise to the input signal. Also add a Constellation Diagram block to plot the demodulated symbols.

```
modelName = 'WWGExport2SimulinkModelWithDemod';
open_system(modelName);
```



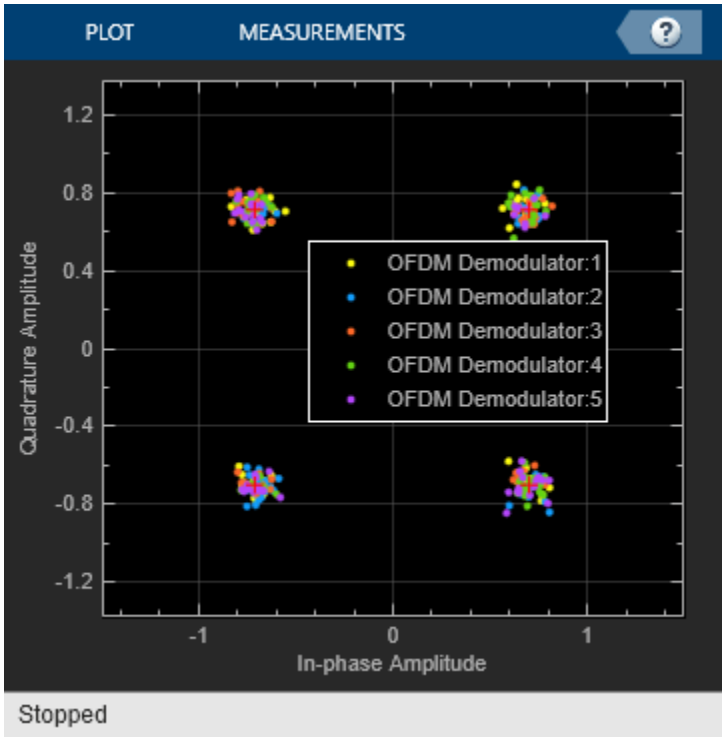

```
% Copyright 2021-2023 The MathWorks, Inc.
```

The parameters that are required to configure the OFDM Demodulator block must match the parameters that are used to configure the exported block, (otherwise, demodulation fails). To access the configuration parameters of the exported block, use the variable `cfg`. This figure shows the parameters of the OFDM Demodulator block.



Because the OFDM Demodulator block requires the entire OFDM waveform for demodulation, set the **Samples per frame** parameter in the exported block to `cfg.WaveformLength`. Simulate the model.

```
sim(modelName);
```

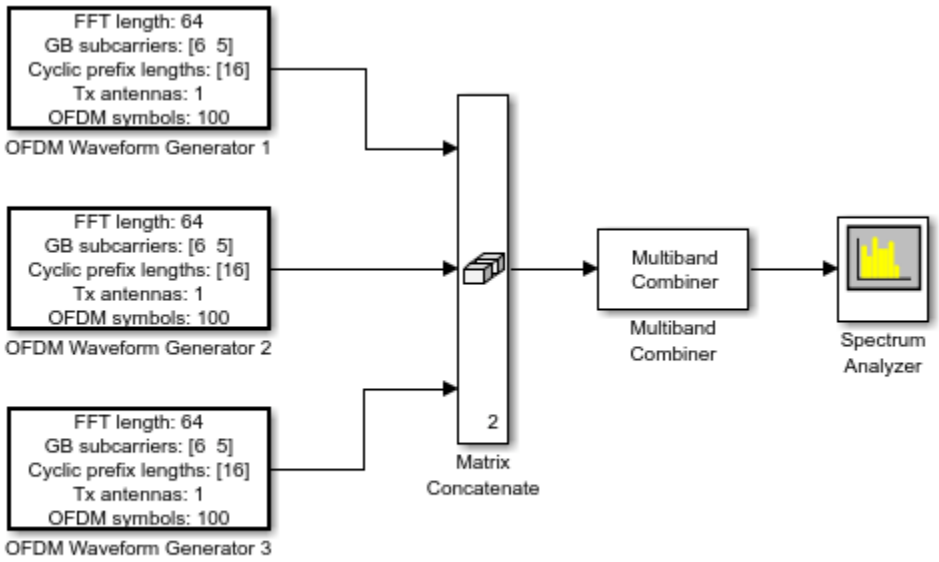


After demodulating the OFDM waveform by using the OFDM Demodulator block, the Constellation Diagram block displays the resulting QAM symbols.

Generate Multicarrier Waveforms

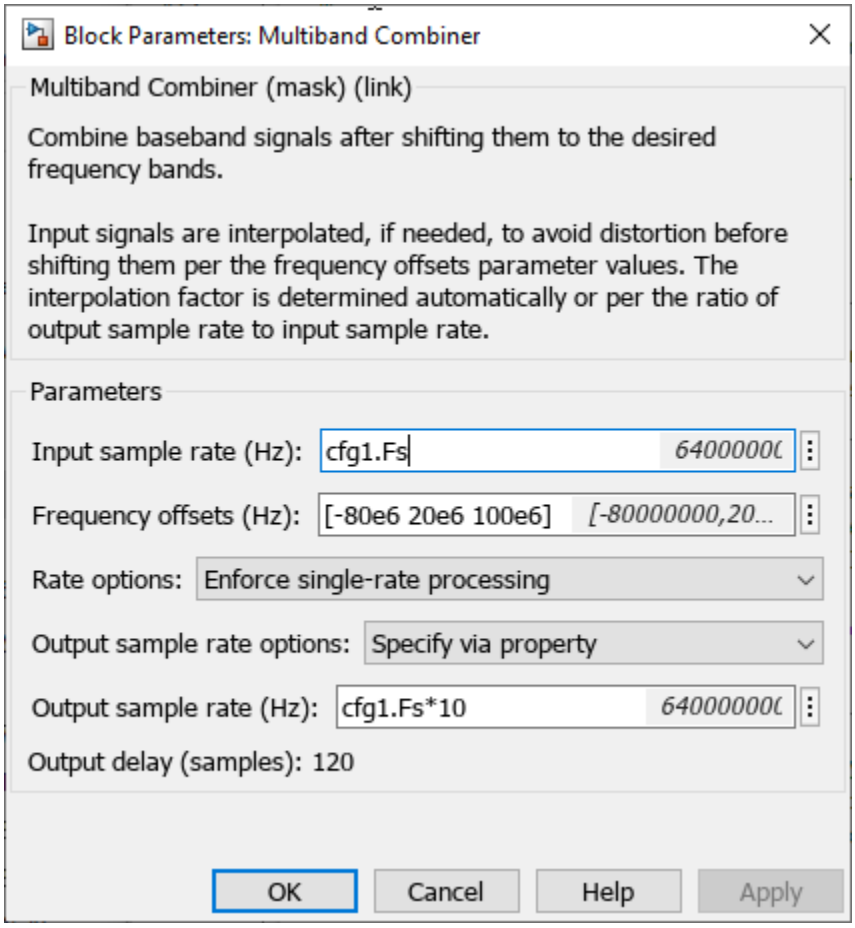
For multicarrier generation, the sampling rates for all of the waveforms must be the same. To shift the waveforms to a carrier offset and aggregate them, you can use the Multiband Combiner block.

```
modelName = 'WWGExport2SimulinkMulticarrier';
open_system(modelName);
```



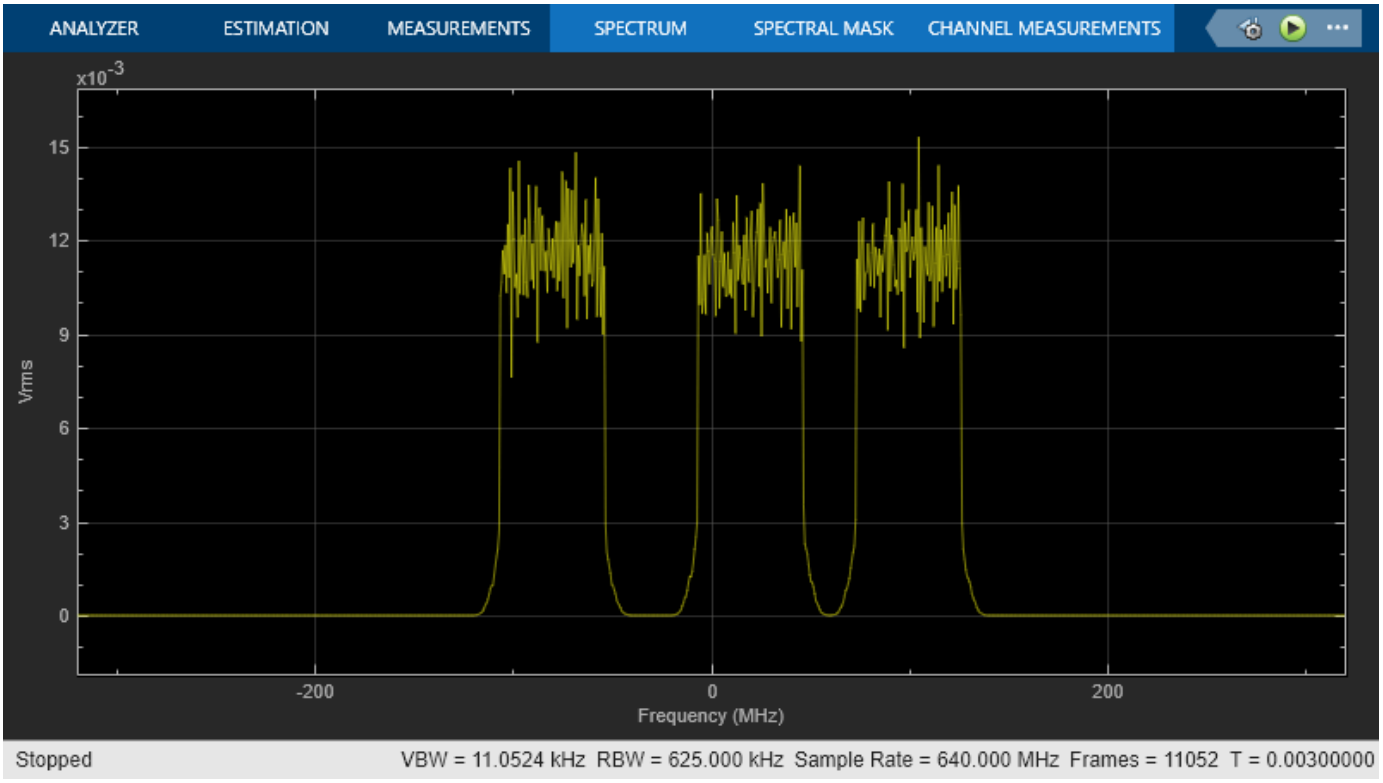

```
% Copyright 2021-2023 The MathWorks, Inc.
```

To shift the waveforms in frequency, you might have to increase the sampling rates. The Multiband Combiner block provides the option to oversample the input waveforms before shifting and combining them. This figure shows the parameters of the Multiband Combiner block.



Simulate the model to visualize the waveforms that are centered at -80, 20, and 100 MHz.

```
sim(modelName);
```


ANALYZER
ESTIMATION
MEASUREMENTS
SPECTRUM
SPECTRAL MASK
CHANNEL MEASUREMENTS
x10^-3
15
12
9
6
3
0
Vrms
-200
0
200
Frequency (MHz)
Stopped
VBW = 11.0524 kHz RBW = 625.000 kHz Sample Rate = 640.000 MHz Frames = 11052 T = 0.00300000

App-Based Bluetooth LE Waveform Generation

Generate Bluetooth® low energy (LE) waveforms by using the **Bluetooth LE Waveform Generator** app.

Open Bluetooth LE Waveform Generator App

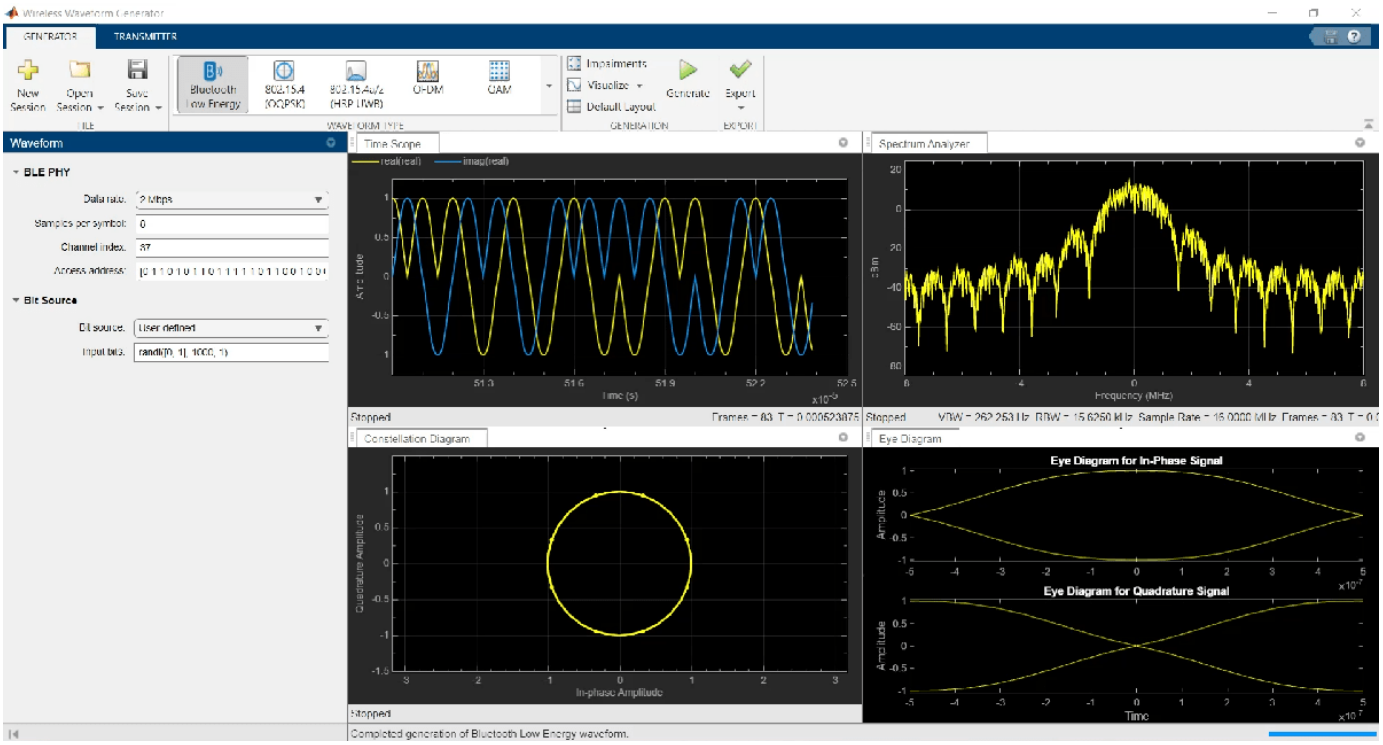
On the **Apps** tab of the MATLAB® toolstrip under **Signal Processing and Communication**, select the **Bluetoth Low Energy** app icon. This opens the **Wireless Waveform Generator** app configured for Bluetooth LE waveform generation.

Generate Bluetooth LE Waveform

To generate a Bluetooth LE waveform, perform these steps.

1. Select the PHY **Data rate** of the Bluetooth LE waveform you want to generate. The app enables you to generate Bluetooth LE waveforms with PHY data rates of 2 Mbps, 1 Mbps, 500 Kbps, or 125 Kbps.
2. Specify the values for **Samples per symbol**, **Channel index**, and **Access address**.
3. Specify the **Bit source** value.
4. To visualize the waveform, click **Generate**.

For example, this figure shows the **Time Scope, Spectrum Analyzer, Constellation Diagram,** and **Eye Diagram** visualization results for a Bluetooth LE waveform generated by using random input bits at a data rate of 2 Mbps.



Export Generated Waveform

You can export the generated waveform and its parameters by clicking **Export**. You can export the waveform as one of these options.

- A MATLAB script with the `.m` extension, which you can run to generate the waveform without the app
- A file with a `.bb` or `.mat` extension
- Your MATLAB workspace, as a structure
- A Simulink block, which you can use to generate the waveform in a Simulink model without the app

Transmit Bluetooth LE Waveform

Transmitting waveforms requires a license for Instrument Control Toolbox™. To transmit a generated waveform, click the **Transmitter** tab on the app toolstrip and configure the instruments. You can use any instrument supported by the `rfsiggen` function.

Generate Bluetooth LE Waveform and Add RF Impairments

Using this example, you can:

1. Generate a Bluetooth® low energy (LE) waveform.
2. Configure the parameters of radio frequency (RF) impairments.
3. Impair the generated Bluetooth LE waveform and plot the spectrum of the transmitted and impaired Bluetooth LE waveform.

Specify the data length. Create a message column vector of the specified data length containing random binary values.

```
dataLength = 2056;                          % In bits
message = randi([0 1],dataLength,1);
symbolRate = 1e6;
```

Specify the values of the physical layer (PHY) mode, channel index, samples per symbol, and access address.

```
phyMode = 'LE125K';
chanIdx = 2;
sps = 4;
accAdd = [1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 ...
    0 1 0 1 1 0 0].';
```

Generate the Bluetooth LE waveform.

```
txWaveform = bleWaveformGenerator(message,'Mode',phyMode,'SamplesPerSymbol',sps,'ChannelIndex',cl
```

Initialize the RF impairments for the specified PHY mode and samples per symbol by using the `helperBLEImpairmentsInit` function. The helper function returns a structure with phase frequency offset and variable fractional delay fields.

```
initRFImp = helperBLEImpairmentsInit(phyMode,sps)
```

```
initRFImp = struct with fields:
         pfo: [1x1 comm.PhaseFrequencyOffset]
    varDelay: [1x1 dsp.VariableFractionalDelay]
      pnoise: [1x1 comm.PhaseNoise]
```

Specify the values of the frequency offset and phase offset.

```
initRFImp.pfo.FrequencyOffset = 150;        % In Hz
initRFImp.pfo.PhaseOffset = -1;             % In degrees
```

Specify the values of static timing offset, timing drift, variable timing offset, and DC offset.

```
staticTimingOff = 0.15*sps;
timingDrift = 10;
```

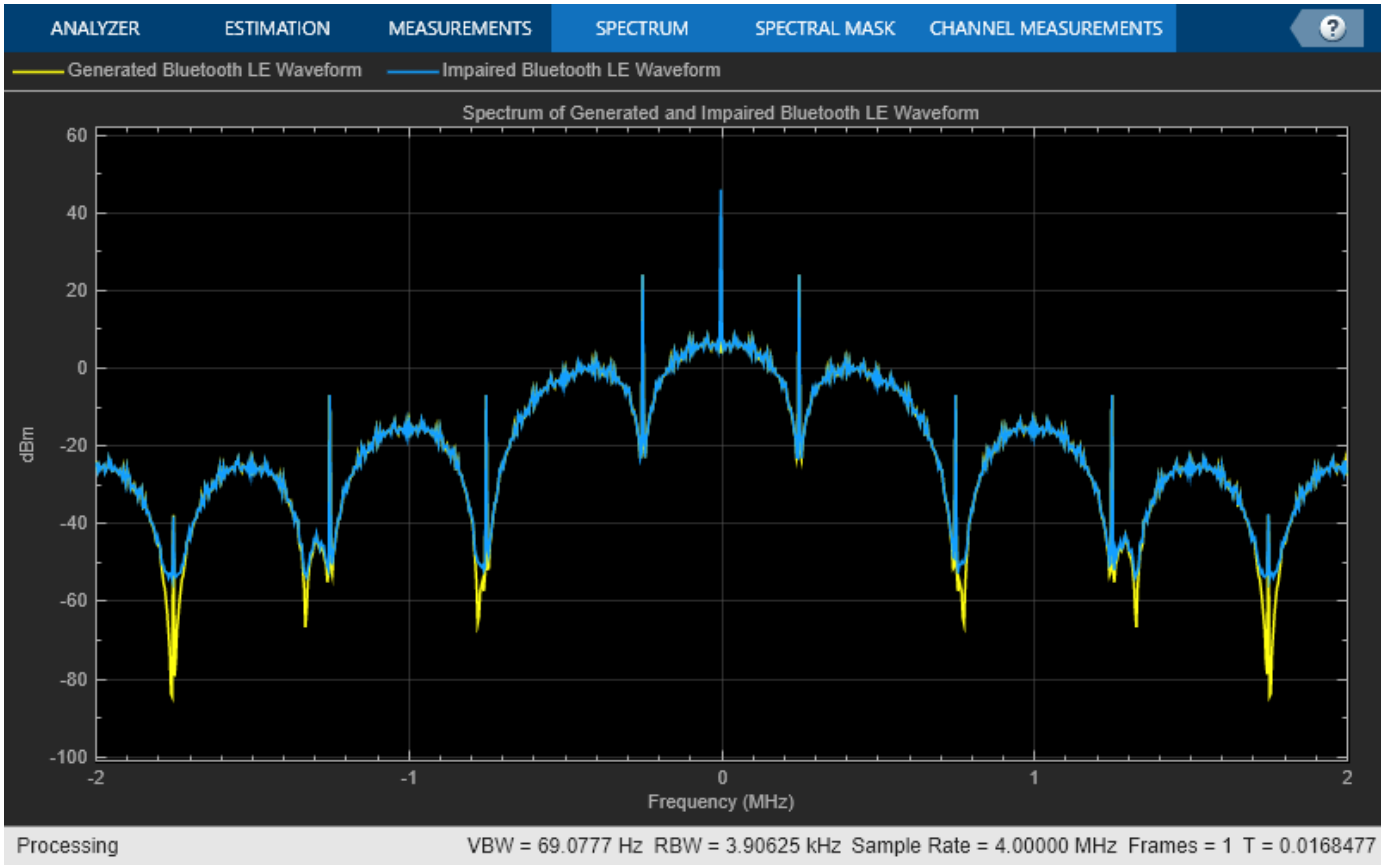
```
initRFImp.vdelay = (staticTimingOff:timingDrift:staticTimingOff + timingDrift * (length(txWavefo
initRFImp.dc =  6                                                              20;
```

Add RF impairments to the generated Bluetooth LE waveform by using the `helperBLEImpairmentsAddition` function.

```
txImpairedWaveform = helperBLEImpairmentsAddition(txWaveform,initRFImp);
```

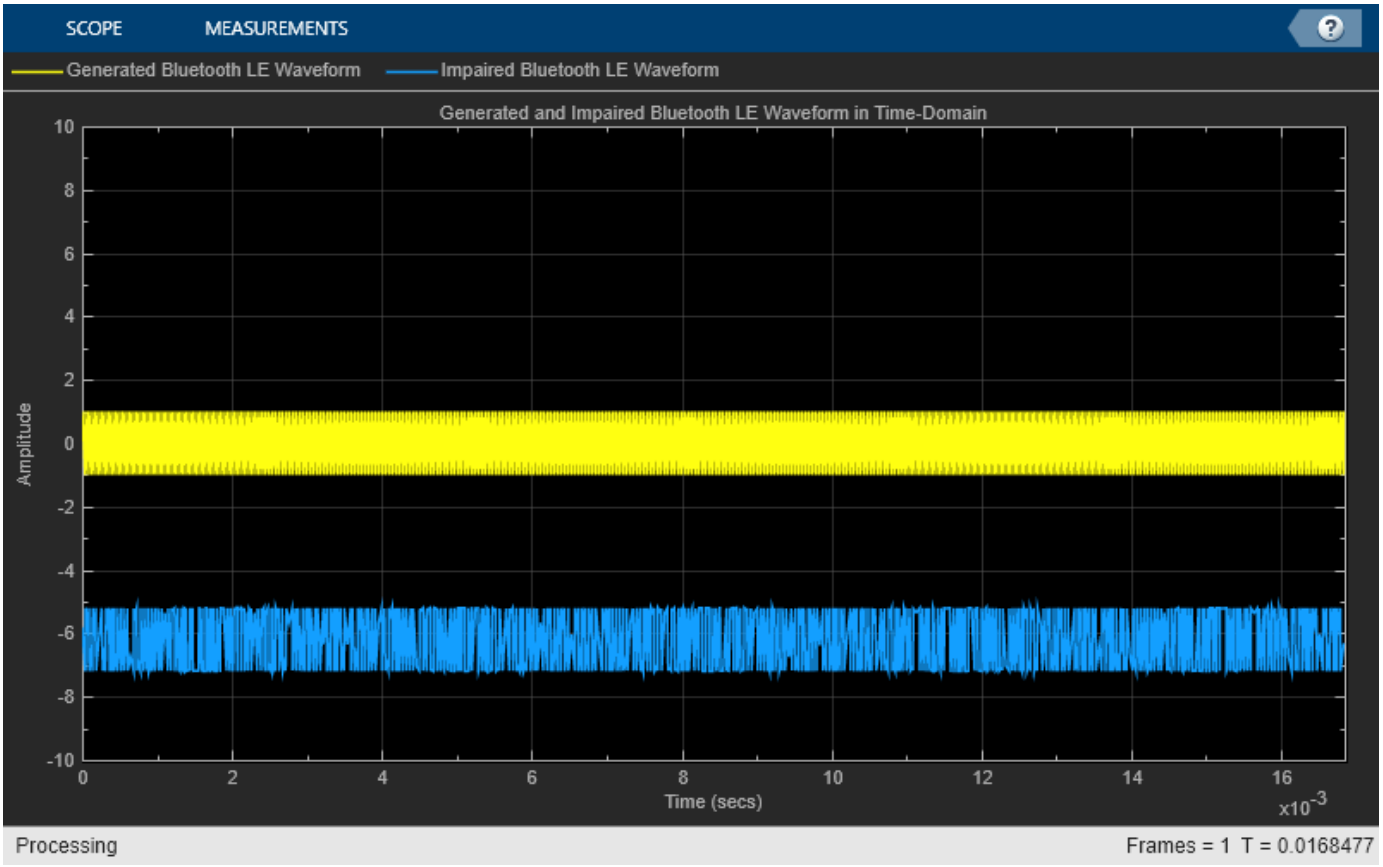
Create a default spectrum analyzer System object. Then, set the sample rate of the frequency spectrum. Visualize the generated and impaired Bluetooth LE waveform in the spectrum analyzer.

```
scope = spectrumAnalyzer;
scope.SampleRate = sps*symbolRate;
scope.NumInputPorts = 2;
scope.Title = 'Spectrum of Generated and Impaired Bluetooth LE Waveform';
scope.ShowLegend = true;
scope.ChannelNames = {'Generated Bluetooth LE Waveform','Impaired Bluetooth LE Waveform'};
scope(txWaveform,txImpairedWaveform);
```



Visualize the generated and impaired Bluetooth LE waveform in time-domain.

```
timeScope = timescope('SampleRate',symbolRate*sps,'TimeSpanSource','Auto','ShowLegend',true);
timeScope.Title = 'Generated and Impaired Bluetooth LE Waveform in Time-Domain';
timeScope.ChannelNames = {'Generated Bluetooth LE Waveform','Impaired Bluetooth LE Waveform'};
timeScope(real(txWaveform),real(txImpairedWaveform));
```



References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed December 27, 2021. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

Functions

`bleWaveformGenerator` | `bleIdealReceiver`

More About

- “End-to-End Bluetooth LE PHY Simulation Using Path Loss Model, RF Impairments, and AWGN”
- “End-to-End Bluetooth BR/EDR PHY Simulation with Path Loss, RF Impairments, and AWGN”
- “End-to-End Bluetooth LE PHY Simulation with AWGN, RF Impairments and Corrections”

Create, Configure, and Simulate Bluetooth LE Network

This example shows you how to simulate a Bluetooth® low energy (LE) network by using Bluetooth® Toolbox and Communications Toolbox™ Wireless Network Simulation Library.

Using this example, you can:

1. Create and configure a Bluetooth LE piconet with Central and Peripheral nodes.
2. Create and configure a link layer (LL) connection between Central and Peripheral nodes.
3. Add application traffic from the Central to Peripheral nodes.
4. Simulate Bluetooth LE network and retrieve the statistics of the Central and Peripheral nodes.

Check if the Communications Toolbox™ Wireless Network Simulation Library support package is installed. If the support package is not installed, MATLAB® returns an error with a link to download and install the support package.

```
wirelessnetworkSupportPackageCheck;
```

Create a wireless network simulator.

```
networkSimulator = wirelessNetworkSimulator.init();
```

Create a Bluetooth LE node, specifying the role as `"central"`. Specify the name and position of the node.

```
centralNode = bluetoothLENode("central");
centralNode.Name = "CentralNode";
centralNode.Position = [0 0 0];                 % In x-, y-, and z-coordinates in meters
```

Create a Bluetooth LE node, specifying the role as `"peripheral"`. Specify the name and position of the node.

```
peripheralNode = bluetoothLENode("peripheral");
peripheralNode.Name = "PeripheralNode";
peripheralNode.Position = [10 0 0]              % In x-, y-, and z-coordinates in meters
```

```
peripheralNode =
  bluetoothLENode with properties:

        TransmitterPower: 20
         TransmitterGain: 0
           ReceiverRange: 100
            ReceiverGain: 0
     ReceiverSensitivity: -100
             NoiseFigure: 0
    InterferenceFidelity: 0
                    Name: "PeripheralNode"
                Position: [10 0 0]

   Read-only properties:
                    Role: "peripheral"
        ConnectionConfig: [1x1 bluetoothLEConnectionConfig]
          TransmitBuffer: [1x1 struct]
                      ID: 2
```

Create a default Bluetooth LE configuration object to share the LL connection between the Central and Peripheral nodes.

```
cfgConnection = bluetoothLEConnectionConfig;
```

Specify the connection interval and connection offset. Throughout the simulation, the object establishes LL connection events for the duration of each connection interval. The connection offset is from the beginning of the connection interval.

```
cfgConnection.ConnectionInterval = 0.01; % In seconds
cfgConnection.ConnectionOffset = 0;      % In seconds
```

Specify the active communication period after the connection event is established between the Central and Peripheral nodes.

```
cfgConnection.ActivePeriod = 0.01 % In seconds
```

```
cfgConnection =
  bluetoothLEConnectionConfig with properties:

    ConnectionInterval: 0.0100
         AccessAddress: "5DA44270"
          UsedChannels: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 2
             Algorithm: 1
          HopIncrement: 5
     CRCInitialization: "012345"
    SupervisionTimeout: 1
               PHYMode: "LE1M"
         InstantOffset: 6
      ConnectionOffset: 0
          ActivePeriod: 0.0100
```

Configure the connection between Central and Peripheral nodes by using the `configureConnection` object function of the `bluetoothLEConnectionConfig` object.

```
configureConnection(cfgConnection,centralNode,peripheralNode);
```

Create a `networkTrafficOnOff` object to generate an On-Off application traffic pattern. Specify the data rate in kb/s and the packet size in bytes. Enable packet generation to generate an application packet with a payload.

```
traffic = networkTrafficOnOff(DataRate=100, ...
                              PacketSize=10, ...
                              GeneratePacket=true);
```

Add application traffic from the Central to the Peripheral node by using the `addTrafficSource` object function.

```
addTrafficSource(centralNode,traffic,"DestinationNode",peripheralNode.Name);
```

Create a Bluetooth LE network consisting of a Central and a Peripheral node.

```
nodes = {centralNode peripheralNode};
```

Add the Central and Peripheral nodes to the wireless network simulator.

```
addNodes(networkSimulator,nodes)
```

Set the simulation time in seconds and run the simulation.

```
simulationTime = 0.5;
run(networkSimulator,simulationTime);
```

```
Custom channel model is not added. Using free space path loss (fspl) model as the default channe
```

Retrieve application, link layer (LL), and physical layer (PHY) statistics corresponding to the broadcaster and receiver nodes. For more information about the statistics, see “Bluetooth LE Node Statistics”.

```
centralStats = statistics(centralNode)
```

```
centralStats = struct with fields:
    Name: "CentralNode"
      ID: 1
     App: [1x1 struct]
      LL: [1x1 struct]
     PHY: [1x1 struct]
```

```
peripheralStats = statistics(peripheralNode)
```

```
peripheralStats = struct with fields:
    Name: "PeripheralNode"
      ID: 2
     App: [1x1 struct]
      LL: [1x1 struct]
     PHY: [1x1 struct]
```

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed November 22, 2021. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

Functions
`addTrafficSource` | `statistics` | `configureConnection`

Objects
`bluetoothLENode` | `bluetoothLEConnectionConfig`

More About

- “Create, Configure, and Simulate Bluetooth LE Broadcast Audio Network”
- “Create and Visualize Bluetooth LE Broadcast Audio Residential Scenario”
- “Create, Configure and Simulate Bluetooth Mesh Network”
- “Establish Friendship Between Friend Node and LPN in Bluetooth Mesh Network”

- “Bluetooth LE Node Statistics”

Create, Configure and Simulate Bluetooth Mesh Network

This example shows you how to simulate a Bluetooth® mesh network by using Bluetooth® Toolbox and Communications Toolbox™ Wireless Network Simulation Library.

Using this example, you can:

1. Create and configure a Bluetooth mesh network.
2. Enable or disable relay feature of the mesh node.
3. Add application traffic between the source and destination nodes.
4. Simulate Bluetooth mesh network and retrieve the statistics of mesh nodes.

Check if the Communications Toolbox™ Wireless Network Simulation Library support package is installed. If the support package is not installed, MATLAB® returns an error with a link to download and install the support package.

```
wirelessnetworkSupportPackageCheck;
```

Create a wireless network simulator.

```
networkSimulator = wirelessNetworkSimulator.init;
```

Create a Bluetooth mesh profile configuration object, specifying the element address of the source node.

```
cfgMeshSource = bluetoothMeshProfileConfig(ElementAddress="0001")
```

```
cfgMeshSource =
  bluetoothMeshProfileConfig with properties:

             ElementAddress: "0001"
                      Relay: 0
                     Friend: 0
                   LowPower: 0
       NetworkTransmissions: 1
    NetworkTransmitInterval: 0.0100
                        TTL: 127
```

Create a Bluetooth LE node, specifying the role as `"broadcaster-observer"`. Specify the position of the source node. Assign the mesh profile configuration to the source node.

```
sourceNode = bluetoothLENode("broadcaster-observer");
sourceNode.Position = [0 0 0];
sourceNode.MeshConfig = cfgMeshSource;
```

Create a Bluetooth mesh profile configuration object, specifying the element address and enabling the relay feature of the Bluetooth LE node.

```
cfgMeshRelay = bluetoothMeshProfileConfig(ElementAddress="0002",Relay=true)
```

```
cfgMeshRelay =
  bluetoothMeshProfileConfig with properties:

             ElementAddress: "0002"
```

```
                    Relay: 1
                   Friend: 0
                 LowPower: 0
       NetworkTransmissions: 1
    NetworkTransmitInterval: 0.0100
                      TTL: 127
       RelayRetransmissions: 1
    RelayRetransmitInterval: 0.0100
```

Create a Bluetooth LE node, specifying the role as `"broadcaster-observer"`. Specify the position of the relay node. Assign the mesh profile configuration to the relay node.

```
relayNode = bluetoothLENode("broadcaster-observer");
relayNode.Position = [25 0 0];
relayNode.MeshConfig = cfgMeshRelay;
```

Create a Bluetooth mesh profile configuration object, specifying the element address of the Bluetooth LE node.

```
cfgMeshDestination = bluetoothMeshProfileConfig(ElementAddress="0003")
```

```
cfgMeshDestination =
  bluetoothMeshProfileConfig with properties:

             ElementAddress: "0003"
                      Relay: 0
                     Friend: 0
                   LowPower: 0
       NetworkTransmissions: 1
    NetworkTransmitInterval: 0.0100
                        TTL: 127
```

Create a Bluetooth LE node, specifying the role as `"broadcaster-observer"`. Specify the position of the destination node. Assign the mesh profile configuration to the destination node.

```
destinationNode = bluetoothLENode("broadcaster-observer");
destinationNode.Position = [50 0 0];
destinationNode.MeshConfig = cfgMeshDestination;
```

Create a `networkTrafficOnOff` object to generate an On-Off application traffic pattern. Specify the on time, data rate in kb/s, and packet size in bytes. Generate an application packet with a payload by enabling packet generation.

```
traffic = networkTrafficOnOff(OnTime=inf, ...
    DataRate=1, ...
    PacketSize=15, ...
    GeneratePacket=true);
```

Add application traffic between the source and destination nodes.

```
addTrafficSource(sourceNode,traffic, ...
    SourceAddress=cfgMeshSource.ElementAddress, ...
    DestinationAddress=cfgMeshDestination.ElementAddress,TTL=10);
```

Create a Bluetooth mesh network consisting of the source node, relay node, and destination node.

```
nodes = {sourceNode relayNode destinationNode};
```

Add the mesh nodes to the wireless network simulator.

```
addNodes(networkSimulator,nodes)
```

Set the simulation time and run the simulation.

```
simulationTime = 1;                    % In seconds
run(networkSimulator,simulationTime);
```

```
Custom channel model is not added. Using free space path loss (fspl) model as the default channe
```

Retrieve application, link layer (LL) , and physical layer (PHY) statistics related to the source, relay, and destination nodes by using the `statistics` object function. For more information about the statistics, see “Bluetooth LE Node Statistics”.

```
sourceStats = statistics(sourceNode)
```

```
sourceStats = struct with fields:
         Name: "Node1"
           ID: 1
          App: [1x1 struct]
    Transport: [1x1 struct]
      Network: [1x1 struct]
           LL: [1x1 struct]
          PHY: [1x1 struct]
```

```
relayStats = statistics(relayNode)
```

```
relayStats = struct with fields:
         Name: "Node2"
           ID: 2
          App: [1x1 struct]
    Transport: [1x1 struct]
      Network: [1x1 struct]
           LL: [1x1 struct]
          PHY: [1x1 struct]
```

```
destinationStats = statistics(destinationNode)
```

```
destinationStats = struct with fields:
         Name: "Node3"
           ID: 3
          App: [1x1 struct]
    Transport: [1x1 struct]
      Network: [1x1 struct]
           LL: [1x1 struct]
          PHY: [1x1 struct]
```

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed November 22, 2021. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

Functions
`addTrafficSource` | `statistics`

Objects
`bluetoothLENode` | `bluetoothMeshProfileConfig`

More About

- “Create, Configure, and Simulate Bluetooth LE Network”
- “Create, Configure, and Simulate Bluetooth LE Broadcast Audio Network”
- “Create and Visualize Bluetooth LE Broadcast Audio Residential Scenario”
- “Establish Friendship Between Friend Node and LPN in Bluetooth Mesh Network”
- “Bluetooth LE Node Statistics”
- “Energy Profiling of Bluetooth Mesh Nodes in Wireless Sensor Networks”
- “Bluetooth Mesh Flooding in Wireless Sensor Networks”

Create, Configure, and Simulate Bluetooth LE Broadcast Audio Network

This example shows how to simulate a Bluetooth® low energy (LE) isochronous broadcast audio network by using Bluetooth® Toolbox and Communications Toolbox™ Wireless Network Simulation Library.

Using this example, you can:

1. Create and configure a Bluetooth LE piconet with an isochronous broadcaster and receivers.
2. Add application traffic at the broadcaster.
3. Simulate the broadcast isochronous network and retrieve the statistics of the broadcaster and receivers.

Check if the Communications Toolbox™ Wireless Network Simulation Library support package is installed. If the support package is not installed, MATLAB® returns an error with a link to download and install the support package.

```
wirelessnetworkSupportPackageCheck;
```

Create a wireless network simulator.

```
networkSimulator = wirelessNetworkSimulator.init;
```

Create a Bluetooth LE node, specifying the role as `"isochronous-broadcaster"`. Specify the name and position of the node.

```
broadcasterNode = bluetoothLENode("isochronous-broadcaster");
broadcasterNode.Name = "Broadcaster";
broadcasterNode.Position = [0 0 0];                              % In x-, y-, and z-coordinates, i
```

Create two Bluetooth LE nodes, specifying the role as `"synchronized-receiver"`. Specify the name and position of the nodes.

```
receiverNode1 = bluetoothLENode("synchronized-receiver");
receiverNode1.Name = "Receiver1";
receiverNode1.Position = [10 0 0];
receiverNode2 = bluetoothLENode("synchronized-receiver");
receiverNode2.Name = "Receiver2";
receiverNode2.Position = [20 0 0]
```

```
receiverNode2 =
  bluetoothLENode with properties:

        TransmitterPower: 20
         TransmitterGain: 0
           ReceiverRange: 100
            ReceiverGain: 0
     ReceiverSensitivity: -100
             NoiseFigure: 0
    InterferenceFidelity: 0
                    Name: "Receiver2"
                Position: [20 0 0]
```

```
   Read-only properties:
                   Role: "synchronized-receiver"
              BIGConfig: [1x1 bluetoothLEBIGConfig]
         TransmitBuffer: [1x1 struct]
                     ID: 3
```

Create a default Bluetooth LE broadcast isochronous group (BIG) configuration object.

```
cfgBIG = bluetoothLEBIGConfig
```

```
cfgBIG =
  bluetoothLEBIGConfig with properties:

        SeedAccessAddress: "78E52493"
                  PHYMode: "LE1M"
                   NumBIS: 1
              ISOInterval: 0.0050
               BISSpacing: 0.0022
              SubInterval: 0.0022
                   MaxPDU: 251
              BurstNumber: 1
    PretransmissionOffset: 0
          RepetitionCount: 1
             NumSubevents: 1
           BISArrangement: "sequential"
                BIGOffset: 0
        ReceiveBISNumbers: 1
             UsedChannels: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 2
            InstantOffset: 6
    BaseCRCInitialization: "1234"
```

Configure the broadcaster and receiver nodes so they use the default BIG parameters.

```
configureBIG(cfgBIG,broadcasterNode,receiverNode1);
configureBIG(cfgBIG,broadcasterNode,receiverNode2);
```

Create a `networkTrafficOnOff` object to generate an On-Off application traffic pattern. Specify the data rate in kb/s and the packet size in bytes. Enable packet generation to generate an application packet with a payload.

```
traffic = networkTrafficOnOff(DataRate=500, ...
                              PacketSize=10, ...
                              GeneratePacket=true);
```

Add application traffic at the broadcaster node by using the `addTrafficSource` object function.

```
addTrafficSource(broadcasterNode,traffic);
```

Create a broadcast isochronous network consisting of LE broadcast audio nodes.

```
nodes = {broadcasterNode receiverNode1 receiverNode2};
```

Add the LE broadcast audio nodes to the wireless network simulator.

```
addNodes(networkSimulator,nodes)
```

Set the simulation time in seconds and run the simulation.

```
simulationTime = 0.3;
run(networkSimulator,simulationTime);

Custom channel model is not added. Using free space path loss (fspl) model as the default channe
```

Retrieve application, link layer (LL), and physical layer (PHY) statistics corresponding to the broadcaster and receiver nodes. For more information about the statistics, see “Bluetooth LE Node Statistics”.

```
broadcasterStats = statistics(broadcasterNode)

broadcasterStats = struct with fields:
    Name: "Broadcaster"
      ID: 1
     App: [1x1 struct]
      LL: [1x1 struct]
     PHY: [1x1 struct]


receiver1Stats = statistics(receiverNode1)

receiver1Stats = struct with fields:
    Name: "Receiver1"
      ID: 2
     App: [1x1 struct]
      LL: [1x1 struct]
     PHY: [1x1 struct]


receiver2Stats = statistics(receiverNode2)

receiver2Stats = struct with fields:
    Name: "Receiver2"
      ID: 3
     App: [1x1 struct]
      LL: [1x1 struct]
     PHY: [1x1 struct]
```

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed November 22, 2021. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

Functions
`addTrafficSource` | `statistics` | `configureBIG`

Objects
`bluetoothLENode` | `bluetoothLEBIGConfig`

More About

- “Bluetooth LE Audio”
- “Create, Configure, and Simulate Bluetooth LE Network”
- “Create and Visualize Bluetooth LE Broadcast Audio Residential Scenario”
- “Estimate Packet Delivery Ratio of LE Broadcast Audio in Residential Scenario”
- “Bluetooth LE Node Statistics”

Parameterize Bluetooth LE Direction Finding Features

The Bluetooth Core Specification 5.1 [2] provided by the Bluetooth Special Interest Group (SIG) added direction finding features in Bluetooth low energy (LE) technology. The Bluetooth direction-finding capabilities, angle of arrival (AoA) and angle of departure (AoD), are introduced in the Bluetooth Core Specification 5.1 [2]. For more information about Bluetooth LE direction finding, see the “Bluetooth Location and Direction Finding” topic and “Bluetooth LE Positioning by Using Direction Finding” and “Bluetooth LE Direction Finding for Tracking Node Position” examples.

The Bluetooth Toolbox enables you to configure and simulate these Bluetooth LE direction finding capabilities.

Set Simulation Parameters for Bluetooth LE Location and Direction Finding

Specify the number of Bluetooth LE locators and the dimensions in which the Bluetooth LE node position is to be determined. To estimate the 2-D or 3-D position of a Bluetooth LE node, specify at least two or three locators, respectively.

```
numDimensions = 2;
numLocators = 3;
```

Specify the bit energy-to-noise density ratio (Eb/No) range (in dB) and the number of iterations to simulate each Eb/No point.

```
EbNo = 6:2:16;
numIterations = 200;
```

Specify the direction finding method, the direction finding packet type, and the physical layer (PHY) transmission mode. The PHY transmission mode must be LE1M or LE2M for a connection-oriented constant tone extension (CTE) and LE1M for a connectionless CTE.

```
dfMethod = AoA;
dfPacketType = ConnectionCTE;
phyMode = LE1M;
```

Specify the antenna array parameters. The antenna array size must be a scalar or vector for 2-D or 3-D positioning, respectively. The scalar or vector array size represents a uniform linear array (ULA) or uniform rectangular array (URA), respectively. Specify the normalized element spacing between the antenna elements with respect to the wavelength. Specify the antenna switching pattern as a 1-by-M row vector, where M is in the range [2, $\frac{74}{\text{slotDuration}}$ + 1].

```
arraySize = 16;
elementSpacing = 0.5;
switchingPattern = 1:prod(arraySize);
```

Specify the Bluetooth LE waveform generation parameters. The length of the CTE must be in microseconds, in the range [16, 160], with a step size of 8 microseconds.

```
slotDuration = 2;          % Slot duration in microseconds
cteLength = 160;
sps = 8;
chanIndex = 17;
crcInit = '555551';
accAddress = '01234567';
payloadLength = 1;         % Payload length in bytes
```

Create Bluetooth LE Angle Estimation Configuration Object

Create a default Bluetooth LE angle estimation configuration object by using the `bleAngleEstimateConfig` object. This object enables you to configure different parameters for Bluetooth LE angle estimation.

```
cfgAngle = bleAngleEstimateConfig
```

```
cfgAngle =
  bleAngleEstimateConfig with properties:

           ArraySize: 4
      ElementSpacing: 0.5000
    EnableCustomArray: 0
         SlotDuration: 2
     SwitchingPattern: [1 2 3 4]
```

Specify a URA antenna design by setting the antenna array size of the configuration object to [4 4]. Set the row element spacing and column element spacing to 0.4 and 0.3, respectively. Specify the value of the antenna switching pattern.

```
cfgAngle.ArraySize = [4 4];
cfgAngle.ElementSpacing = [0.4 0.3];
cfgAngle.SwitchingPattern = 1:16
```

```
cfgAngle =
  bleAngleEstimateConfig with properties:

           ArraySize: [4 4]
      ElementSpacing: [0.4000 0.3000]
    EnableCustomArray: 0
         SlotDuration: 2
     SwitchingPattern: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]
```

Generate Random Positions for Bluetooth LE Locators

A Bluetooth LE locator represents a receiving device and a transmitting device in AoA and AoD calculation, respectively. To place a Bluetooth LE node at the origin and the locators randomly in the 2-D or 3-D space, use helperBLEGeneratePositions function. Specify the number of locators as 3 and the number of dimensions as 2. The function returns the 2-D position of the Bluetooth LE node at the origin, a matrix representing the position of the three locators, and the AoA or AoD (in degrees) between the Bluetooth LE node and the locators.

```
[nodePos,locatorPos,angle] = helperBLEGeneratePositions(3,2)
```

```
nodePos = 2×1

     0
     0


locatorPos = 2×3

  -23.7249  -57.5071  -12.5811
  -77.9415   69.9823   -1.7241


angle = 3×1

   73.0701
  -50.5887
    7.8032
```

Generate Bluetooth LE Direction Finding Packet

Set the simulation parameters to generate a Bluetooth LE direction finding packet.

```
dfPacketType = ConnectionCTE;
cteLength = 160;
dfMethod = AoA;
payloadLength = 1;              % Payload length in bytes
crcInit = '555551';
slotDuration = 2;               % Slot duration in microseconds
```

Derive the type of CTE based on the slot duration and the direction finding method.

```
if strcmp(dfMethod,'AoA')
    cteType = [0;0];
else
    cteType = [0;1];
    if slotDuration == 1
        cteType = [1;0];
```

```
    end
end
```

To generate a direction finding packet corresponding to the type of CTE, use helperBLEGenerateDFPDU function.

```
dfPacket = helperBLEGenerateDFPDU(dfPacketType,cteLength,cteType,payloadLength,crcInit);
```

Perform Antenna Steering and Switching on Bluetooth LE Waveform

Set Bluetooth LE direction finding simulation parameters to perform antenna steering and switching on a Bluetooth LE waveform.

```
dfPacketType = ConnectionCTE;
cteLength = 160;
dfMethod = AoA;
payloadLength = 1;          % Payload length in bytes
crcInit = '555551';
slotDuration = 2;           % Slot duration in microseconds
phyMode = LE1M;
sps = 8;
chanIndex = 17;
```

Derive the type of CTE based on the slot duration and the direction finding method.

```
if strcmp(dfMethod,'AoA')
    cteType = [0;0];
else
    cteType = [0;1];
    if slotDuration == 1
        cteType = [1;0];
    end
end
```

Create a default Bluetooth LE angle estimation configuration object. Specify the antenna slot duration.

```
obj = bleAngleEstimateConfig;
obj.SlotDuration = slotDuration;
```

Generate a direction finding packet corresponding to the type of CTE by using the helperBLEGenerateDFPDU function.

```
dfPacket = helperBLEGenerateDFPDU(dfPacketType,cteLength,cteType,payloadLength,crcInit);
```

Using the direction finding packet, generate the Bluetooth LE waveform.

```
bleWaveform = bleWaveformGenerator(dfPacket,'Mode',phyMode,'SamplesPerSymbol',sps, ...
    'ChannelIndex',chanIndex,'DFPacketType',dfPacketType);
```

Use the helperBLESteerSwitchAntenna function to steer the Bluetooth LE waveform by 45 degrees in azimuth and 0 degrees in elevation and switch between the antennas according to the switching pattern.

```
dfWaveform = helperBLESteerSwitchAntenna(bleWaveform,45, ...
    phyMode,sps,dfPacketType,payloadLength,obj);
```

Decode Bluetooth LE Waveform with Connection-Oriented CTE

Set the simulation parameters to decode the Bluetooth LE waveform.

```
dfPacketType = ConnectionCTE;
cteLength = 160;
dfMethod = AoA;
payloadLength = 1;          % Payload length in bytes
crcInit = '555551';
slotDuration = 2;           % Slot duration in microseconds
phyMode = LE1M;
sps = 8;
chanIndex = 17;
```

Derive the type of CTE based on the slot duration and the direction finding method.

```
if strcmp(dfMethod,'AoA')
    cteType = [0;0];
else
    cteType = [0;1];
    if slotDuration == 1
        cteType = [1;0];
    end
end
```

Create a default Bluetooth LE angle estimation configuration object. Specify the antenna slot duration.

```
obj = bleAngleEstimateConfig;
obj.SlotDuration = slotDuration;
```

To generate a direction finding packet corresponding to the type of CTE, use the helperBLEGenerateDFPDU function.

```
dfPacket = helperBLEGenerateDFPDU(dfPacketType,cteLength,cteType,payloadLength,crcInit);
```

Generate the Bluetooth LE waveform by using the direction finding packet.

```
bleWaveform = bleWaveformGenerator(dfPacket, ...
    'Mode',phyMode, ...
    'SamplesPerSymbol',sps, ...
    'ChannelIndex',chanIndex, ...
    'DFPacketType',dfPacketType);
```

Get the in-phase and quadrature (IQ) samples by decoding the Bluetooth LE waveform.

```
[bits,accAddr,iqSamples] = bleIdealReceiver(bleWaveform, ...
    'Mode',phyMode, ...
    'SamplesPerSymbol',sps, ...
    'ChannelIndex',chanIndex, ...
    'DFPacketType',dfPacketType, ...
    'SlotDuration',slotDuration);
```

Estimate AoA of Bluetooth LE Waveform

Create a Bluetooth LE angle estimation configuration object, specifying the values of the antenna array size, slot duration, and antenna switching pattern.

```
cfgAngle = bleAngleEstimateConfig('ArraySize',2,'SlotDuration',2, ...
    'SwitchingPattern',[1 2]);
```

Estimate the AoA of the Bluetooth LE waveform by using the `bleAngleEstimate` function. The function accepts IQ samples and the Bluetooth LE angle estimation configuration object as inputs. You can either use the IQ samples obtained by decoding the Bluetooth LE waveform or use the IQ samples corresponding to the connection data channel protocol data unit (PDU).

Specify the IQ samples of a connection data PDU with an AoA CTE of 2 *µs* slots, CTE time of 16 *µs*, and azimuth rotation of 70 degrees.

```
IQsamples = [0.8507 + 0.5257i; -0.5257 + 0.8507i; -0.8507 - 0.5257i; ...
    0.5257 - 0.8507i; 0.8507 + 0.5257i; -0.5257 + 0.8507i; ...
    -0.8507 - 0.5257i; 0.5257 - 0.8507i; -0.3561 + 0.9345i];
```

Estimate the AoA of the Bluetooth LE waveform.

```
angle = bleAngleEstimate(IQsamples,cfgAngle)
```

```
angle = 70
```

Estimate Bluetooth LE Transmitter Position in 2-D Network Using Angulation

Set the positions of the Bluetooth LE receivers (locators).

```
rxPosition = [-18 -40;-10 70];          % In meters
```

Specify the azimuth angle of the signal between each Bluetooth LE receiver and transmitter.

```
azimuthAngles = [29.0546 -60.2551];     % In degrees
```

Specify the localization method. Because the angle of the signal between each Bluetooth LE receiver and transmitter is known, set the localization method to `'angulation'`.

```
localizationMethod = "angulation";
```

Estimate the position of the Bluetooth LE transmitter. The actual position of the Bluetooth LE transmitter is at the origin: (0, 0).

```
txPosition = blePositionEstimate(rxPosition,localizationMethod, ...
    azimuthAngles)
```

```
txPosition = 2×1
10^-4 ×

    0.2374
    0.1150
```

References

[1] Bluetooth Technology Website. “Bluetooth Technology Website | The Official Website of Bluetooth Technology.” Accessed September 1, 2020. https://www.bluetooth.com/.

[2] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.1. https://www.bluetooth.com/.

[3] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3. https://www.bluetooth.com/.

See Also

Functions
`bleAngleEstimate` | `bleWaveformGenerator` | `bleIdealReceiver`

Objects
`bleAngleEstimateConfig`

More About

- “Bluetooth Location and Direction Finding”
- “Bluetooth LE Direction Finding for Tracking Node Position”
- “Bluetooth LE Positioning by Using Direction Finding”
- “Estimate Bluetooth LE Node Position”