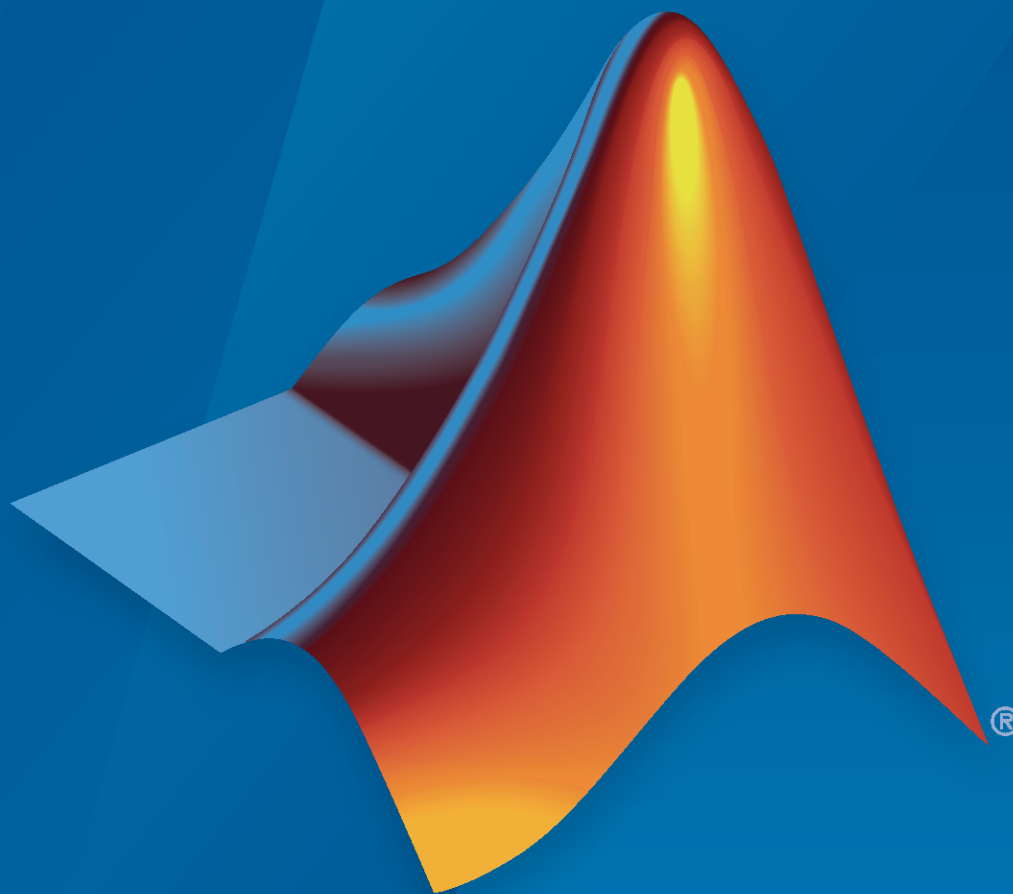# WLAN Toolbox™

Getting Started Guide

MATLAB®

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

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

October 2015	Online only	New for Version 1.0 (Release 2015b)
March 2016	Online only	Revised for Version 1.1 (Release 2016a)
September 2016	Online only	Revised for Version 1.2 (Release 2016b)
March 2017	Online only	Revised for Version 1.3 (Release 2017a)
September 2017	Online only	Revised for Version 1.4 (Release 2017b)
March 2018	Online only	Revised for Version 1.5 (Release 2018a)
September 2018	Online only	Revised for Version 2.0 (Release 2018b)
March 2019	Online only	Revised for Version 2.1 (Release 2019a)
September 2019	Online only	Revised for Version 2.2 (Release 2019b)
March 2020	Online only	Revised for Version 3.0 (Release 2020a)
September 2020	Online only	Revised for Version 3.1 (Release 2020b)
March 2021	Online only	Revised for Version 3.2 (Release 2021a)

Contents

Introduction

WLAN Toolbox Product Description

Simulate, analyze, and test WLAN communications systems

WLAN Toolbox provides standards-compliant functions for the design, simulation, analysis, and testing of wireless LAN communications systems. It includes configurable physical layer waveforms for IEEE® 802.11ax/ac/ad/ah and 802.11b/a/g/n/j/p standards. It also provides transmitter, channel modeling, and receiver operations, including channel coding (BCC and LDPC), modulation (OFDM, DSSS, and CCK), spatial stream mapping, channel models (TGay, TGax, TGac, TGah, and TGn), and MIMO receivers.

The toolbox provides reference designs to help you perform baseband link-level simulations and multi-node system-level simulations. You can generate and parse common MAC frames. You can also perform signal measurements such as channel power, spectrum mask, and occupied bandwidth, and create test benches for the end-to-end simulation of WLAN communications links.

You can study the effects of RF designs and interference on system performance. Using WLAN Toolbox with RF instruments or hardware support packages, you can connect your transmitter and receiver models to radio devices and verify your designs via over-the-air transmission and reception.

About WLAN

What Is WLAN?

In general, a wireless local area network (WLAN) refers to a wireless computer network. More commonly, WLAN is equated with the implementation specified by the IEEE 802.11™ group of standards and branded as Wi-Fi® by the Wi-Fi Alliance. The Wi-Fi Alliance certifies interoperability between IEEE 802.11 devices from different manufacturers. With WLAN Toolbox, you can model IEEE 802.11 standardized implementations of the WLAN physical (PHY) and medium access control (MAC) layers. You can also explore variations on implementations for future evolution of the standard.

Network Architecture

IEEE 802.11 defines the network architectures. In IEEE 802.11, a group of stations (STAs) within a defined coverage area and with appropriate association to each other form a basic service set (BSS). The BSS is a basic building block for 802.11 network architecture. A basic service area (BSA) defines an area containing STAs within a BSS. STAs can be associated in overlapping BSSs. In terms of mobility, STAs are either fixed, portable, or mobile. Any compliant STA can serve as an access point (AP).

This figure depicts WLAN components and network architectures built up from BSSs.
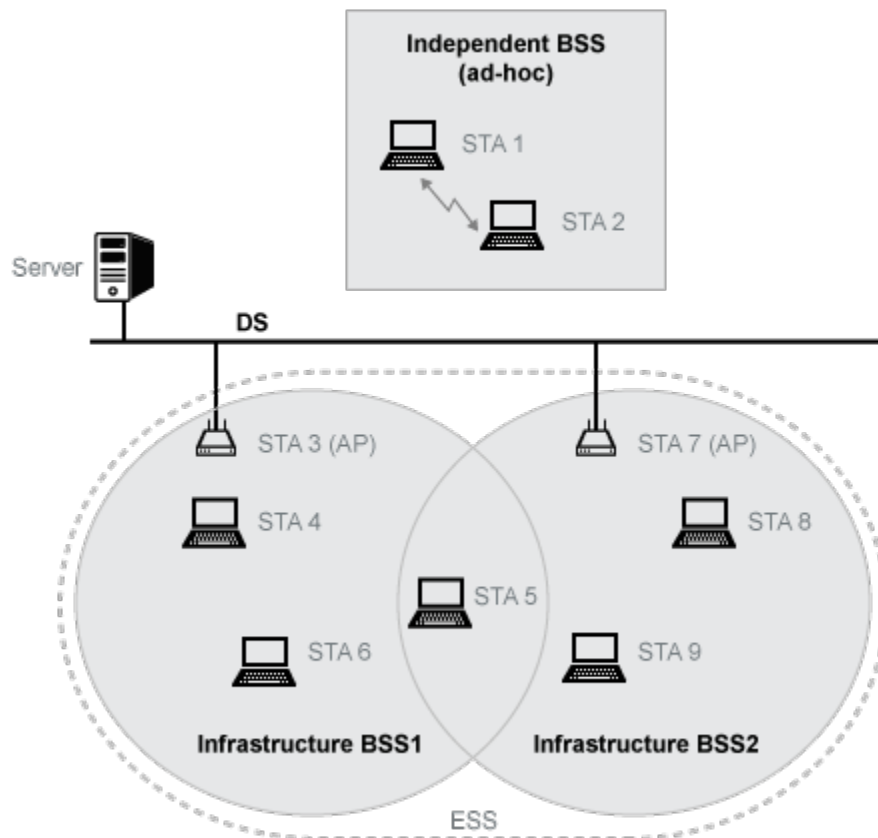
- Independent BSS (IBSS) describes STAs communicating directly with one another in an ad-hoc fashion. An IBSS has no connection to the wired network.

- Infrastructure BSS describes STAs associated with a central STA that manages the BSS. The central STA is referred to as an access point (AP). This deployment is commonly used in home, office, and hotspot network installations. Generally speaking, the AP connects wirelessly with associated STAs and is wired to the Internet. This connection enables associated STAs to communicate beyond the local BSS. The APs also wirelessly serve STAs in a BSA, providing internet connectivity for those STAs.

- Distributed systems (DS) interconnect infrastructure BSSs via their APs. Typically the DS backbone is an 802.3 Ethernet LAN.

- Extended service set (ESS) describes a set of infrastructure BSSs interconnected by a DS. In an ESS, APs communicate among themselves to forward traffic from one BSS to another and to facilitate the movement of mobile station from one BSS to another.

WLAN Protocol Stack

The interworking reference model shown here includes a subset of the network components associated with the data link layer (DLL) and physical layer (PHY). Section 4.9.2 of [2] describes the interworking reference model for 802.11. The medium access control (MAC) is a sublayer of the DLL.

The 802.11 standards focus on the MAC and PHY as a whole. WLAN Toolbox functionality focuses on the physical-medium-dependent (PMD) and physical layer convergence procedure (PLCP) sublayers of the PHY, the MAC sublayer, and their interfaces.

WLAN Message Exchange

Data and control information messages are exchanged between layers of the protocol stack within an individual STA and between peer layers in communicating STAs.

- Data and control information exchanged between peer STA layers are protocol information transfers. See (A-)MPDU and PPDU in the figure.
- Data and control information exchanged between layers within an STA are service information transfers. See MSDU and PSDU in the figure.
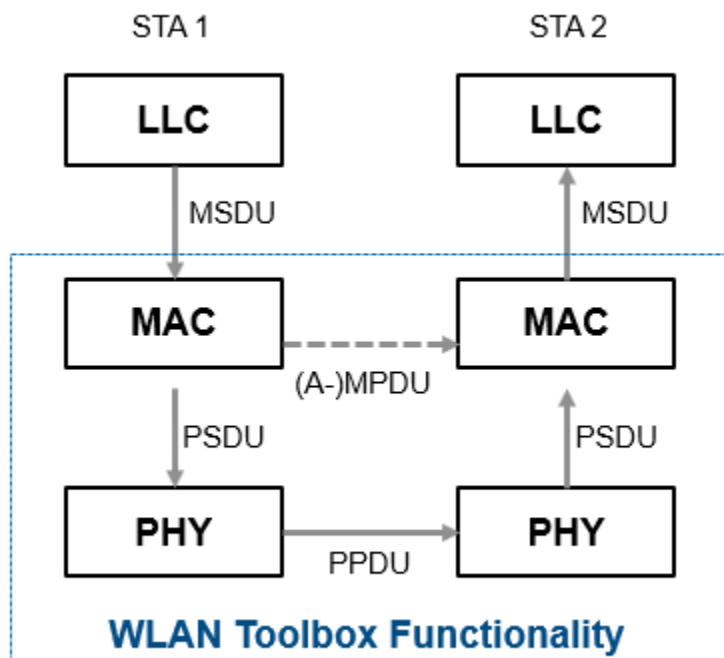
WLAN Toolbox functionality focuses on MAC and PHY implementations. Specifically, the toolbox models the exchange of PPDUs between PHY peers, and the exchange of MPDUs or A-MPDUs between MAC peers. Messages exchanged between protocol stack layers are briefly described here. For more information on these messages, see [2].

Message	Description
MSDU — MAC service data unit	Messages that transfer information between the logical link control (LLC) layer and the MAC layer within an STA
MPDU or A-MPDU — MAC protocol data unit or aggregated MAC protocol data unit	Messages that transfer information between MAC layer peers in communicating STAs
PSDU — PLCP service data unit	Messages that transfer information between the MAC and PHY layers within an STA
PPDU — PLCP protocol data unit	Messages that transfer information between PHY layer peers in communicating STAs

This figure shows the distinction between these WLAN message data units for a nonaggregated MAC frame.
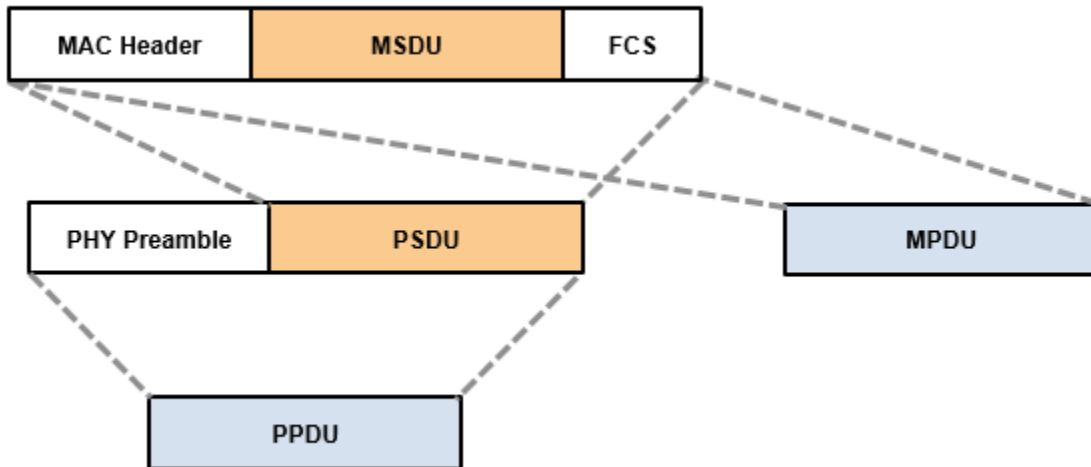
Note In reference to PSDU, the terms PLCP SDU and PHY SDU appear in the 802.11 standard. PLCP is the physical layer convergence procedure sublayer of the PHY. No distinction is made when the terms are used between layers.

Physical Layer Evolution

The IEEE 802.11 standardized implementation of WLAN has evolved since its first release in 1997. Today, it is deployed worldwide in unlicensed regions of the radio frequency spectrum. Since the first release, the 802.11 standard has progressed to include several physical layer implementations and has ensured backward compatibility with legacy releases. Over time, the maximum achievable transmission data rate has grown from 1 megabit per second (Mbps) to nearly 7 gigabit per second (Gbps).

WLAN Toolbox provides native support for the various 802.11 standard versions listed here. The toolbox focuses on the PHY and MAC layers, and enables adaptation of standards-based functionality to explore custom implementations.

Standard	Release Year	Modulation	Base Frequency (GHz)	Bandwidth (MHz)	Maximum Throughput (Mbps)	Antenna Scheme	PPDU Format
802.11	1997	DSSS	2.4	11	2	SISO	non-HT
802.11b™	1999	HR/DSSS/CCK	2.4	11	11	SISO	non-HT
802.11a™	1999	OFDM	5	5, 10, 20	54	SISO	non-HT
802.11g™	2003	802.11b and 802.11a @ 2.4 GHz					
802.11j™	2004	OFDM	4.9 and 5	10, 20	27	SISO	non-HT

Standard	Release Year	Modulation	Base Frequency (GHz)	Bandwidth (MHz)	Maximum Throughput (Mbps)	Antenna Scheme	PPDU Format
802.11n™ (Wi-Fi 4)	2009	OFDM	2.4 and 5	20, 40	< 600	MIMO, up to four streams	HT
802.11p™	2010	OFDM	5	5, 10	27	SISO	non-HT
802.11ad™	2012	SC/OFDM	60 GHz	1760 (SC), 2640 (OFDM)	< 7000	MIMO single stream with beamforming	DMG
802.11ac™ (Wi-Fi 5)	2013	OFDM	5	20, 40, 80, 160, 80+80	< 7000	DL MU-MIMO up to eight streams	VHT
802.11ah™	2016	OFDM	< 1	1, 2, 4, 8, 16	346	DL MU-MIMO up to four streams	S1G
802.11ax™ (Wi-Fi 6)	2020 (anticipated)	OFDMA	2.4 and 5	20, 40, 80, 160, 80+80	< 10,000	UL and DL MU-MIMO up to eight streams	HE

Deployment and commercial uptake grew with the increased data rates offered by 802.11b direct sequence spread spectrum (DSSS) with complementary code keying (CCK). At that time, companies began offering 802.11b products and systems for WLAN.

The 802.11a amendment increased data rates by introducing an orthogonal frequency division multiplexing (OFDM) physical layer. However, OFDM was deployed at only 5 GHz, so uptake was slow. A short time later, the Federal Communications Commission (FCC) allowed the use of OFDM at 2.4 GHz.

The adoption of the 802.11g amendment offered the opportunity to operate the PHY defined by 802.11a at 2.4 GHz, with backward compatibility to the 802.11b PHY.

With 802.11n, a data rate increase came by way of widened channel bandwidth and allowance of up to four input/output streams.

For 802.11ac, wider channels and up to eight input/output streams offers higher maximum throughputs. This increased throughput capability enables users to stream video to mobile devices in the home or at public mobile hot spots.

The 802.11ad amendment specifies operation in the 60-GHz band.

The 802.11ah amendment uses sub-1-GHz frequencies (unlicensed 900-MHz bands) to provide extended range, and has low energy consumption to support the concepts involving the Internet of Things (IoT).

The 802.11ax amendment introduces orthogonal frequency-division multiple access (OFDMA) to improve overall spectral efficiency, and higher-order 1024-point quadrature amplitude modulation

(1024-QAM) support for increased throughput. The demand for bandwidth continues to grow and the IEEE 802.11 working groups continue to advance standards to raise the throughput ceiling.

For the history of IEEE 802.11 and to monitor working group activities, consult the IEEE website.

References

[1] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[2] IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[3] IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016 as amended by IEEE Std 802.11ai™-2016). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[4] IEEE STD 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae™-2012, IEEE Std 802.11a™-2012, and IEEE Std 802.11ad-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4: Enhancements for Very High Throughput Operation in Bands below 6 GHz." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[5] IEEE STD 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae™-2012 and IEEE Std 802.11a™-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 4: Enhancements for Very High Throughput Operation in Bands below 6 GHz." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[6] Perahia, E., and R. Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. 2nd Edition. United Kingdom: Cambridge University Press, 2013.

See Also

Related Examples

External Websites

- https://standards.ieee.org/

WLAN PPDU Structure

Physical Layer Protocol Data Unit

IEEE 802.11[1] is a packet-based protocol. Each physical layer protocol data unit (PPDU) contains preamble and data fields. The preamble field contains the transmission vector format information. The data field contains the user payload and higher layer headers, such as medium access control (MAC) fields and cyclic redundancy check (CRC). The transmission vector format and the PPDU structure vary between 802.11 versions. The transmission vector (*TXVECTOR*) format parameter is classified as:

- *HE* to specify a high-efficiency (HE) physical layer (PHY) implementation.

  - HE refers to fields formatted for association with 802.11ax data. Reference [2] defines and describes the HE PHY layer and PPDU.

  - For HE, the *TXVECTOR* parameters, as defined in Table 27-1 of [2], determine the structure of PPDUs transmitted by an HE STA.

- *DMG* to specify a directional multi-gigabit (DMG) PHY implementation.

  - DMG refers to preamble fields formatted for association with 802.11ad data. Section 20 of [1] define and describe the DMG PHY layer and PPDU.

  - For DMG, the *TXVECTOR* parameters, as defined in Table 20-1 of [1], determines the structure of PPDUs transmitted by a DMG STA. For a DMG STA, the *MCS* parameter determines the overall structure of the DMG PPDU.

- *S1G* to specify a sub-1-GHz (S1G) PHY implementation.

  - S1G refers to preamble fields formatted for association with 802.11ah data. Reference [3] defines and describes the S1G PHY layer and PPDU.

  - For S1G, the *TXVECTOR* parameters, as defined in Table 23-1 of [3], determines the structure of PPDUs transmitted by an S1G STA. For an S1G STA, the *FORMAT* parameter determines the overall structure of the S1G PPDU.

- *VHT* to specify a very-high-throughput (VHT) PHY implementation.

  - VHT refers to preamble fields formatted for association with 802.11ac data. Section 21 of [1] defines and describes the VHT PHY layer and PPDU.

  - For VHT, the *TXVECTOR* parameters, as defined in Table 21-1 of [1], determine the structure of PPDUs transmitted by a VHT STA. For a VHT STA, the *FORMAT* parameter determines the overall structure of the PPDU and enables:

    - Non-HT format (*NON_HT*), based on Section 17 and including non-HT duplicate format.

    - HT-mixed format (*HT_MF*), as specified in Section 19.

    - HT-greenfield format (*HT_GF*), as specified in Section 19. WLAN Toolbox does not support HT_GF format.

    - VHT format (*VHT*), as specified in Section 21. The VHT format PPDUs contain a preamble compatible with Section 17 and Section 19 STAs. The non-VHT portions of the VHT preamble (the parts that precede the VHT-SIG-A field) are defined to enable decoding of the PPDU by VHT STAs.

---

1. IEEE Std 802.11-2016 Adapted and reprinted with permission from IEEE. Copyright IEEE 2016. All rights reserved.

- *HT* to specify a high-throughput (HT) PHY implementation.

  - HT refers to preamble fields formatted for association with 802.11n data. Section 19 of [1] defines and describes the HT PHY layer and PPDU. The standard defines two HT formats:

    - *HT_MF* indicates the HT-mixed format and contains a preamble compatible with HT and non-HT receivers. Support for HT-mixed format is mandatory.

    - • *HT_GF* indicates the HT-greenfield format and does not contain a non-HT compatible part. WLAN Toolbox does not support HT_GF format.

- *non-HT* to specify a PHY implementation that is not HT and is not VHT.

  - Non-HT refers to preamble fields formatted for association with pre-802.11n data. Section 17 of [1] defines and describes the OFDM PHY layer and PPDU for non-HT transmission. In addition to supporting non-HT synchronization, the non-HT preamble fields are used in support of HT and VHT synchronization.

The table shows 802.11 versions that the toolbox supports, along with the supported *TXVECTOR* options and associated modulation formats.

802.11 Version	Transmission Vector Format	Modulation Format	Bandwidths/MHz
802.11b	non-HT	DSSS/CCK	11
802.11a	non-HT	OFDM only	5, 10, 20
802.11j	non-HT	OFDM only	10
802.11p	non-HT	OFDM only	5, 10
802.11g	non-HT	OFDM	20
	non-HT	DSSS/CCK	11
802.11n (Wi-Fi 4)	HT_MF, Non-HT	OFDM only	20, 40
802.11ac (Wi-Fi 5)	VHT, HT_MF, Non-HT	OFDM only	20, 40, 80, 160
802.11ah	S1G	OFDM only	1, 2, 4, 8, 16
802.11ad	DMG	Single Carrier and OFDM	2640
802.11ax (Wi-Fi 6)	HE	OFDMA	20, 40, 80, 160

WLAN Toolbox configuration objects define the properties that enable creation of PPDUs and waveforms for the specified 802.11 transmission format. See `wlanHEMUConfig`, `wlanHESUConfig`, `wlanDMGConfig`, `wlanS1GConfig`, `wlanVHTConfig`, `wlanHTConfig`, and `wlanNonHTConfig`.

HE PPDU Field Structure

In HE, there are four transmission modes supported. The field structure for HE PPDUs consists of preamble and data portions. The legacy preamble fields (L-STF, L-LTF, and L-SIG) are common to all four HE transmission modes and with the VHT, HT, and non-HT format preambles.

HE preamble fields include additional format-specific signaling fields. Each format defines a data field for transmission of user payload data.
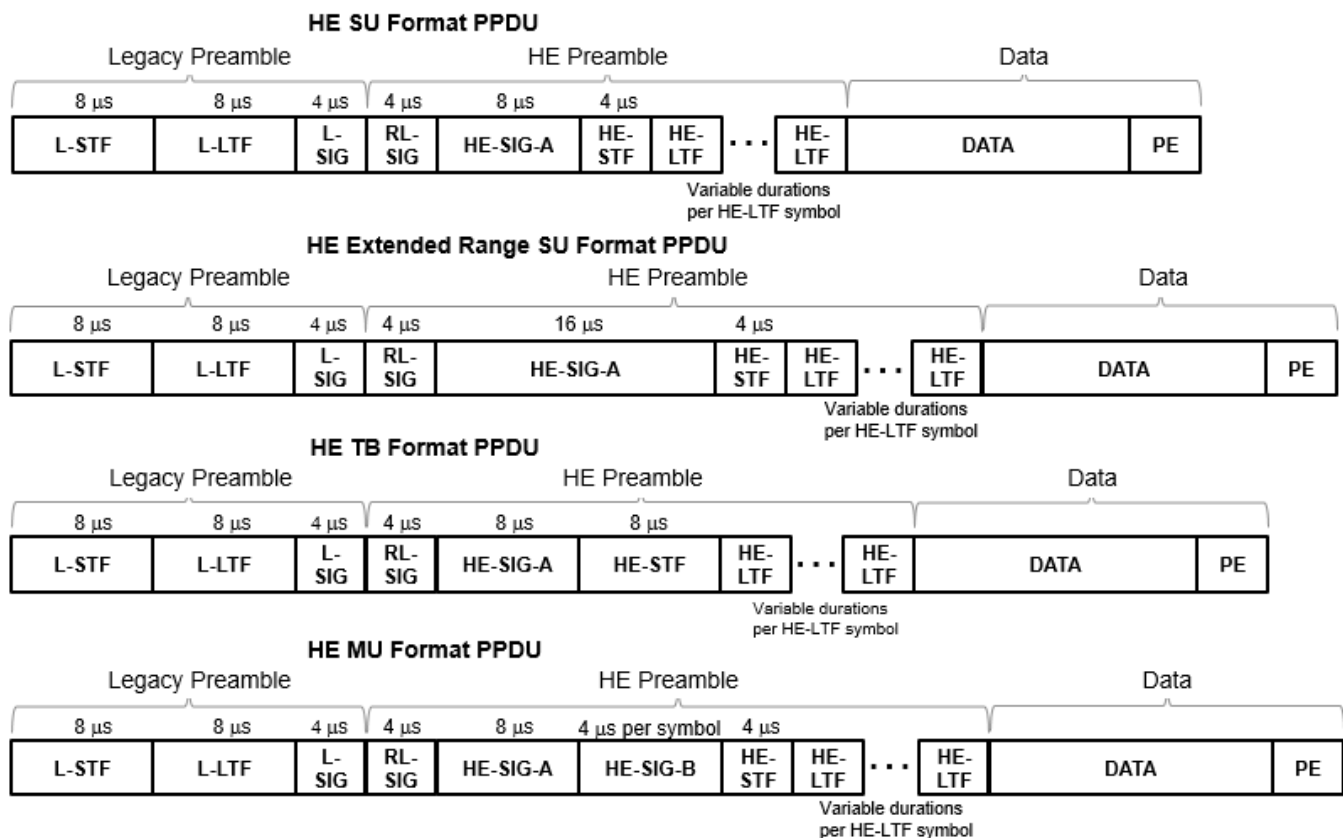
PPDU Field Abbreviation	Description
L-STF	Non-HT Short Training field
L-LTF	Non-HT Long Training field
L-SIG	Non-HT Signal field
RL-SIG	Repeated Non-HT Signal field
HE-SIG-A	HE Signal A field
HE-SIG-B	HE Signal B field
HE-STF	HE Short Training field
HE-LTF	HE Long Training field
HE-Data	Data field carrying the PSDUs
PE	Packet Extension field

The RL-SIG, HE-SIG-A, HE-STF, HE-LTF, and PE fields are present in all HE PPDU formats. The HE-SIG-B field is present only in the HE MU PPDU. For more information, see Section 27.3.4 of [2].

DMG Format PPDU Field Structure

In DMG, there are three physical layer (PHY) modulation schemes supported: control, single carrier, and OFDM.
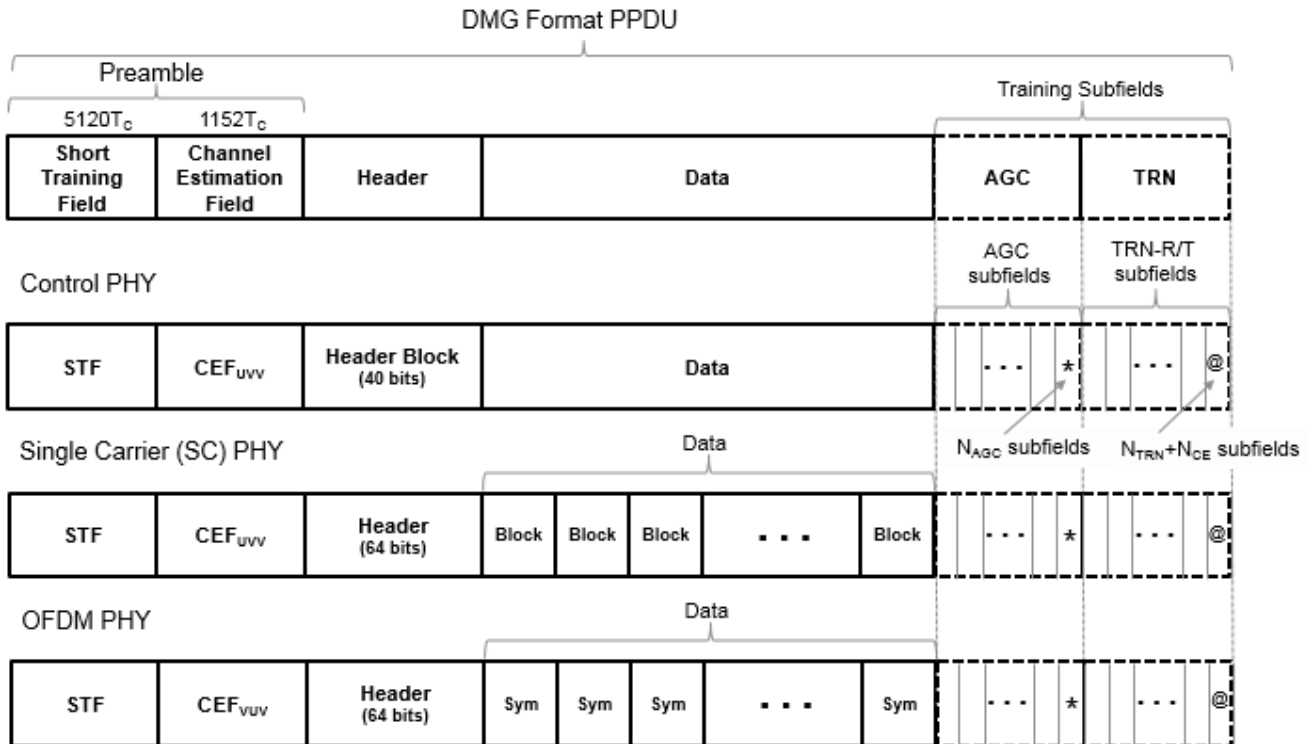
The single-carrier chip timing, $T_C = 1/F_C = 0.57$ ns. For more information, see Waveform Sampling Rate on the `wlanWaveformGenerator` function reference page.

The supported DMG format PPDU field structures each contain these fields:

- The preamble contains a short training field (STF) and channel estimation field (CEF). The preamble is used for packet detection, AGC, frequency offset estimation, synchronization, indication of modulation type (Control, SC, or OFDM), and channel estimation. The format of the preamble is common to the Control, SC, and OFDM PHY packets.

  - The STF is composed of Golay *Ga* sequences as specified in section 20.3.6.2 of [1].
  - The CEF is composed of Golay *Gu* and *Gv* sequences as specified in section 20.3.6.2 of [1].

    - When the header and data fields of the packet are modulated using a single carrier (control PHY and SC PHY), the Golay sequencing for the CEF waveform is shown in Figure 20-6 of [1].
    - When the header and data fields of the packet are modulated using OFDM (OFDM PHY), the Golay sequencing for the CEF waveform is shown in Figure 20-7 of [1].

- The header field is decoded by the receiver to determine transmission parameters.
- The data field is variable in length. It carries the user data payload.
- The training fields (AGC and TRN-R/T subfields) are optional. They can be included to refine beamforming.

Section 20.3 of [1] specifies the common aspects of the DMG PPDU packet structure. The PHY modulation-specific aspects of the packet structure are specified in these sections:

- The DMG control PHY packet structure is specified in Section 20.4.
- The DMG OFDM PHY packet structure is specified in Section 20.5.
- The DMG SC PHY packet structure is specified in Section 20.6.

S1G Format PPDU Field Structure

In S1G, there are three transmission modes:

- ≥2 MHz long preamble mode
- ≥2 MHz short preamble mode
- 1 MHz mode

Each transmission mode has a specific PPDU preamble structure:

- An S1G ≥2 MHz long preamble mode PPDU supports single-user and multi-user transmissions. The long preamble PPDU consists of two portions; the omni-directional portion and the beam-changeable portion.

S1G_LONG Format PPDU

	Omni Portion				Data Portion		
2 symbols	2 symbols	2 symbols	1 symbol	1 symbol per D-LTF	1 symbol		
STF	LTF1	SIG-A	D-STF	D-LTF1 ... D-LTFN$_{LTF}$	SIG-B	Data	

GI2	LTS	LTS
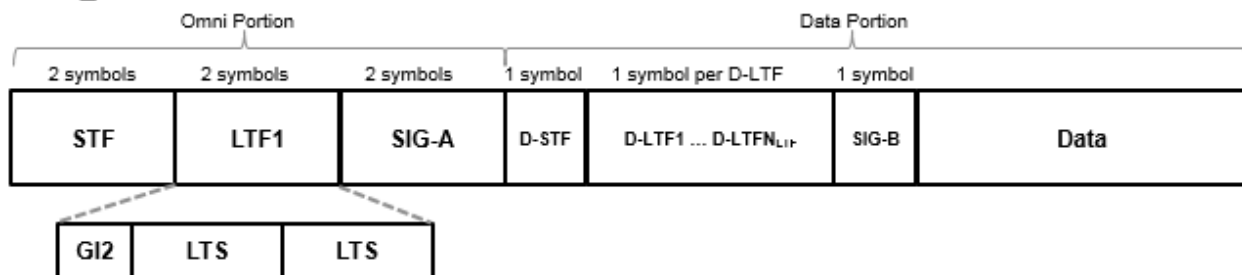
- The omni-directional portion is transmitted to all users without beamforming. It consists of three fields:

    - The short training field (STF) is used for coarse synchronization.
    - The long training field (LTF1) is used for fine synchronization and initial channel estimation.
    - The signal A field (SIG-A) is decoded by the receiver to determine transmission parameters relevant to all users.
- The data portion can be beamformed to each user. It consists of four fields:

    - The beamformed short training field (D-STF) is used by the receiver for automatic gain control.
    - The beamformed long training fields (D-LTF-N) are used for MIMO channel estimation.
    - The signal B field (SIG-B) in a multi-user transmission, signals the MCS for each user. In a single-user transmission, the MCS is signaled in the SIG-A field of the omni-directional portion of the preamble. Therefore, in a single-user transmission the SIG-B symbol transmitted is an exact repetition of the first D-LTF. This repetition allows for improved channel estimation.

- The data field is variable in length. It carries the user data payload.
- An S1G ≥2 MHz short preamble mode PPDU supports single-user transmissions. All fields in the PPDU can be beamformed.

S1G_SHORT Format PPDU
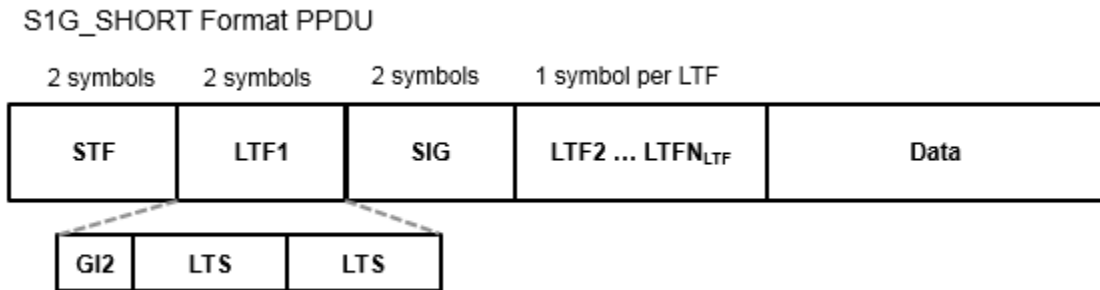
The PPDU consists of these five fields:

- The short training field (STF) is used for coarse synchronization.
- The first long training field (LTF1) is used for fine synchronization and initial channel estimation.
- The signaling field (SIG) is decoded by the receiver to determine transmission parameters.
- The subsequent long training fields (LTF2-N) are used for MIMO channel estimation. $N_{\text{SYMBOLS}} = 1$ per subsequent LTF
- The data field is variable in length. It carries the user data payload.
- An S1G 1 MHz mode PPDU supports single-user transmissions. It is composed of the same five fields as the S1G ≥2 MHz short preamble mode PPDU and all fields can be beamformed. An S1G 1 MHz mode PPDU has longer STF, LTF1, and SIG fields, so this mode can achieve sensitivity that is similar to the S1G ≥2 MHz short-preamble mode transmissions.

S1G_1M Format PPDU
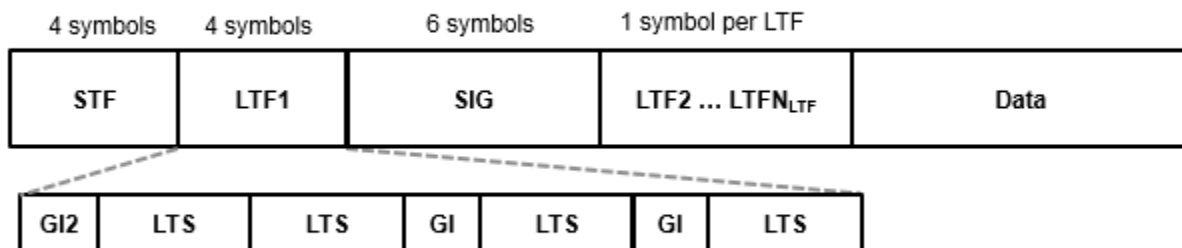
VHT, HT-Mixed, and Non-HT Format PPDU Field Structures

The field structure for VHT, HT, and non-HT PPDUs consist of preamble and data portions. The legacy preamble fields (L-STF, L-LTF, and L-SIG) are common to VHT, HT, and non-HT format preambles.

VHT and HT format preamble fields include additional format-specific training and signaling fields. Each format defines a data field for transmission of user payload data.
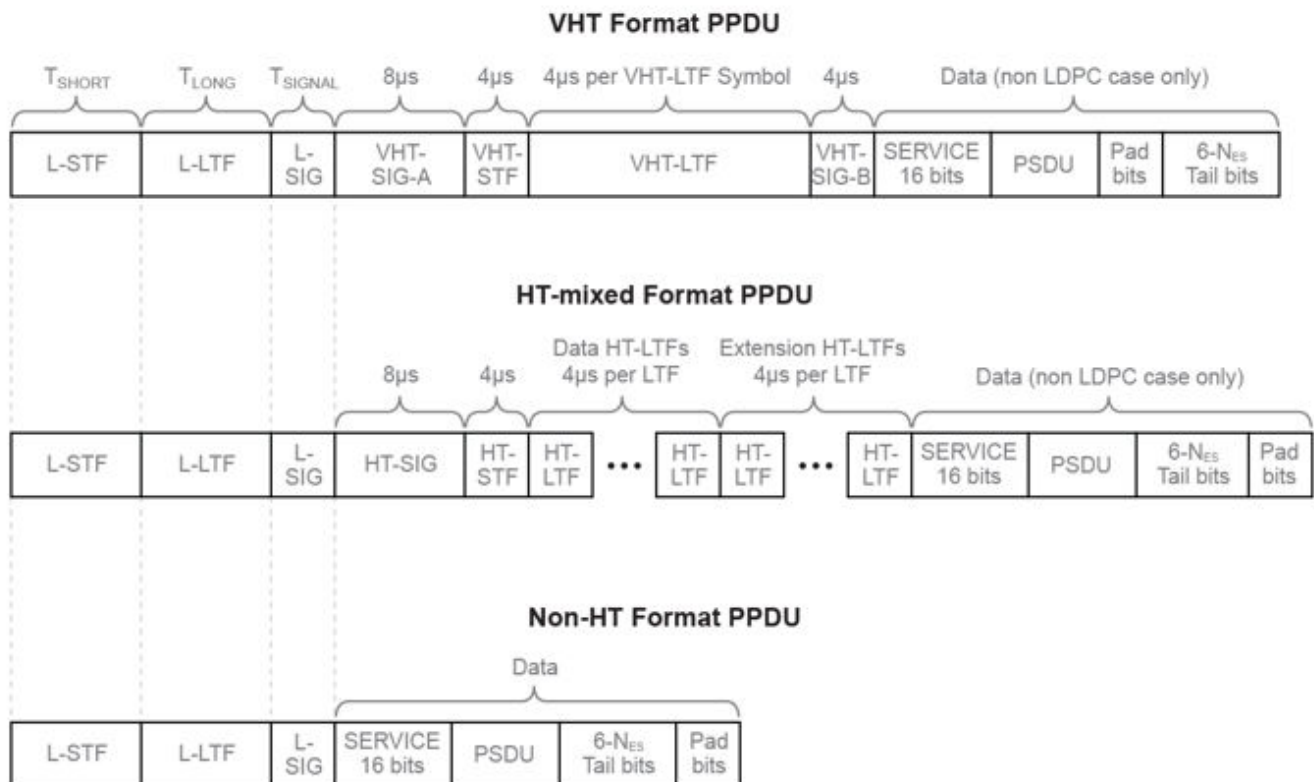
VHT Format PPDU

HT-mixed Format PPDU

Non-HT Format PPDU

PPDU Field Abbreviation	Description
L-STF	Non-HT Short Training field
L-LTF	Non-HT Long Training field
L-SIG	Non-HT SIGNAL field
HT-SIG	HT SIGNAL field
HT-STF	HT Short Training field
HT-LTF	HT Long Training field, multiple HT-LTFs are transmitted as indicated by the MCS
VHT-SIG-A	VHT Signal A field
VHT-STF	VHT Short Training field
VHT-LTF	VHT Long Training field
VHT-SIG-B	VHT Signal B field
Data	VHT, HT, and non-HT Data fields include the service bits, PSDU, tail bits, and pad bits

For more information, see section 19.3.2 of [1].

Non-HT (Legacy) Short Training Field

The legacy short training field (L-STF) is the first field of the 802.11 OFDM PLCP legacy preamble. The L-STF is a component of VHT, HT, and non-HT PPDUs.
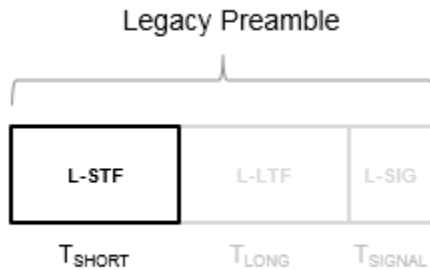
The L-STF duration varies with channel bandwidth.

Channel Bandwidth (MHz)	Subcarrier Frequency Spacing, Δ_F (kHz)	Fast Fourier Transform (FFT) Period ($T_{FFT} = 1 / \Delta_F$)	L-STF Duration ($T_{SHORT} = 10 \times T_{FFT} / 4$)
20, 40, 80, and 160	312.5	3.2 μs	8 μs
10	156.25	6.4 μs	16 μs
5	78.125	12.8 μs	32 μs

Because the sequence has good correlation properties, it is used for start-of-packet detection, for coarse frequency correction, and for setting the AGC. The sequence uses 12 of the 52 subcarriers that are available per 20 MHz channel bandwidth segment. For 5 MHz, 10 MHz, and 20 MHz bandwidths, the number of channel bandwidths segments is 1.

Non-HT (Legacy) Long Training Field

The legacy long training field (L-LTF) is the second field in the 802.11 OFDM PLCP legacy preamble. The L-LTF is a component of VHT, HT, and non-HT PPDUs.
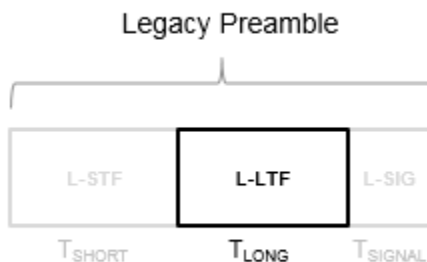
Channel estimation, fine frequency offset estimation, and fine symbol timing offset estimation rely on the L-LTF.

The L-LTF is composed of a cyclic prefix (CP) followed by two identical long training symbols (C1 and C2). The CP consists of the second half of the long training symbol.

L-LTF

CP	C1	C2
T_{GI2}	T_{FFT}	T_{FFT}
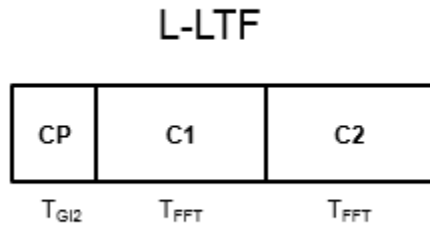
The L-LTF duration varies with channel bandwidth.

Channel Bandwidth (MHz)	Subcarrier Frequency Spacing, Δ_F (kHz)	Fast Fourier Transform (FFT) Period ($T_{FFT} = 1 / \Delta_F$)	Cyclic Prefix or Training Symbol Guard Interval (GI2) Duration ($T_{GI2} = T_{FFT} / 2$)	L-LTF Duration ($T_{LONG} = T_{GI2} + 2 \times T_{FFT}$)
20, 40, 80, and 160	312.5	3.2 µs	1.6 µs	8 µs
10	156.25	6.4 µs	3.2 µs	16 µs
5	78.125	12.8 µs	6.4 µs	32 µs

Non-HT (Legacy) Signal Field

The legacy signal (L-SIG) field is the third field of the 802.11 OFDM PLCP legacy preamble. It consists of 24 bits that contain rate, length, and parity information. The L-SIG is a component of HE, VHT, HT, and non-HT PPDUs. It is transmitted using BPSK modulation with rate 1/2 binary convolutional coding (BCC).

Legacy Preamble

| L-STF	L-LTF	L-SIG
T_{SHORT}	T_{LONG}	T_{SIGNAL}
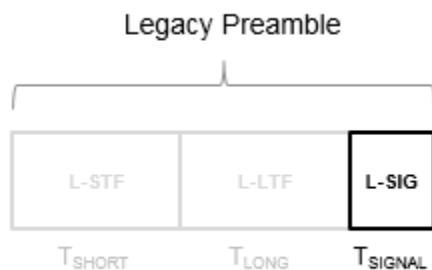
The L-SIG is one OFDM symbol with a duration that varies with channel bandwidth.

Channel Bandwidth (MHz)	Subcarrier frequency spacing, Δ_F (kHz)	Fast Fourier Transform (FFT) period ($T_{FFT} = 1 / \Delta_F$)	Guard Interval (GI) Duration ($T_{GI} = T_{FFT} / 4$)	L-SIG duration ($T_{SIGNAL} = T_{GI} + T_{FFT}$)
20, 40, 80, and 160	312.5	3.2 µs	0.8 µs	4 µs
10	156.25	6.4 µs	1.6 µs	8 µs
5	78.125	12.8 µs	3.2 µs	16 µs

The L-SIG contains packet information for the received configuration,
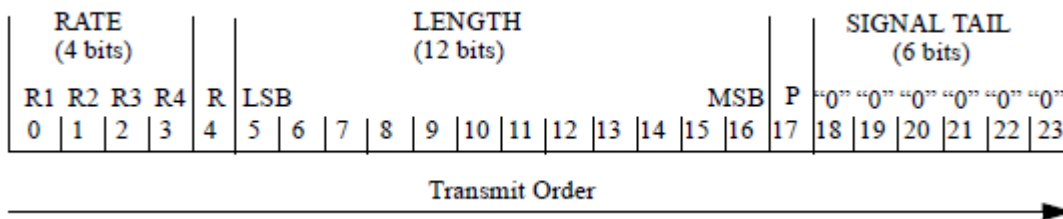
- Bits 0 through 3 specify the data rate (modulation and coding rate) for the non-HT format.

Rate (bits 0–3)	Modulation	Coding rate (R)	Data Rate (Mb/s)		
			20 MHz channel bandwidth	10 MHz channel bandwidth	5 MHz channel bandwidth
1101	BPSK	1/2	6	3	1.5
1111	BPSK	3/4	9	4.5	2.25
0101	QPSK	1/2	12	6	3
0111	QPSK	3/4	18	9	4.5
1001	16-QAM	1/2	24	12	6
1011	16-QAM	3/4	36	18	9
0001	64-QAM	2/3	48	24	12
0011	64-QAM	3/4	54	27	13.5

For HT and VHT formats, the L-SIG rate bits are set to `'1 1 0 1'`. Data rate information for HT and VHT formats is signaled in format-specific signaling fields.

- Bit 4 is reserved for future use.

- Bits 5 through 16:

  - For non-HT, specify the data length (amount of data transmitted in octets) as described in Table 17-1 and section 10.26.4 IEEE Std 802.11-2016.

  - For HT-mixed, specify the transmission time as described in sections 19.3.9.3.5 and 10.26.4 of IEEE Std 802.11-2016.

  - For VHT, specify the transmission time as described in section 21.3.8.2.4 of IEEE Std 802.11-2016.

- Bit 17 has the even parity of bits 0 through 16.
- Bits 18 through 23 contain all zeros for the signal tail bits.

Note Signaling fields added for HT (`wlanHTSIG`) and VHT (`wlanVHTSIGA`, `wlanVHTSIGB`) formats provide data rate and configuration information for those formats.

- For the HT-mixed format, section 19.3.9.4.3 of IEEE Std 802.11-2016 describes HT-SIG bit settings.
- For the VHT format, sections 21.3.8.3.3 and 21.3.8.3.6 of IEEE Std 802.11-2016 describe bit settings for the VHT-SIG-A and VHT-SIG-B fields, respectively.

Non-HT Data Field

The non-high throughput Data (non-HT Data) field is used to transmit MAC frames and is composed of a service field, a PSDU, tail bits, and pad bits.
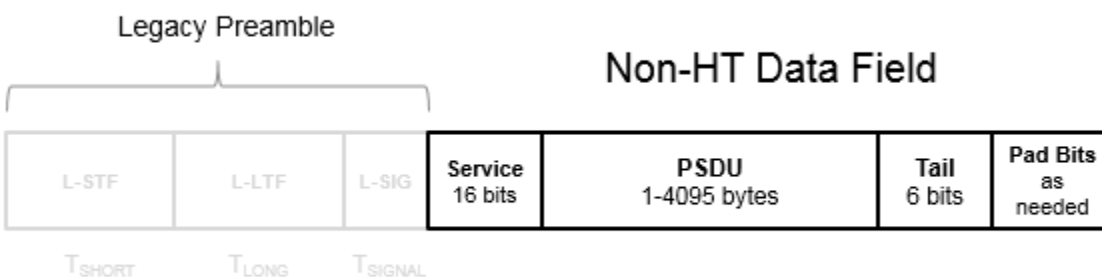
- **Service field** — Contains 16 zeros to initialize the data scrambler.
- **PSDU** — Variable-length field containing the PLCP service data unit (PSDU).
- **Tail** — Tail bits required to terminate a convolutional code. The field uses six zeros for the single encoding stream.
- **Pad Bits** — Variable-length field required to ensure that the non-HT data field contains an integer number of symbols.

Processing of an 802.11a data field is defined in section 17.3.5 of [3].

The six tail bits are set to zero after a 127-bit scrambling sequence has been applied to the full data field. The receiver uses the first seven bits of the service field to determine the initial state of the scrambler. Rate 1/2 BCC encoding is performed on the scrambled data. The zeroed tail bits cause the BCC encoder to return to a zero state. Puncturing is applied as needed for the selected rate.

The coded data is grouped into several bits per symbol, and two permutations of block interleaving are applied to each group of data. The groups of bits are then modulated to the selected rate (BPSK, QPSK, 16-QAM, or 64-QAM) and the complex symbols are then mapped onto corresponding

subcarriers. For each symbol, the pilot subcarriers are inserted. An IFFT is used to transform each symbol group to the time domain and the cyclic prefix is prepended.

The final processing preceding DAC up-conversion to RF and the power amplifier is to apply a pulse shaping filter on the data to smooth transitions between symbols. The standard provides an example pulse shaping function but does not specifically require one.

High Throughput Signal Field

The high throughput signal (HT-SIG) field is located between the L-SIG field and HT-STF and is part of the HT-mixed format preamble. It is composed of two symbols, HT-SIG$_1$ and HT-SIG$_2$.
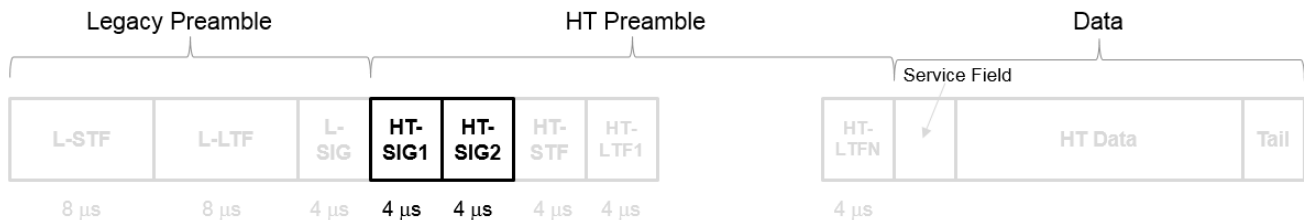
HT-SIG carries information used to decode the HT packet, including the MCS, packet length, FEC coding type, guard interval, number of extension spatial streams, and whether there is payload aggregation. The HT-SIG symbols are also used for auto-detection between HT-mixed format and legacy OFDM packets.
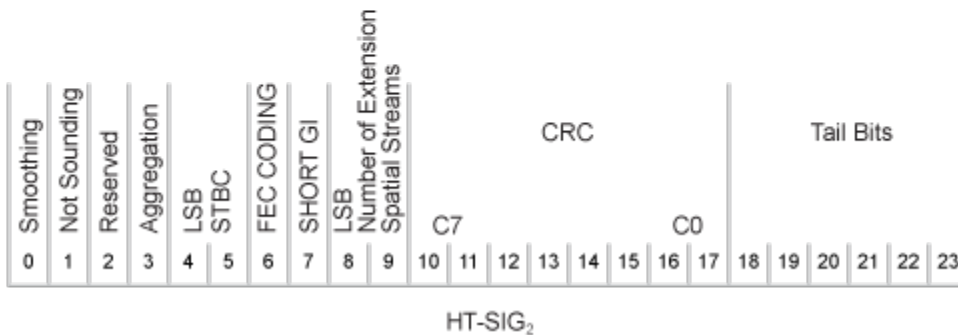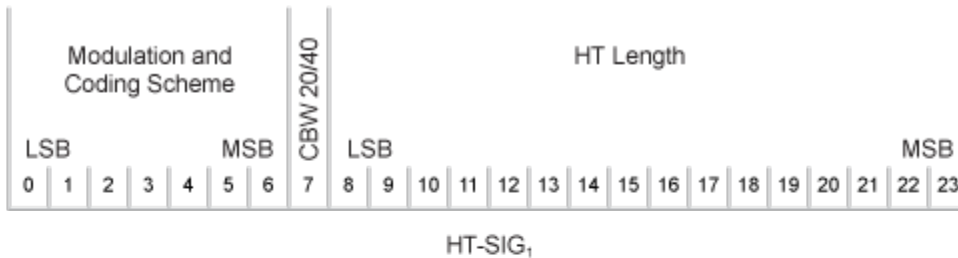
HT-SIG$_1$

HT-SIG$_2$

For a detailed description of the HT-SIG field, see Section 19.3.9.4.3 of IEEE Std 802.11-2016.

High Throughput Short Training Field

The high throughput short training field (HT-STF) is located between the HT-SIG and HT-LTF fields of an HT-mixed packet. The HT-STF is 4 μs in length and is used to improve automatic gain control estimation for a MIMO system. For a 20 MHz transmission, the frequency sequence used to construct the HT-STF is identical to that of the L-STF. For a 40 MHz transmission, the upper subcarriers of the HT-STF are constructed from a frequency-shifted and phase-rotated version of the L-STF.

Legacy Preamble			HT Preamble					Data		
L-STF	L-LTF	L-SIG	HT-SIG1	HT-SIG2	**HT-STF**	HT-LTF1		HT-LTFN	Service Field · HT Data	Tail
8 μs	8 μs	4 μs	4 μs	4 μs	**4 μs**	4 μs		4 μs		
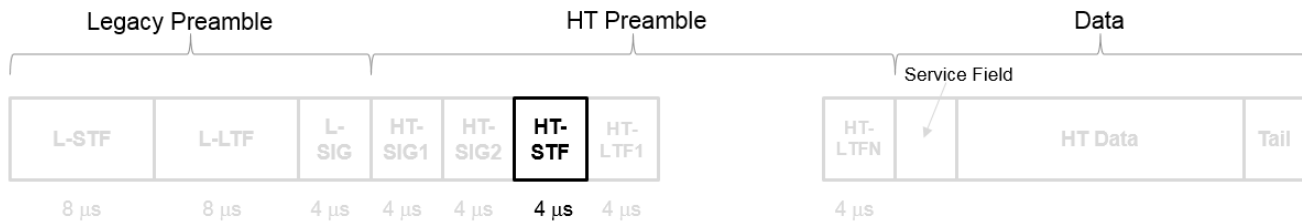
High Throughput Long Training Fields

The high throughput long training field (HT-LTF) is located between the HT-STF and data field of an HT-mixed packet.

Legacy Preamble			HT Preamble						Data		
L-STF	L-LTF	L-SIG	HT-SIG1	HT-SIG2	HT-STF	**HT-LTF1**	**. . .**	**HT-LTFN**	Service Field · HT Data	Tail
8 μs	8 μs	4 μs	4 μs	4 μs	4 μs	**4 μs**		**4 μs**		
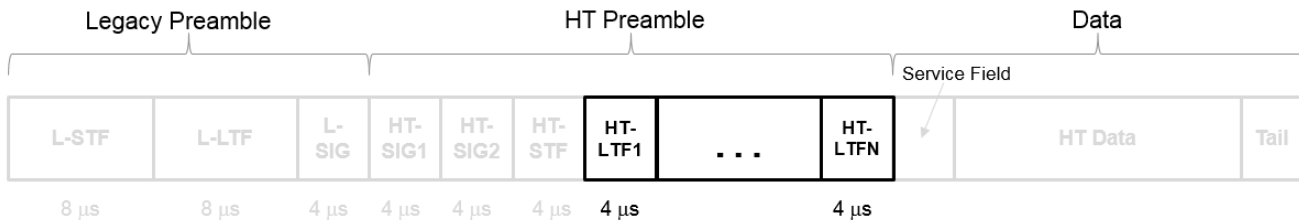
As described in Section 19.3.9.4.6 of IEEE Std 802.11-2016, the receiver can use the HT-LTF to estimate the MIMO channel between the set of QAM mapper outputs (or, if STBC is applied, the STBC encoder outputs) and the receive chains. The HT-LTF portion has one or two parts. The first part consists of one, two, or four HT-LTFs that are necessary for demodulation of the HT-Data portion of the PPDU. These HT-LTFs are referred to as HT-DLTFs. The optional second part consists of zero, one, two, or four HT-LTFs that can be used to sound extra spatial dimensions of the MIMO channel not utilized by the HT-Data portion of the PPDU. These HT-LTFs are referred to as HT-ELTFs. Each HT long training symbol is 4 μs. The number of space-time streams and the number of extension streams determines the number of HT-LTF symbols transmitted.

Tables 19-12, 19-13 and 90-14 from IEEE Std 802.11-2012 are reproduced here.

N_{STS} Determination	N_{HTDLTF} Determination	N_{HTELTF} Determination
Table 19-12 defines the number of space-time streams (N_{STS}) based on the number of spatial streams (N_{SS}) from the MCS and the STBC field.	Table 19-13 defines the number of HT-DLTFs required for the N_{STS}.	Table 19-14 defines the number of HT-ELTFs required for the number of extension spatial streams (N_{ESS}). N_{ESS} is defined in HT-SIG$_2$.

N_{STS} Determination			N_{HTDLTF} Determination		N_{HTELTF} Determination	
N_{SS} from MCS	STBC field	N_{STS}	N_{STS}	N_{HTDLTF}	N_{ESS}	N_{HTELTF}
1	0	1	1	1	0	0
1	1	2	2	2	1	1
2	0	2	3	4	2	2
2	1	3	4	4	3	4
2	2	4				
3	0	3				
3	1	4				
4	0	4				

Additional constraints include:

- $N_{HTLTF} = N_{HTDLTF} + N_{HTELTF} \leq 5$.
- $N_{STS} + N_{ESS} \leq 4$.

  - When $N_{STS} = 3$, N_{ESS} cannot exceed one.
  - If $N_{ESS} = 1$ when $N_{STS} = 3$ then $N_{HTLTF} = 5$.

HT Data Field

The HT-Data field follows the last HT-long training field (HT-LTF) of an HT-mixed packet.
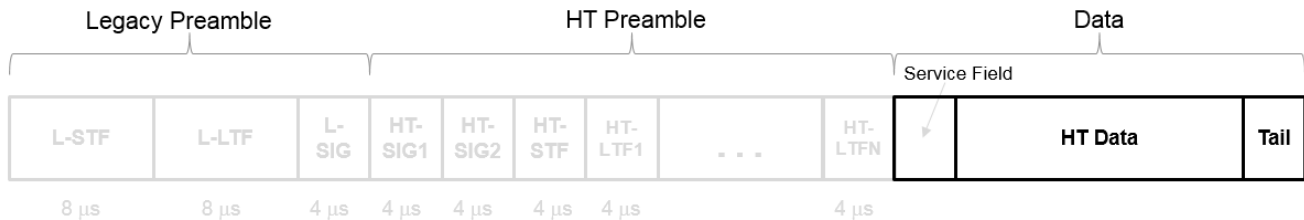
The HT-Data field is carries one or more frames from the medium access control (MAC) layer and consists of four subfields.

- **Service** — Contains 16 zeros to initialize the data scrambler
- **PSDU** — Variable-length field containing a PLCP service data unit (PSDU)
- **Tail** — Contains six zeros for each encoding stream, required to terminate a convolutional code

- **Pad Bits** — Variable-length field required to ensure that the HT-Data field consists of an integer number of symbols

Very High Throughput SIG-A Field

The very high throughput signal A (VHT-SIG-A) field contains information required to interpret VHT format packets. Similar to the non-HT signal (L-SIG) field for the non-HT OFDM format, this field stores the actual rate value, channel coding, guard interval, MIMO scheme, and other configuration details for the VHT format packet. Unlike the HT-SIG field, this field does not store the packet length information. Packet length information is derived from L-SIG and is captured in the VHT-SIG-B field for the VHT format.

For a detailed description of the VHT-SIG-A field, see section 21.3.8.3.3 of IEEE Std 802.11-2016. The VHT-SIG-A field consists of two symbols: VHT-SIG-A1 and VHT-SIG-A2. These symbols are located between the L-SIG and the VHT-STF portion of the VHT format PPDU.

VHT-SIG-A1 Structure

Composite Name:	BW	Reserved	STBC	Group ID	NSTS/Partial AID				TXOP_PS_NOT_ALLOWED	Reserved
SU Name:					SU NSTS	Partial AID				
MU Name:					MU[0] NSTS	MU[1] NSTS	MU[2] NSTS	MU[3] NSTS		
Bits:	2	1	1	6	3	3	3	3	1	1

VHT-SIG-A2 Structure

Composite Name:	Short GI	Short GI NSYM Disambiguation	SU/MU[0] Coding	LDPC Extra OFDM Symbol	SU VHT-MCS/MU[1-3] Coding				Beamformed	Reserved	CRC	Tail
SU Name:					SU VHT-MCS				Beamformed			
MU Name:					MU[1] Coding	MU[2] Coding	MU[3] Coding	Reserved	Reserved			
Bits:	1	1	1	1	1	1	1	1	1	1	8	6

The VHT-SIG-A field includes these components. The bit field structures for VHT-SIG-A1 and VHT-SIG-A2 vary for single user or multiuser transmissions.

- **BW** — A two-bit field that indicates 0 for 20 MHz, 1 for 40 MHz, 2 for 80 MHz, or 3 for 160 MHz.
- **STBC** — A bit that indicates the presence of space-time block coding.

- **Group ID** — A six-bit field that indicates the group and user position assigned to a STA.
- **N$_{STS}$** — A three-bit field for a single user or 4 three-bit fields for a multiuser scenario, that indicates the number of space-time streams per user.
- **Partial AID** — An identifier that combines the association ID and the BSSID.
- **TXOP_PS_NOT_ALLOWED** — An indicator bit that shows if client devices are allowed to enter dose state. This bit is set to false when the VHT-SIG-A structure is populated, indicating that the client device is allowed to enter dose state.
- **Short GI** — A bit that indicates use of the 400 ns guard interval.
- **Short GI NSYM Disambiguation** — A bit that indicates if an extra symbol is required when the short GI is used.
- **SU/MU[0] Coding** — A bit field that indicates if convolutional or LDPC coding is used for a single user or for user MU[0] in a multiuser scenario.
- **LDPC Extra OFDM Symbol** — A bit that indicates if an extra OFDM symbol is required to transmit the data field.
- **MCS** — A four-bit field.

  - For a single user scenario, it indicates the modulation and coding scheme used.
  - For a multiuser scenario, it indicates use of convolutional or LDPC coding and the MCS setting is conveyed in the VHT-SIG-B field.

- **Beamformed** — An indicator bit set to 1 when a beamforming matrix is applied to the transmission.
- **CRC** — An eight-bit field used to detect errors in the VHT-SIG-A transmission.
- **Tail** — A six-bit field used to terminate the convolutional code.

Very High Throughput Short Training Field

The very high throughput short training field (VHT-STF) is a single OFDM symbol (4 μs in length) that is used to improve automatic gain control estimation in a MIMO transmission. It is located between the VHT-SIG-A and VHT-LTF portions of the VHT packet.

The frequency domain sequence used to construct the VHT-STF for a 20 MHz transmission is identical to the L-STF sequence. Duplicate L-STF sequences are frequency shifted and phase rotated to support VHT transmissions for the 40 MHz, 80 MHz, and 160 MHz channel bandwidths. As such, the L-STF and HT-STF are subsets of the VHT-STF.

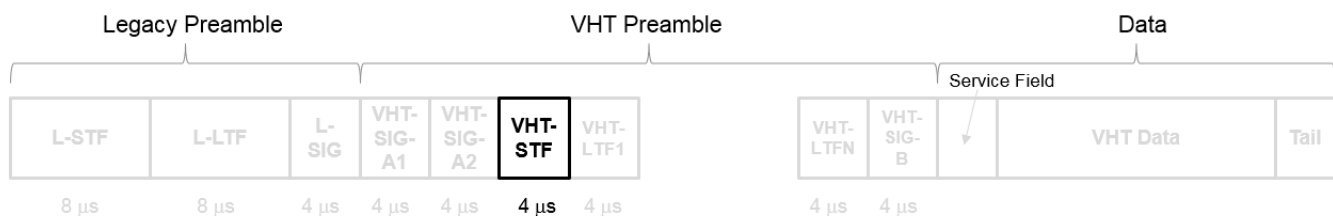For a detailed description of the VHT-STF, see section 21.3.8.3.4 of IEEE Std 802.11-2016.

Very High Throughput Long Training Fields

The very high throughput long training field (VHT-LTF) is located between the VHT-STF and VHT-SIG-B portion of the VHT packet.

Legacy Preamble			VHT Preamble						Data			
L-STF	L-LTF	L-SIG	VHT-SIG-A1	VHT-SIG-A2	VHT-STF	VHT-LTF1	. . .	VHT-LTFN	VHT-SIG-B	Service Field ↓	VHT Data	Tail
8 µs	8 µs	4 µs	4 µs	4 µs	4 µs	4 µs		4 µs	4 µs			
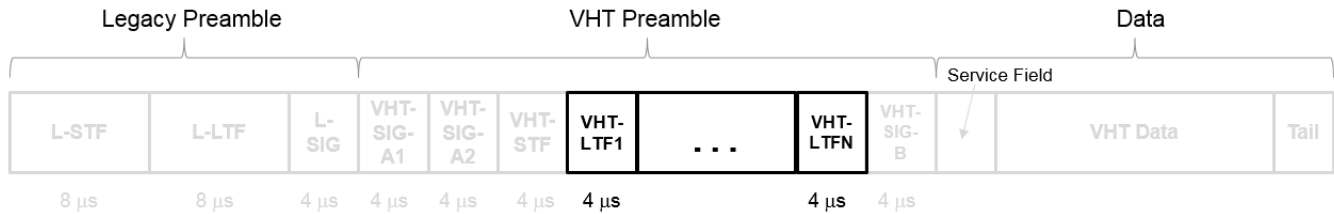
It is used for MIMO channel estimation and pilot subcarrier tracking. The VHT-LTF includes one VHT long training symbol for each spatial stream indicated by the selected MCS. Each symbol is 4 µs long. A maximum of eight symbols are permitted in the VHT-LTF.

For a detailed description of the VHT-LTF, see section 21.3.8.3.5 of IEEE Std 802.11-2016.

Very High Throughput SIG-B Field

The very high throughput signal B field (VHT-SIG-B) is used for multiuser scenario to set up the data rate and to fine-tune MIMO reception. It is modulated using MCS 0 and is transmitted in a single OFDM symbol.

The VHT-SIG-B field consists of a single OFDM symbol located between the VHT-LTF and the data portion of the VHT format PPDU.

Legacy Preamble			VHT Preamble							Data		
L-STF	L-LTF	L-SIG	VHT-SIG-A1	VHT-SIG-A2	VHT-STF	VHT-LTF1		VHT-LTFN	VHT-SIG-B	Service Field ↓	VHT Data	Tail
8 µs	8 µs	4 µs	4 µs	4 µs	4 µs	4 µs		4 µs	4 µs			
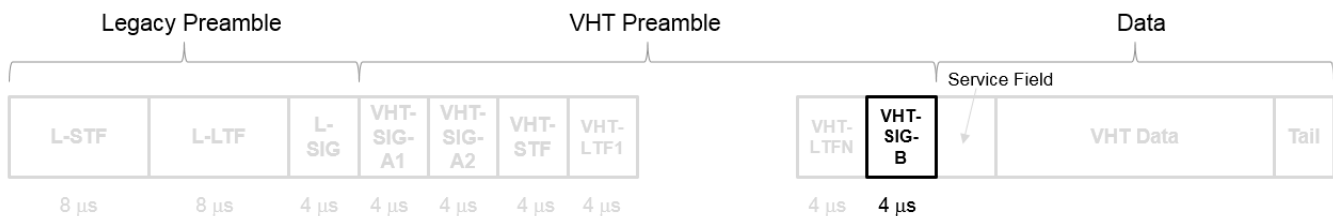
The very high throughput signal B (VHT-SIG-B) field contains the actual rate and A-MPDU length value per user. For a detailed description of the VHT-SIG-B field, see section 21.3.8.3.6 of IEEE Std 802.11-2016. The number of bits in the VHT-SIG-B field varies with the channel bandwidth and the assignment depends on whether single user or multiuser scenario in allocated. For single user configurations, the same information is available in the L-SIG field but the VHT-SIG-B field is included for continuity purposes.

Field	VHT MU PPDU Allocation (bits)			VHT SU PPDU Allocation (bits)			Description
	20 MHz	**40 MHz**	**80 MHz, 160 MHz**	**20 MHz**	**40 MHz**	**80 MHz, 160 MHz**	
VHT-SIG-B	B0-15 (16)	B0-16 (17)	B0-18 (19)	B0-16 (17)	B0-18 (19)	B0-20 (21)	A variable-length field that indicates the size of the data payload in four-byte units. The length of the field depends on the channel bandwidth.
VHT-MCS	B16-19 (4)	B17-20 (4)	B19-22 (4)	N/A	N/A	N/A	A four-bit field that is included for multiuser scenarios only.
Reserved	N/A	N/A	N/A	B17–19 (3)	B19-20 (2)	B21-22 (2)	All ones
Tail	B20-25 (6)	B21-26 (6)	B23-28 (6)	B20-25 (6)	B21-26 (6)	B23-28 (6)	Six zero-bits used to terminate the convolutional code.
Total # bits	26	27	29	26	27	29	
Bit field repetition	1	2	4 *For 160 MHz, the 80 MHz channel is repeated twice.*	1	2	4 *For 160 MHz, the 80 MHz channel is repeated twice.*	

For a null data packet (NDP), the VHT-SIG-B bits are set according to Table 21-15 of IEEE Std 802.11-2016.

VHT Data Field

The VHT-Data field carries one or more frames from the medium access control (MAC) layer. This field follows the VHT-SIG-B field in a VHT PPDUs.
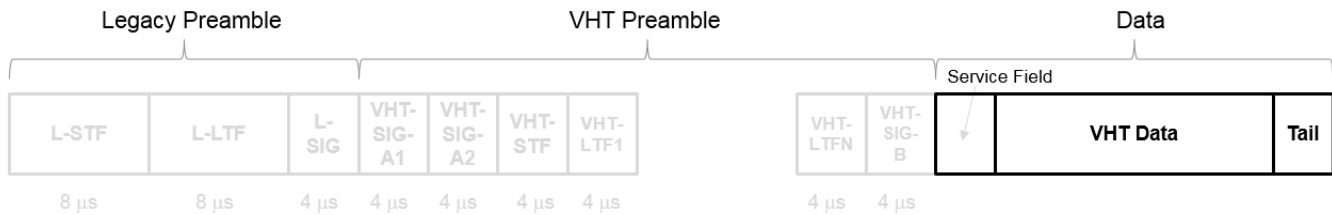
For a detailed description of the VHT-Data field, see section 21.3.10 of IEEE Std 802.11-2016. The VHT Data field consists of four subfields.
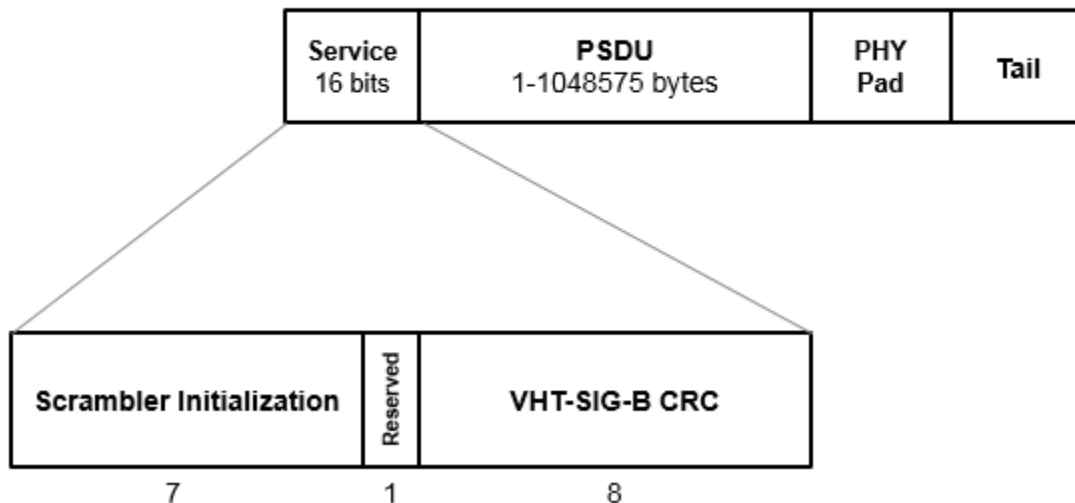
- **Service field** — Contains a seven-bit scrambler initialization state, one bit reserved for future considerations, and eight bits for the VHT-SIG-B cyclic redundancy check (CRC) field
- **PSDU** — Variable-length field containing a PLCP service data unit
- **PHY Pad** — Variable number of bits passed to the transmitter to create a complete OFDM symbol
- **Tail** — Bits required to terminate a convolutional code (not required when the transmission uses LDPC channel coding)

References

[1] IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[2] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[3] IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016 as amended by IEEE Std 802.11ai™-2016). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[4] Perahia, E., and R. Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. 2nd Edition. United Kingdom: Cambridge University Press, 2013.

See Also

"Waveform Generation" on page 3-22 | "What Is WLAN?" on page 2-2 | "Mapping 802.11 Standards to WLAN Toolbox Configuration Objects" on page 3-2 | "HE MU Transmission" on page 3-15

Packet Size and Duration Dependencies

WLAN standards specify a maximum packet duration (*TXTIME*) for the various formats. The HE and S1G formats additionally specify the maximum PSDU length (*PSDU_LENGTH*) and number of symbols (N_{SYM}). These WLAN properties are functions of transmission properties set in WLAN Toolbox configuration objects. The settings of WLAN format-specific configuration objects are validated when the object is used. Command-line feedback informs you when the configuration violates the packet size or duration limits.

This table indicates relevant properties that help determine the packet duration and length for the various WLAN formats. It also provides references to the IEEE standards for further details.

WLAN Format	Length-Related Validation	Relevant and Dependent Properties

HE	*TXTIME* and *PSDU_LENGTH* require validation.
	
	TXTIME and *PSDU_LENGTH* are defined by the equations in section 27.4.3 of [2].
	
	As specified by *aPPDUMaxTime* and *aPSDUMaxLength* in Table 27-53, the maximum *TXTIME* is 5.484 ms and the maximum *PSDU_Length* is 6,500,631 octets.
		For single user HE and single user extended range HE:
		
		For PSDU_LENGTH:
		
		1 ChannelBandwidth
		2 NumSpaceTimeStreams
		3 STBC
		4 MCS
		5 DCM
		6 ChannelCoding
		7 APEPLength
		
		For TXTIME:
		
		1 All the above PSDU_LENGTH properties
		2 ExtendedRange
		3 GuardInterval
		4 HELTFType
		5 HighDoppler
		6 MidamblePeriodictiy[he_1]
		
		For multi-user HE:
		
		The PSDU_LENGTH is different for each user in the configuration, but the transmit time is the same (like VHT). For PSDU_LENGTH:
		
		• Can vary per user:
		
		  1 User RU size is based on:
		    a AllocationIndex
		    b ChannelBandwidth, which is derived from AllocationIndex
		    c Lower26ToneRU[he_2]
		    d Upper26ToneRU[he_2]
		  2 NumSpaceTimeStreams
		  3 MCS
		  4 DCM
		  5 ChannelCoding
		  6 APEPLength

WLAN Format	Length-Related Validation	Relevant and Dependent Properties
		• Same for all users: • STBC For TXTIME: **1** The above PSDU_LENGTH properties that can vary per user **2** GuardInterval **3** HELTFType **4** SIGBCompression[(he_3)(he_4)] **5** SIGBMCS **6** SIGBDCM **7** HighDoppler **8** MidamblePeriodictiy[(he_1)] Notes: [(he_1)] MidamblePeriodictiy is only relevant when HighDoppler is true. [(he_2)] Lower26ToneRU and Upper26ToneRU are only relevant if the user for which the PSDU_LENGTH is derived is allocated on either of these RUs. [(he_3)] If SIG-B compression is used, there is common field in HE-SIG-B field that affects the TXTIME computation. [(he_4)] The SIGBCompression property is only relevant when a full-band 20 MHz allocation is specified.
DMG	*TXTIME* requires validation. *TXTIME* is defined by the equations in section 20.12.3 of [1]. As specified by *aPPDUMaxTime* in Table 20-32, the maximum *TXTIME* is 2 ms.	**1** MCS [(dmg_1)] **2** PSDULength **3** TrainingLength [(dmg_1)] **4** PacketType [(dmg_1)] **5** BeamTrackingRequest [(dmg_1)] Notes: [(dmg_1)] The property helps determine whether the packet is a beam refinement protocol (BRP) packet containing training fields or if it is a general packet signaling the number of fields to append. For more information, see [1]

WLAN Format	Length-Related Validation	Relevant and Dependent Properties

S1G	*TXTIME*, *PSDU_LENGTH*, and N_{SYM} require validation.
	
	The equations for all three of these characteristics are in section 24.4.3 of [3], and the maximum *TXTIME* and *PSDU_LENGTH* are defined in Table 24-37.
	
	In Table 24-37:
	
	• As specified by *aPPDUMaxTime*, the maximum *TXTIME* is 27.92 ms. This *TXTIME* is the maximum PPDU duration for an S1G_1M PPDU with:
	
	  • A bandwidth of 1 MHz
	  • S1G MCS set to 10
	  • One spatial stream, limited by a PSDU length of 511 octets.
	
	• As specified by *aPPDUMaxLength*, the maximum *PSDU_LENGTH* is 797,159 octets. This PSDU length is the maximum length in octets for an S1G SU PPDU with:
	
	  • A bandwidth of 16 MHz
	  • S1G-MCS set to 9
	  • Four spatial streams, limited by 511 data symbols supported by the *Length* field in the S1G SIG field, excluding the *SERVICE* field and tail bits.
	
	• The maximum N_{SYM} is 511.
		For *TXTIME*, the relevant properties are:
		
		1 `ChannelBandwidth`
		2 `STBC` [s1g_1]
		3 `GuardInterval`
		4 `ChannelCoding` [MU]
		5 `APEPLength` [MU]
		6 `PSDULength` – This property is read-only. When undefined, `PSDULength` is returned as empty of size 1×0. An empty return can happen when the set of property values for the object define an invalid state. [MU]
		7 `MCS` [MU]
		8 `NumSpaceTimeStreams` [MU]
		9 `NumUsers`
		
		For *PSDU_LENGTH* and N_{SYM}, the relevant properties are:
		
		1 `ChannelBandwidth`
		2 `STBC` [s1g_1]
		3 `ChannelCoding` [MU]
		4 `APEPLength` [MU]
		5 `PSDULength` – This property is read-only. When undefined, `PSDULength` is returned as empty, `[]`. An empty return can happen when the set of property values for the object define an invalid state. [MU]
		6 `MCS` [MU]
		7 `NumSpaceTimeStreams` [MU]
		8 `NumUsers`
		
		Notes:
		
		[s1g_1] The property is relevant only when `NumUsers = 1`.
		
		[MU] The property has multiple values for multi-user operation.

WLAN Format	Length-Related Validation	Relevant and Dependent Properties
VHT	*TXTIME* requires validation. *TXTIME* is defined by the equations in section 21.4.3 in [1]. As specified by *aPPDUMaxTime* in Table 21-29, the maximum *TXTIME* is 5.484 ms.	**1** ChannelBandwidth **2** STBC [vht_1] **3** GuardInterval **4** ChannelCoding [MU] **5** APEPLength [MU] **6** PSDULength – This property is read only. When undefined, it is returned as an empty of size 1×0. [MU] **7** MCS [MU] **8** NumSpaceTimeStreams [MU] **9** NumUsers Notes: [vht_1] The property is relevant only when NumUsers = 1. [MU] The property has multiple values for multi-user operation.
HT	*TXTIME* requires validation. *TXTIME* is defined by the equations in section 19.4.3 of [1]. As specified in Table 9-19, the maximum *TXTIME* for HT mixed PPDUs is 5.484 ms.	**1** ChannelBandwidth **2** GuardInterval **3** ChannelCoding **4** PSDULength **5** MCS **6** NumSpaceTimeStreams **7** NumExtensionStreams
non-HT	*TXTIME* requires validation. *TXTIME* for OFDM modulation is defined by the equations in section 17.4.3 of [1]. Based on equation (17–29) and a valid combination of property settings, the maximum *TXTIME* is 21.936 ms.	**1** Modulation [non-ht_1] **2** PSDULength **3** MCS [non-ht_1] **4** DateRate [non-ht_1] Notes: [non-ht_1] DataRate or MCS might be relevant depending on the Modulation setting.

References

[1] IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[2] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[3] IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016 as amended by IEEE Std 802.11ai™-2016). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

See Also

More About

- "WLAN PPDU Structure" on page 2-9

WLAN Radio Frequency Channels

WLAN operates in unlicensed radio frequency (RF) spectrum allocated by governing bodies in individual countries for signal transmissions. Appropriate regulatory bodies specify maximum allowable output power.

Refer to IEEE Std 802.11-2016, Annex E for detailed description of country information, operating classes, and behavior limits. The discussion here is restricted to identification of the WLAN operating frequency channel designations.

In general, the 2.4 GHz and 5 GHz bands of operation designate channels spaced 5 MHz apart, with noted exceptions. As an example, the 2.4 GHz band designates channels 1 through 13 spaced 5 MHz apart plus a 14th channel 12 MHz from channel 13. Defined WLAN channel bandwidths are greater than 5 MHz, therefore cross-channel interference limits the number of designated usable channels. Access point deployments manage interference from neighboring cells by operating on non-overlapping channels. In the United States, the 2.4 GHz band designated usable non-overlapping channels are 1, 6, and 11.
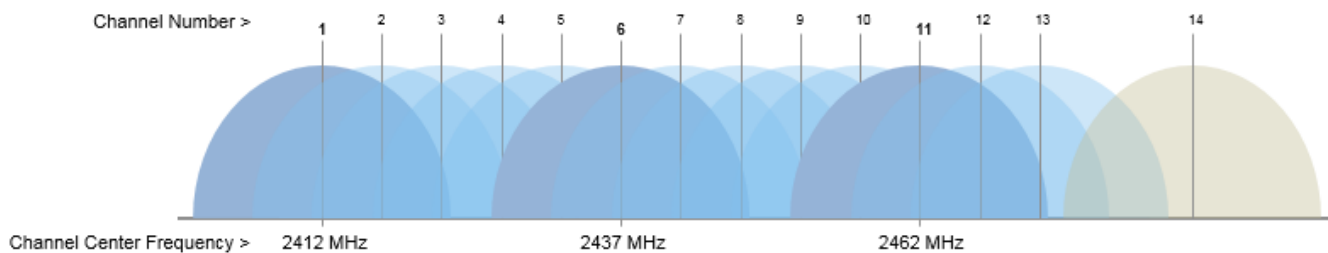
The channel center frequency, F_{CENTER}, is calculated using the starting frequency, F_{START}, and the channel number.

F_{CENTER} in MHz $= F_{START} + (5 \times Channel\ Number)$

Example: Determine the center frequency for channel number 6 in the 2.4 GHz band.

F_{CENTER} in MHz $= 2407 + (5 \times 6) = 2437$ MHz.

802.11 channels		
Channel Number	F_{START}, **Starting Frequency**	**Comments**
1, ..., 13	2407 MHz	For country- and release-specific restrictions, refer to [1]
14	2414 MHz	
132, 133, 134, 136, 137, 138	3000 MHz	
131, ..., 138	3002.5 MHz	
183, ..., 197	4000 MHz	
182, ..., 189	4002.5 MHz	
21, 25	4850 MHz	
11, 13, 15, 17, 19	4890 MHz	
1, ..., 10	4937.5 MHz	

802.11 channels		
Channel Number	F_{START}, **Starting Frequency**	**Comments**
7, ..., 12, 16 34, ..., 60 in increments of 2 64 100, 104, 106, 108 112, 114, 116 120, 122, 124, 128 132, 136, 138 140, 144, 149 153, 155, 157 161, 165, 169 171, ..., 184 in increments of 1	5000 MHz	
6, ..., 11 170, ..., 184 in increments of 1	5002.5 MHz	
1, 2, 3, 4	56.16 GHz	

References

[1] IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[2] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

Acknowledgments

This table lists the copyright owners of content used in the WLAN Toolbox documentation.

Source	Copyright Owner
Content from IEEE Std 802.11-2016	Adapted and reprinted with permission from IEEE. Copyright IEEE 2016. All rights reserved.
Content from IEEE Std 802.11ah-2016	Adapted and reprinted with permission from IEEE. Copyright IEEE 2016. All rights reserved.

Tutorials

Mapping 802.11 Standards to WLAN Toolbox Configuration Objects

WLAN Toolbox configuration objects define transmission parameters for IEEE 802.11 waveforms. This table shows the mapping between 802.11 versions, their associated packet formats, and WLAN Toolbox configuration objects.

802.11 Version	Transmission Packet Format	Toolbox Configuration Object
802.11 b/a/g/j/p	non-HT	`wlanNonHTConfig`
802.11n (Wi-Fi 4)	HT	`wlanHTConfig`
802.11ac (Wi-Fi 5)	VHT	`wlanVHTConfig`
802.11ah	S1G	`wlanS1GConfig`
802.11ad	DMG	`wlanDMGConfig`
802.11ax (Wi-Fi 6)	HE SU, HE ER SU, HE MU	`wlanHESUConfig`
	HE MU	`wlanHEMUConfig`
	HE TB	`wlanHETBConfig`

See Also

"WLAN PPDU Structure" on page 2-9 | "Create Configuration Objects" on page 3-3 | "Waveform Generation" on page 3-22 | "HE MU Transmission" on page 3-15

Create Configuration Objects

WLAN Toolbox configuration objects initialize, store, and validate configuration properties. These properties correspond to parameters that define the characteristics of IEEE 802.11b/a/g/n/j/p/ac/ah/ad/ax waveforms. The functions in the toolbox initialize parameter settings for waveform transmission and reception by using the relevant configuration object properties. Configuration object creation is the first step in many signal transmission and recovery workflows.

Create HE MU Configuration Object

This example shows how to create HE MU configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create an HE MU configuration object with the `AllocationIndex` set to 0 and view the default settings.

```
cfgHEMU = wlanHEMUConfig(0)

cfgHEMU =
  wlanHEMUConfig with properties:

                     RU: {1x9 cell}
                   User: {1x9 cell}
      NumTransmitAntennas: 1
                   STBC: 0
          GuardInterval: 3.2000
              HELTFType: 4
                SIGBMCS: 0
                SIGBDCM: 0
        UplinkIndication: 0
               BSSColor: 0
            SpatialReuse: 0
            TXOPDuration: 127
             HighDoppler: 0

   Read-only properties:
        ChannelBandwidth: 'CBW20'
         AllocationIndex: 0
```

Modify the defaults to specify four transmit antennas.

```
cfgHEMU.NumTransmitAntennas = 4

cfgHEMU =
  wlanHEMUConfig with properties:

                     RU: {1x9 cell}
                   User: {1x9 cell}
      NumTransmitAntennas: 4
                   STBC: 0
          GuardInterval: 3.2000
              HELTFType: 4
```

```
                 SIGBMCS: 0
                 SIGBDCM: 0
         UplinkIndication: 0
                BSSColor: 0
             SpatialReuse: 0
             TXOPDuration: 127
              HighDoppler: 0

     Read-only properties:
          ChannelBandwidth: 'CBW20'
           AllocationIndex: 0
```

Create Object and Override Default Property Values

Create an HE MU configuration object with `AllocationIndex` set to 192. Use `Name,Value` pairs to set the spatial reuse to 3.

```
cfgHEMU = wlanHEMUConfig(192,'SpatialReuse',3)

cfgHEMU =
  wlanHEMUConfig with properties:

                       RU: {[1x1 wlanHEMURU]}
                     User: {[1x1 wlanHEMUUser]}
        NumTransmitAntennas: 1
                     STBC: 0
             GuardInterval: 3.2000
                 HELTFType: 4
           SIGBCompression: 1
                   SIGBMCS: 0
                   SIGBDCM: 0
           UplinkIndication: 0
                  BSSColor: 0
               SpatialReuse: 3
               TXOPDuration: 127
                HighDoppler: 0

     Read-only properties:
          ChannelBandwidth: 'CBW20'
           AllocationIndex: 192
```

Create Single User HE Configuration Object

This example shows how to create single user HE configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a single user HE configuration object and view the default settings.

```
hesu = wlanHESUConfig

hesu =
  wlanHESUConfig with properties:
```

```
        ChannelBandwidth: 'CBW20'
           ExtendedRange: 0
      NumTransmitAntennas: 1
      NumSpaceTimeStreams: 1
           SpatialMapping: 'Direct'
      PreHESpatialMapping: 0
                     STBC: 0
                      MCS: 0
                      DCM: 0
            ChannelCoding: 'LDPC'
               APEPLength: 100
            GuardInterval: 3.2000
                HELTFType: 4
         UplinkIndication: 0
                 BSSColor: 0
              SpatialReuse: 0
             TXOPDuration: 127
               HighDoppler: 0
      NominalPacketPadding: 0
     PostFECPaddingSource: 'mt19937ar with seed'
        PostFECPaddingSeed: 73
```

Modify the defaults to specify an four transmit antennas.

```
hesu.NumTransmitAntennas = 4
```

```
hesu =
  wlanHESUConfig with properties:

        ChannelBandwidth: 'CBW20'
           ExtendedRange: 0
      NumTransmitAntennas: 4
      NumSpaceTimeStreams: 1
           SpatialMapping: 'Direct'
      PreHESpatialMapping: 0
                     STBC: 0
                      MCS: 0
                      DCM: 0
            ChannelCoding: 'LDPC'
               APEPLength: 100
            GuardInterval: 3.2000
                HELTFType: 4
         UplinkIndication: 0
                 BSSColor: 0
              SpatialReuse: 0
             TXOPDuration: 127
               HighDoppler: 0
      NominalPacketPadding: 0
     PostFECPaddingSource: 'mt19937ar with seed'
        PostFECPaddingSeed: 73
```

Create Object and Override Default Property Values

Create a single user HE configuration object. Use `Name,Value` pairs to set the modulation and coding scheme to 9 and to enable space-time block coding.

```
hesu2 = wlanHESUConfig('MCS',9,'STBC',true)

hesu2 =
  wlanHESUConfig with properties:

         ChannelBandwidth: 'CBW20'
            ExtendedRange: 0
       NumTransmitAntennas: 1
       NumSpaceTimeStreams: 1
            SpatialMapping: 'Direct'
        PreHESpatialMapping: 0
                      STBC: 1
                       MCS: 9
                       DCM: 0
            ChannelCoding: 'LDPC'
               APEPLength: 100
            GuardInterval: 3.2000
                 HELTFType: 4
          UplinkIndication: 0
                  BSSColor: 0
              SpatialReuse: 0
              TXOPDuration: 127
               HighDoppler: 0
       NominalPacketPadding: 0
       PostFECPaddingSource: 'mt19937ar with seed'
         PostFECPaddingSeed: 73
```

Create DMG Configuration Object

This example shows how to create DMG configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a DMG configuration object and view the default settings. By default, the configuration object creates properties to model the DMG control PHY.

```
dmg = wlanDMGConfig

dmg =
  wlanDMGConfig with properties:

                          MCS: '0'
               TrainingLength: 0
                    PSDULength: 1000
        ScramblerInitialization: 2
                    Turnaround: 0
```

Model the SC PHY by modifying the defaults to specify an MCS of 5.

```
dmg.MCS = 5

dmg =
  wlanDMGConfig with properties:
```

```
                      MCS: 5
           TrainingLength: 0
               PSDULength: 1000
    ScramblerInitialization: 2
            AggregatedMPDU: 0
                  LastRSSI: 0
                Turnaround: 0
```

For the various configurations, different sets of configuration fields apply and are visible. By changing the MCS setting from 0 to 5, we see that the configured object includes the `AggregationMPDU` and `LastRSSI` fields.

Create Object and Override Default Property Values

Create a DMG configuration object for OFDM PHY. Use `Name,Value` pairs to set the MCS to `14` and specify four training fields.

```
dmg2 = wlanDMGConfig('MCS',14,'TrainingLength',4)

dmg2 =
  wlanDMGConfig with properties:

                       MCS: 14
            TrainingLength: 4
                PacketType: 'TRN-R'
        BeamTrackingRequest: 0
            TonePairingType: 'Static'
                PSDULength: 1000
    ScramblerInitialization: 2
            AggregatedMPDU: 0
                  LastRSSI: 0
                Turnaround: 0
```

Create S1G Configuration Object

This example shows how to create S1G configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a S1G configuration object and view the default settings.

```
s1g = wlanS1GConfig

s1g =
  wlanS1GConfig with properties:

        ChannelBandwidth: 'CBW2'
                Preamble: 'Short'
                NumUsers: 1
     NumTransmitAntennas: 1
     NumSpaceTimeStreams: 1
          SpatialMapping: 'Direct'
                    STBC: 0
```

```
                      MCS: 0
               APEPLength: 256
             GuardInterval: 'Long'
                PartialAID: 37
           UplinkIndication: 0
                     Color: 0
            TravelingPilots: 0
         ResponseIndication: 'None'
         RecommendSmoothing: 1

      Read-only properties:
             ChannelCoding: 'BCC'
                 PSDULength: 258
```

Modify the defaults to specify an 8 MHz channel bandwidth, three transmit antennas, and three space-time streams.

```
s1g.ChannelBandwidth = 'CBW8';
s1g.NumTransmitAntennas = 3;
s1g.NumSpaceTimeStreams = 3

s1g =
  wlanS1GConfig with properties:

          ChannelBandwidth: 'CBW8'
                  Preamble: 'Short'
                  NumUsers: 1
        NumTransmitAntennas: 3
        NumSpaceTimeStreams: 3
             SpatialMapping: 'Direct'
                      STBC: 0
                       MCS: 0
                APEPLength: 256
              GuardInterval: 'Long'
                 PartialAID: 37
            UplinkIndication: 0
                      Color: 0
             TravelingPilots: 0
          ResponseIndication: 'None'
          RecommendSmoothing: 1

      Read-only properties:
             ChannelCoding: 'BCC'
                 PSDULength: 261
```

Create Object and Override Default Property Values

Create a S1G configuration object. Use `Name,Value` pairs to set the MCS to 5 and to specify two transmit antennas.

```
s1g2 = wlanS1GConfig('MCS',5,'NumTransmitAntennas',2)

s1g2 =
  wlanS1GConfig with properties:

          ChannelBandwidth: 'CBW2'
```

```
                Preamble: 'Short'
                NumUsers: 1
      NumTransmitAntennas: 2
      NumSpaceTimeStreams: 1
           SpatialMapping: 'Direct'
                     STBC: 0
                      MCS: 5
                APEPLength: 256
             GuardInterval: 'Long'
                PartialAID: 37
           UplinkIndication: 0
                    Color: 0
           TravelingPilots: 0
         ResponseIndication: 'None'
         RecommendSmoothing: 1

    Read-only properties:
            ChannelCoding: 'BCC'
               PSDULength: 258
```

As currently configured, this object is not a valid S1G configuration. Validation of the object occurs when it is the input to a calling function. When spatial mapping is `'Direct'`, the number of space-time streams must equal the number of transmit antennas. Changing the number of space time streams to match the number of transmit antennas is one option to make the configuration of the object valid.

```
s1g2.NumSpaceTimeStreams = 2

s1g2 =
  wlanS1GConfig with properties:

          ChannelBandwidth: 'CBW2'
                  Preamble: 'Short'
                  NumUsers: 1
       NumTransmitAntennas: 2
       NumSpaceTimeStreams: 2
            SpatialMapping: 'Direct'
                      STBC: 0
                       MCS: 5
                 APEPLength: 256
              GuardInterval: 'Long'
                 PartialAID: 37
            UplinkIndication: 0
                     Color: 0
            TravelingPilots: 0
          ResponseIndication: 'None'
          RecommendSmoothing: 1

    Read-only properties:
            ChannelCoding: 'BCC'
```

```
            PSDULength: 258
```

Create VHT Configuration Object

This example shows how to create VHT configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a VHT configuration object and view the default settings.

```
vht = wlanVHTConfig

vht =
  wlanVHTConfig with properties:

       ChannelBandwidth: 'CBW80'
               NumUsers: 1
    NumTransmitAntennas: 1
    NumSpaceTimeStreams: 1
         SpatialMapping: 'Direct'
                   STBC: 0
                    MCS: 0
          ChannelCoding: 'BCC'
             APEPLength: 1024
           GuardInterval: 'Long'
                GroupID: 63
             PartialAID: 275

  Read-only properties:
             PSDULength: 1035
```

Modify the defaults to specify a 160 MHz channel bandwidth, two transmit antennas, and two space-time streams.

```
vht.ChannelBandwidth = 'CBW160';
vht.NumTransmitAntennas = 2;
vht.NumSpaceTimeStreams = 2

vht =
  wlanVHTConfig with properties:

       ChannelBandwidth: 'CBW160'
               NumUsers: 1
    NumTransmitAntennas: 2
    NumSpaceTimeStreams: 2
         SpatialMapping: 'Direct'
                   STBC: 0
                    MCS: 0
          ChannelCoding: 'BCC'
             APEPLength: 1024
           GuardInterval: 'Long'
                GroupID: 63
             PartialAID: 275
```

```
Read-only properties:
        PSDULength: 1050
```

Create Object and Override Default Property Values

Create a VHT configuration object. Use `Name,Value` pairs to set the MCS to 7 and to specify two transmit antennas.

```
vht2 = wlanVHTConfig('MCS',7,'NumTransmitAntennas',2)

vht2 =
  wlanVHTConfig with properties:

      ChannelBandwidth: 'CBW80'
              NumUsers: 1
   NumTransmitAntennas: 2
   NumSpaceTimeStreams: 1
        SpatialMapping: 'Direct'
                  STBC: 0
                   MCS: 7
         ChannelCoding: 'BCC'
            APEPLength: 1024
          GuardInterval: 'Long'
               GroupID: 63
            PartialAID: 275

  Read-only properties:
            PSDULength: 1167
```

As currently configured, this object is not a valid VHT configuration. Validation of the object occurs when it is the input to a calling function. When spatial mapping is `Direct`, the number of space-time streams must equal the number of transmit antennas. Changing the number of space time streams to match the number of transmit antennas is one option to make the configuration of the object valid.

```
vht2.NumSpaceTimeStreams = 2

vht2 =
  wlanVHTConfig with properties:

      ChannelBandwidth: 'CBW80'
              NumUsers: 1
   NumTransmitAntennas: 2
   NumSpaceTimeStreams: 2
        SpatialMapping: 'Direct'
                  STBC: 0
                   MCS: 7
         ChannelCoding: 'BCC'
            APEPLength: 1024
          GuardInterval: 'Long'
               GroupID: 63
            PartialAID: 275

  Read-only properties:
```

```
        PSDULength: 1166
```

Create HT Configuration Object

This example shows how to create HT configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create an HT configuration object and view the default settings.

```
ht = wlanHTConfig
```

```
ht =
  wlanHTConfig with properties:

        ChannelBandwidth: 'CBW20'
    NumTransmitAntennas: 1
    NumSpaceTimeStreams: 1
          SpatialMapping: 'Direct'
                     MCS: 0
           GuardInterval: 'Long'
           ChannelCoding: 'BCC'
              PSDULength: 1024
          AggregatedMPDU: 0
      RecommendSmoothing: 1
```

Modify the defaults to specify three transmit antennas and two space-time streams.

```
ht.NumTransmitAntennas = 3;
ht.NumSpaceTimeStreams = 2
```

```
ht =
  wlanHTConfig with properties:

        ChannelBandwidth: 'CBW20'
    NumTransmitAntennas: 3
    NumSpaceTimeStreams: 2
    NumExtensionStreams: 0
          SpatialMapping: 'Direct'
                     MCS: 0
           GuardInterval: 'Long'
           ChannelCoding: 'BCC'
              PSDULength: 1024
          AggregatedMPDU: 0
      RecommendSmoothing: 1
```

As the settings of the object are modified, the set of properties that apply for the current configuration are shown. When the number of transmit antennas is more than the number of space-time streams, the number of extension streams property applies and is shown. Also, as currently configured, this object is not a valid HT configuration because the default `'Direct'` spatial mapping requires the number of space-time streams to equal the number of transmit antennas. Validation of the object occurs when it is input to a calling function.

Create Object and Override Default Property Values

Create an HT configuration object. Use `Name,Value` pairs to define a sounding packet by specifying `PSDULength = 0`, and set the number of transmit antennas and space-time streams to 3.

```
ht2 = wlanHTConfig('PSDULength',0,'NumTransmitAntennas',3,'NumSpaceTimeStreams',3)

ht2 =
  wlanHTConfig with properties:

        ChannelBandwidth: 'CBW20'
     NumTransmitAntennas: 3
     NumSpaceTimeStreams: 3
          SpatialMapping: 'Direct'
                     MCS: 0
            GuardInterval: 'Long'
          ChannelCoding: 'BCC'
              PSDULength: 0
           AggregatedMPDU: 0
      RecommendSmoothing: 1
```

Create Non-HT Configuration Object

This example shows how to create non-HT configuration objects. It also shows how to change the default property settings by using dot notation or by overriding the default settings by using `Name,Value` pairs when creating the object.

Create Object and Then Modify Properties

Create a non-HT configuration object and view the default settings.

```
nonHT = wlanNonHTConfig

nonHT =
  wlanNonHTConfig with properties:

              Modulation: 'OFDM'
         ChannelBandwidth: 'CBW20'
                     MCS: 0
              PSDULength: 1000
     NumTransmitAntennas: 1
   SignalChannelBandwidth: 0
```

Modify the defaults to specify four transmit antennas and to set the MCS to 3.

```
nonHT.NumTransmitAntennas = 4;
nonHT.MCS = 3

nonHT =
  wlanNonHTConfig with properties:

              Modulation: 'OFDM'
         ChannelBandwidth: 'CBW20'
                     MCS: 3
              PSDULength: 1000
     NumTransmitAntennas: 4
```

```
        SignalChannelBandwidth: 0
```

Create Object and Override Default Property Values

Create a non-HT configuration object. Use a `Name,Value` pair change the modulation scheme to DSSS.

```
nonHT2 = wlanNonHTConfig('Modulation','DSSS')

nonHT2 =
  wlanNonHTConfig with properties:

        Modulation: 'DSSS'
          DataRate: '1Mbps'
      LockedClocks: 1
         PSDULength: 1000
```

For the DSSS modulation scheme, a different set of properties apply and are shown for the non-HT configuration object.

See Also

Objects
wlanDMGConfig | wlanHEMUConfig | wlanHERecoveryConfig | wlanHESUConfig | wlanHTConfig | wlanNonHTConfig | wlanS1GConfig | wlanVHTConfig

Related Examples

- "Waveform Generation" on page 3-22
- "What Is WLAN?" on page 2-2

HE MU Transmission

In this section...

Transmission Mode Options

The options for high-efficiency multi-user (HE MU) transmissions are:

- Orthogonal frequency-division multiple access (OFDMA)
- Full-band multi-user multiple-input/multiple-output (MU-MIMO)
- Mixed OFDMA and MU-MIMO

To choose a transmission mode, you must enable or disable SIGB compression by specifying the state of the SIGB compression bit in the HE-SIG-A field.

- For a 20 MHz transmission, specify the SIGB compression bit directly by setting the `SIGBCompression` property of the `wlanHEMUConfig` object.

  - To enable SIGB compression, set the `SIGBCompression` property to `1` (`true`).
  - To disable SIGB compression, set the `SIGBCompression` property to `0` (`false`).
- For a 40, 80, or 160 MHz transmission, enable or disable SIGB compression by setting the `AllocationIndex` property of the `wlanHEMUConfig` object.

When SIGB compression is enabled, the transmission is full-bandwidth MU-MIMO. The HE-SIG-B field contains no common field, and the resource unit (RU) allocation in the user fields adheres to a standard-specified pattern. Because there is no common field in this case, no allocation index is transmitted. The number of users is determined by decoding the HE-SIG-A field.

When SIGB compression is disabled:

- The transmission is either OFDMA or mixed OFDMA and MU-MIMO, depending on the `AllocationIndex` property of the HE MU configuration object.
- The HE-SIG-B common field includes RU allocation subfields to specify the RU assignment and the number of users per RU for each 20 MHz bandwidth segment.

The "802.11ax Parameterization for Waveform Generation and Simulation" example introduces the concepts associated with HE transmission modes, RU allocation, and parameterization.

The "Recovery Procedure for an 802.11ax Packet" example demonstrates the required steps to detect and decode an HE MU transmission.
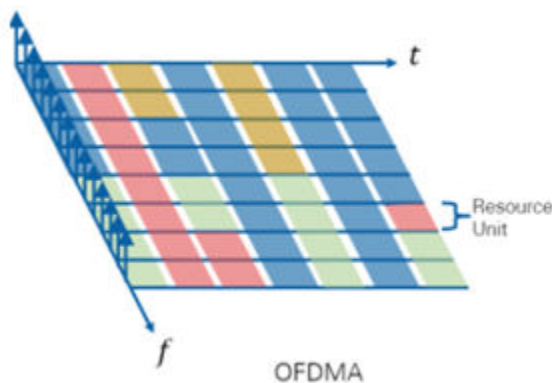
Allocation Index

When creating a `wlanHEMUConfig` object, you must specify the value of the `AllocationIndex` property. Once the object is created, the `AllocationIndex` property is read-only.

The `AllocationIndex` property defines the RU allocation index or a set of RU allocation indices.

- Specify a single allocation index using one integer in either of these forms.

  - An integer in the interval [0, 223]
  - An 8-bit binary sequence specified as a string or character vector

- Specify multiple allocation indices using two, four, or eight integer values in any of these forms.

  - A vector of integers in the interval [0, 223]
  - An 8-bit binary sequence specified as a string array
  - An 8-bit binary sequence specified as a cell array of character vectors

    You can signal punctured 20 MHz or 40 MHz subchannels in an 80 MHz or 160 MHz transmission. To signal a punctured 20 MHz subchannel, set the corresponding element to 113. To signal a punctured 40 MHz subchannel, set the two corresponding adjacent elements to 114. To signal an empty HE-SIG-B user field in an HE-SIG-B content channel, set the corresponding element to 114 or 115.

An RU is a group of 26, 52, 106, 242, 484, 996, or 2×996 subcarriers defining an allocation unit in time and frequency.


The values specified in the `AllocationIndex` property correspond to the 8-bit indices for each 20 MHz subchannel in the first column of Table 27-24 in [1]. The allocation indices define the number of RUs, RU sizes, and number of users assigned to each RU. When SIGB compression is enabled, the number of users is determined by decoding the HE-SIG-A field. When SIGB compression is disabled, the number of users is determined by decoding the HE-SIG-B common field.

When SIGB compression is enabled, the HE-SIG-B field contains only the user field.

When SIGB compression is disabled, the HE-SIG-B field includes both the common and user fields. The common field carries the RU Allocation subfields in one or two content channels. Depending on the PPDU bandwidth, the common field can contain multiple RU Allocation subfields. For a discussion of the frequency-domain mapping of channel contents into the common field, see section 27.3.10.8.3 of [1].

This figure shows the structure of the HE-SIG-B field when SIGB compression is disabled.

HE MU Format PPDU
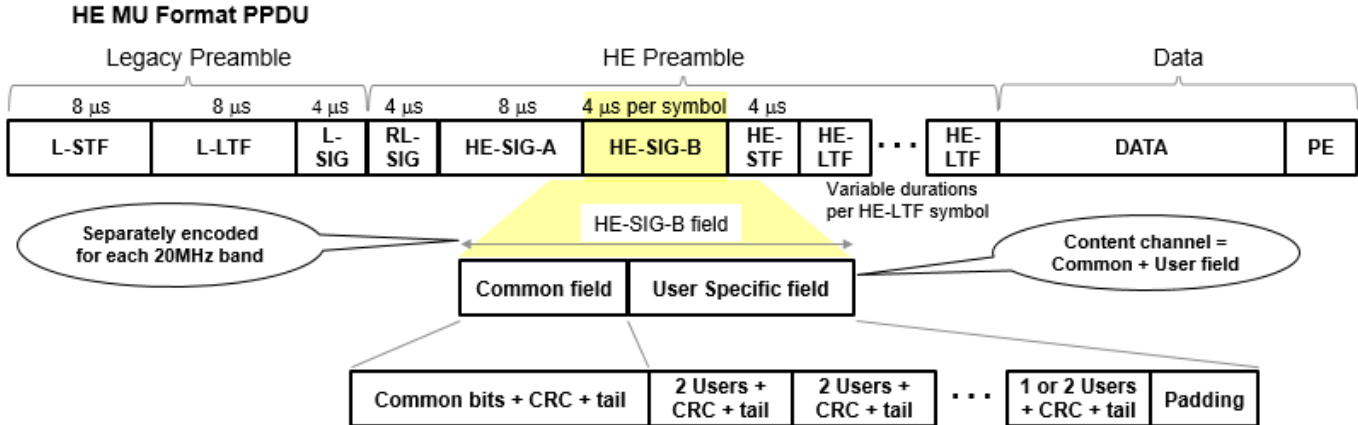
The format of the common field is defined in Table 27-23 of [1]. The RU Allocation subfield in the common field of HE-SIG-B consists of 8 bits that indicate this information for each 20 MHz PPDU bandwidth.

- RU assignment in the frequency domain, which determines the size of the RUs and their placement in the frequency domain.

- Number of user fields in a 20 MHz band within the HE-SIG-B content channel, which determines the number of users multiplexed in the RUs. For RUs of size greater than or equal to 106 tones, which support MU-MIMO, the RU Allocation subfield indicates the number of users multiplexed using MU-MIMO. The HE-SIG-B field consists of N RU Allocation subfields, where:

  - $N = 1$ for 20 MHz and 40 MHz HE MU PPDUs
  - $N = 2$ for 80 MHz HE MU PPDUs
  - $N = 4$ for 160 MHz and 80+80 MHz HE MU PPDUs

This table lists the allocation indices and corresponding RU assignments for 20 MHz subchannels and RUs with at most 242 tones. The table shows the number of tones per RU and the number of users assigned for each allocation index.

Allocation Index	20-MHz Subchannel Resource Unit (RU) Assignment
0	26 · 26 · 26 · 26 · 26 · 26 · 26 · 26 · 26
1	26 · 26 · 26 · 26 · 26 · 26 · 26 · 52
2	26 · 26 · 26 · 26 · 26 · 52 · 26 · 26
3	26 · 26 · 26 · 26 · 26 · 52 · 52
4	26 · 26 · 52 · 26 · 26 · 26 · 26 · 26
5	26 · 26 · 52 · 26 · 26 · 26 · 52
6	26 · 26 · 52 · 26 · 52 · 26 · 26
7	26 · 26 · 52 · 26 · 52 · 52
8	52 · 26 · 26 · 26 · 26 · 26 · 26 · 26
9	52 · 26 · 26 · 26 · 26 · 26 · 52
10	52 · 26 · 26 · 26 · 52 · 26 · 26
11	52 · 26 · 26 · 26 · 52 · 52
12	52 · 52 · 26 · 26 · 26 · 26 · 26
13	52 · 52 · 26 · 26 · 26 · 52
14	52 · 52 · 26 · 52 · 26 · 26
15	52 · 52 · 26 · 52 · 52
16-23 (15 + NumUsers)	52 · 52 · - · 106 (1-8 users)
24-31 (23 + NumUsers)	106 (1-8 users) · - · 52 · 52
32-39 (31 + NumUsers)	26 · 26 · 26 · 26 · 26 · 106 (1-8 users)
40-47 (39 + NumUsers)	26 · 26 · 52 · 26 · 106 (1-8 users)
48-55 (47 + NumUsers)	52 · 26 · 26 · 26 · 106 (1-8 users)
56-63 (55 + NumUsers)	52 · 52 · 26 · 106 (1-8 users)
64-71 (63 + NumUsers)	106 (1-8 users) · 26 · 26 · 26 · 26 · 26
72-79 (71 + NumUsers)	106 (1-8 users) · 26 · 26 · 26 · 52
80-87 (79+ NumUsers)	106 (1-8 users) · 26 · 52 · 26 · 26
88-95 (87 + NumUsers)	106 (1-8 users) · 26 · 52 · 52
96-103 (95 + NumUsers)	106 · - · 106 (1-8 users)
104-112 (103 + NumUsers)	106 (1-8 users) · - · 106
112	52 · 52 · 52 · 52
113	Empty 242-tone RU - No user assigned
116-127	Reserved
128-135 (127 + NumUsers)	106 · 26 · 106 (1-8 users)
136-143 (135 + NumUsers)	106 (2 users) · 26 · 106 (1-8 users)
144-151 (143 + NumUsers)	106 (3 users) · 26 · 106 (1-8 users)
152-159 (151 + NumUsers)	106 (4 users) · 26 · 106 (1-8 users)
160-167 (159 + NumUsers)	106 (5 users) · 26 · 106 (1-8 users)
168-175 (167 + NumUsers)	106 (6 users) · 26 · 106 (1-8 users)
176-183 (175 + NumUsers)	106 (7 users) · 26 · 106 (1-8 users)
184-191 (183 + NumUsers)	106 (8 users) · 26 · 106 (1-8 users)
192-199 (191 + NumUsers)	242 (1-8 users)

Legend:
- RU assigned to 1 user
- RU assigned to 1-8 users, depending on the allocation index
- RU assigned to specified number of users, irrespective of the allocation index

Callouts:
- 26-tone RU assigned to one user as part of a 20-MHz subchannel assignment of nine 26-tone RUs
- No users assigned to this RU; no data field transmitted on these subcarriers
- The number of users assigned to this 106-tone RU depends on the allocation index
- If selected, this 20-MHz subchannel is unused; the subchannel is punctured
- The number of users assigned to the upper 106-tone RU depends on the allocation index, but two users are always assigned to the lower 106-tone RU
- From 2 to 8 users assigned to the lower 106-tone RU depending on the allocation index

This table lists the allocation indices and corresponding RU assignments for subchannels greater than 20 MHz and RUs of more than 242 tones.

Allocation Index	RU Allocation & Number of Users on the Corresponding HE-SIG-B Content Channel for RU Size > 242
114	484-tone RU with no users signaled on the corresponding HE-SIG-B content channel
115	996-tone RU with no users signaled on the corresponding HE-SIG-B content channel
200-207 (199 + NumUsers)	Full band 40 MHz (1-8 users), or 484-tone RU with 1-8 users signaled in the corresponding HE-SIG-B content channel
208-215 (207 + NumUsers)	Full band 80 MHz (1-8 users), or 996-tone RU with 1-8 users signaled in the corresponding HE-SIG-B content channel
216-223 (215 + NumUsers)	Full band 160 MHz (1-8 users)
224-255	Reserved

Legend:
- >242-tone RU allocation

Callouts:
- Must be used with other allocation indices. Signifies a 484-tone RU with zero users signalled on the corresponding HE-SIG-B content channel
- A single allocation index between 200-207 configures a full-band 40 MHz 484-tone RU with 1-8 users

The format of the user field for non-MU-MIMO and MU-MIMO allocations are defined in Tables 27-26 and 27-27, of [1], respectively.

This table shows allocation index options required to specify transmission type for all channel bandwidths.

Transmission Type	20 MHz Transmission	40 MHz Transmission	80 MHz Transmission	160 MHz Transmission
Full-bandwidth MU-MIMO	`wlanHEMUConfig(a,'SIGBCompression', 1)` Specify `a` as an integer in the interval [192, 199].	`wlanHEMUConfig(a)` Specify `a` as an integer in the interval [200, 207].	`wlanHEMUConfig(a)` Specify `a` as an integer in the interval [208, 215].	`wlanHEMUConfig(a)` Specify `a` as an integer in the interval [216, 223].
		The `wlanHEMUConfig` object sets the `SIGBCompression` property to `1` (true), and splits users between the two content channels.		

Full-bandwidth MU-MIMO	`wlanHEMUConfig(a,'SIGBCompression',0)`
	
	 Specify `a` as an integer in the interval [192, 199].
		`wlanHEMUConfig([a 114])`
		
		 Specify `a` as an integer in the interval [200, 207].
			`wlanHEMUConfig([a 115 115 115])`
			
			 `wlanHEMUConfig([115 115 a 115])`
			
			 `wlanHEMUConfig([a 115 b 115])`
			
			 Specify `a` and `b` as integers in the interval [208, 215].
				`wlanHEMUConfig([a 115 115 115 115 115 115 115])`
				
				 `wlanHEMUConfig([a 115 b 115 c 115 d 115])`
				
				 `wlanHEMUConfig([115 115 115 115 a 115 b 115])`
				
				 `wlanHEMUConfig([115 115 115 115 115 115 a 115])`
				
				 `wlanHEMUConfig([115 115 a 115 115 115 b 115])`
				
				 Specify `a`, `b`, `c`, and `d` as integers in the interval [216, 223].

				The `wlanHEMUConfig` object sets the `SIGBCompression` property to `0` (false). All users are in content channel 1.

Transmission Type	20 MHz Transmission	40 MHz Transmission	80 MHz Transmission	160 MHz Transmission

Full-bandwidth MU-MIMO	`wlanHEMUConfig(a,'SIGBCompression',0)`
	
	Specify a as an integer in the interval [192, 199].
		`wlanHEMUConfig([114 a])`
		
		Specify a as an integer in the interval [200, 207].
			`wlanHEMUConfig([115 a 115 115])`
			
			`wlanHEMUConfig([115 115 115 a])`
			
			`wlanHEMUConfig([115 a 115 b])`
			
			Specify a and b as integers in the interval [208, 215].
				`wlanHEMUConfig([115 a 115 115 115 115 115 115])`
				
				`wlanHEMUConfig([115 a 115 b 115 c 115 d])`
				
				`wlanHEMUConfig([115 115 115 115 115 a 115 b])`
				
				`wlanHEMUConfig([115 115 115 115 115 115 115 a])`
				
				`wlanHEMUConfig([115 115 115 a 115 115 115 b])`
				
				Specify a, b, c, and d as integers in the interval [216, 223].

		The `wlanHEMUConfig` object sets the `SIGBCompression` property to `0` (false). All users are in content channel 2.		
Full bandwidth MU-MIMO	`wlanHEMUConfig(a,'SIGBCompression',0)` Specify a as an integer in the interval [192, 199].	`wlanHEMUConfig([a b])` Specify a and b as integers in the interval [200, 207].	`wlanHEMUConfig([a b c d])` Specify a, b, c, and d as integers in the interval [208, 215].	`wlanHEMUConfig([a b c d e f g h])` Specify a, b, c, d, e, f, g, and h as integers in the interval [216, 223].
		The `wlanHEMUConfig` object sets the `SIGBCompression` property to `0` (false). Users are in their respective content channels. For example, in the 80 MHz transmission case, the users represented by allocation indices a and c are in content channel 1, and the users represented by allocation indices b and d are in content channel 2.		

Transmission Type	20 MHz Transmission	40 MHz Transmission	80 MHz Transmission	160 MHz Transmission
Mixed OFDMA and MU-MIMO	`wlanHEMUConfig(a,'SIGBCompression',0)` Specify a as an integer in the interval [0, 191].	`wlanHEMUConfig([a b])` Specify a and b as integers in the interval [0, 199].	`wlanHEMUConfig([a b c d])` Specify a, b, c, and d as integers in the interval [0, 207].	`wlanHEMUConfig([a b c d e f g h])` Specify a, b, c, d, e, f, g, and h as integers in the interval [0, 215].
	20 MHz transmissions have only one content channel.	The `wlanHEMUConfig` object sets the `SIGBCompression` property to `0` (false). Users are in their respective content channels. For example, in the 80 MHz transmission case, the users represented by allocation indices a and c are in content channel 1, and the users represented by allocation indices b and d are in content channel 2.		

References

[1] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

See Also

More About
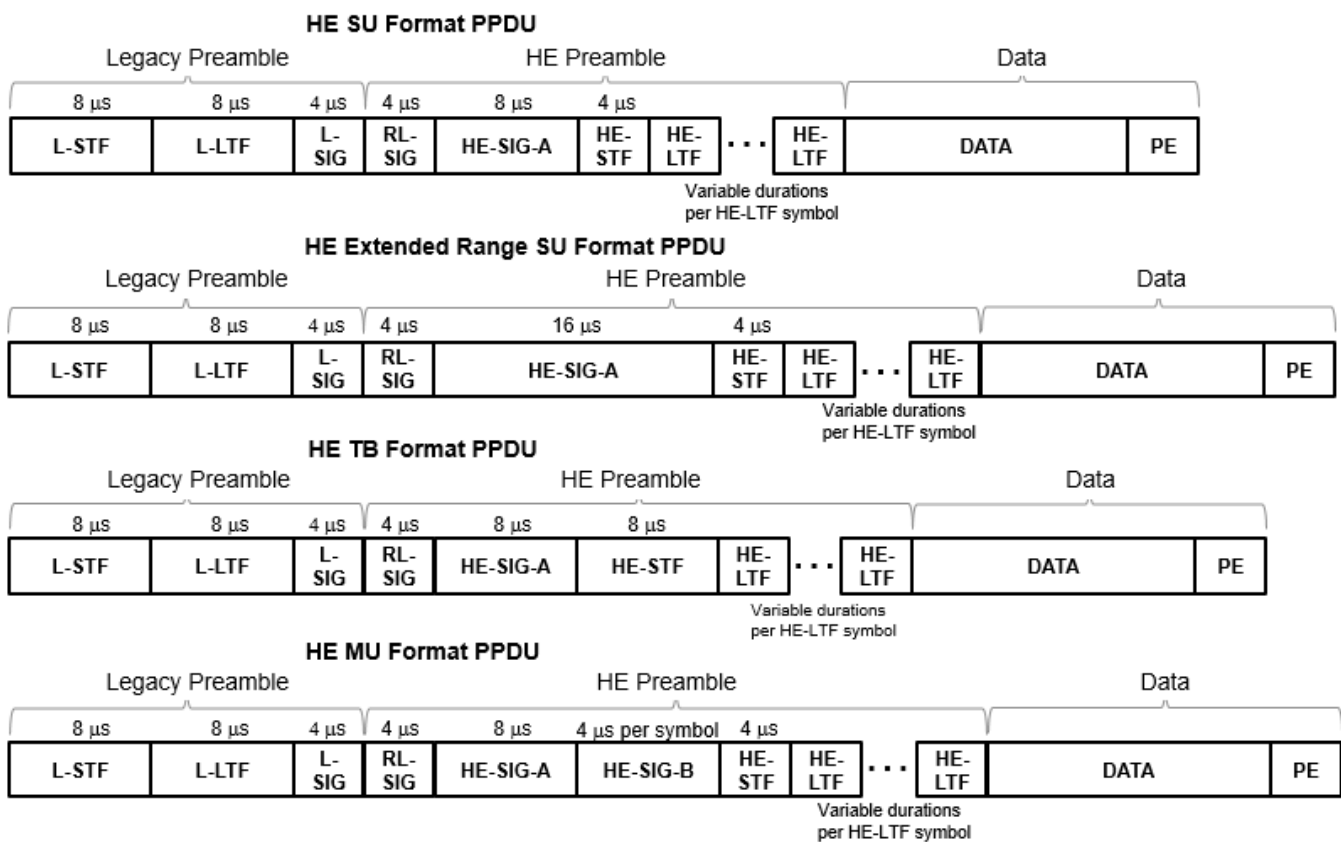
- "WLAN PPDU Structure" on page 2-9

Waveform Generation

After you create the necessary configuration objects described in "Create Configuration Objects" on page 3-3, you can use the objects to generate the desired WLAN format waveform.

The IEEE 802.11[2] standards define a physical layer protocol data unit (PPDU) as the transmission unit at the physical layer. For a detailed description of the PPDU field structures for each transmission format, see "WLAN PPDU Structure" on page 2-9.
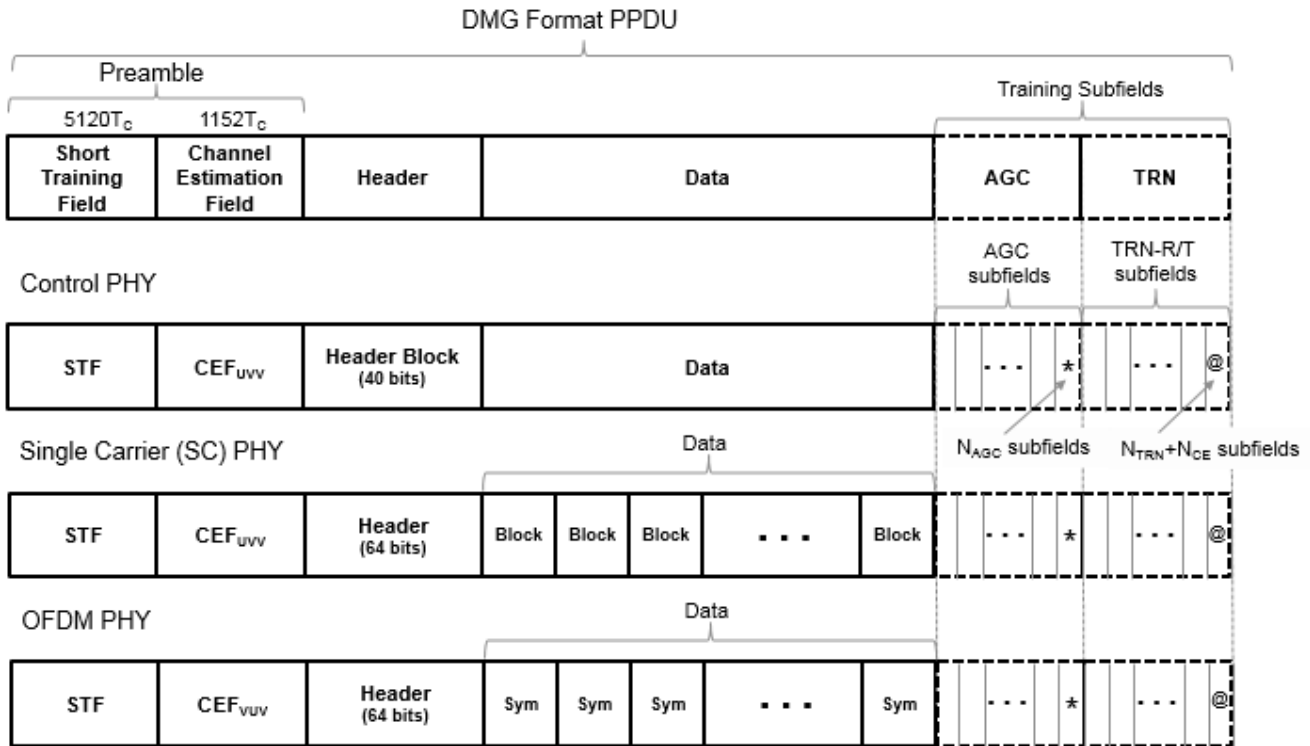
HE Format

In HE, there are four transmission modes supported: single user, single user extended range, trigger-based, and multi-user.

HE SU Format PPDU

Legacy Preamble			HE Preamble					Data	
8 µs	8 µs	4 µs	4 µs	8 µs	4 µs				
L-STF	L-LTF	L-SIG	RL-SIG	HE-SIG-A	HE-STF	HE-LTF	••• HE-LTF	DATA	PE

Variable durations per HE-LTF symbol

HE Extended Range SU Format PPDU

Legacy Preamble			HE Preamble					Data	
8 µs	8 µs	4 µs	4 µs	16 µs	4 µs				
L-STF	L-LTF	L-SIG	RL-SIG	HE-SIG-A	HE-STF	HE-LTF	••• HE-LTF	DATA	PE

Variable durations per HE-LTF symbol

HE TB Format PPDU

Legacy Preamble			HE Preamble				Data	
8 µs	8 µs	4 µs	4 µs	8 µs	8 µs			
L-STF	L-LTF	L-SIG	RL-SIG	HE-SIG-A	HE-STF	HE-LTF	••• HE-LTF	DATA	PE

Variable durations per HE-LTF symbol

HE MU Format PPDU

Legacy Preamble			HE Preamble					Data	
8 µs	8 µs	4 µs	4 µs	8 µs	4 µs per symbol	4 µs			
L-STF	L-LTF	L-SIG	RL-SIG	HE-SIG-A	HE-SIG-B	HE-STF	HE-LTF	••• HE-LTF	DATA	PE

Variable durations per HE-LTF symbol

DMG PPDU

In DMG, there are three physical layer (PHY) modulation schemes supported: control, single carrier, and OFDM.

---

2.   IEEE Std 802.11-2016 Adapted and reprinted with permission from IEEE. Copyright IEEE 2016. All rights reserved.

DMG Format PPDU

S1G Format

In S1G, there are three transmission modes: S1G_LONG, S1G_SHORT, and S1G_1M. Each transmission mode has a specific PPDU preamble structure.
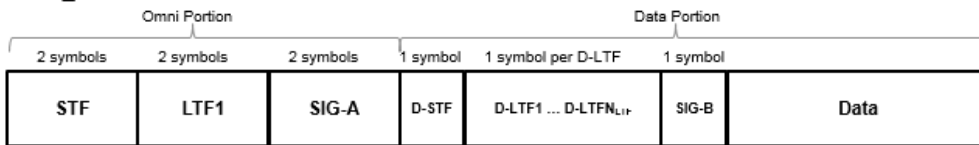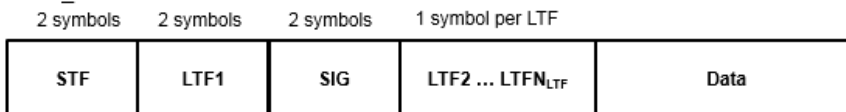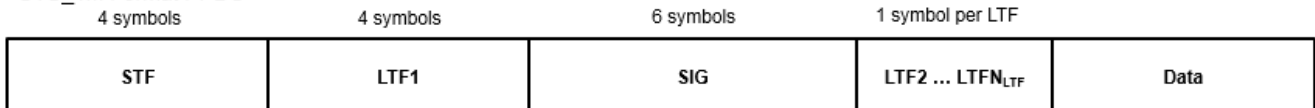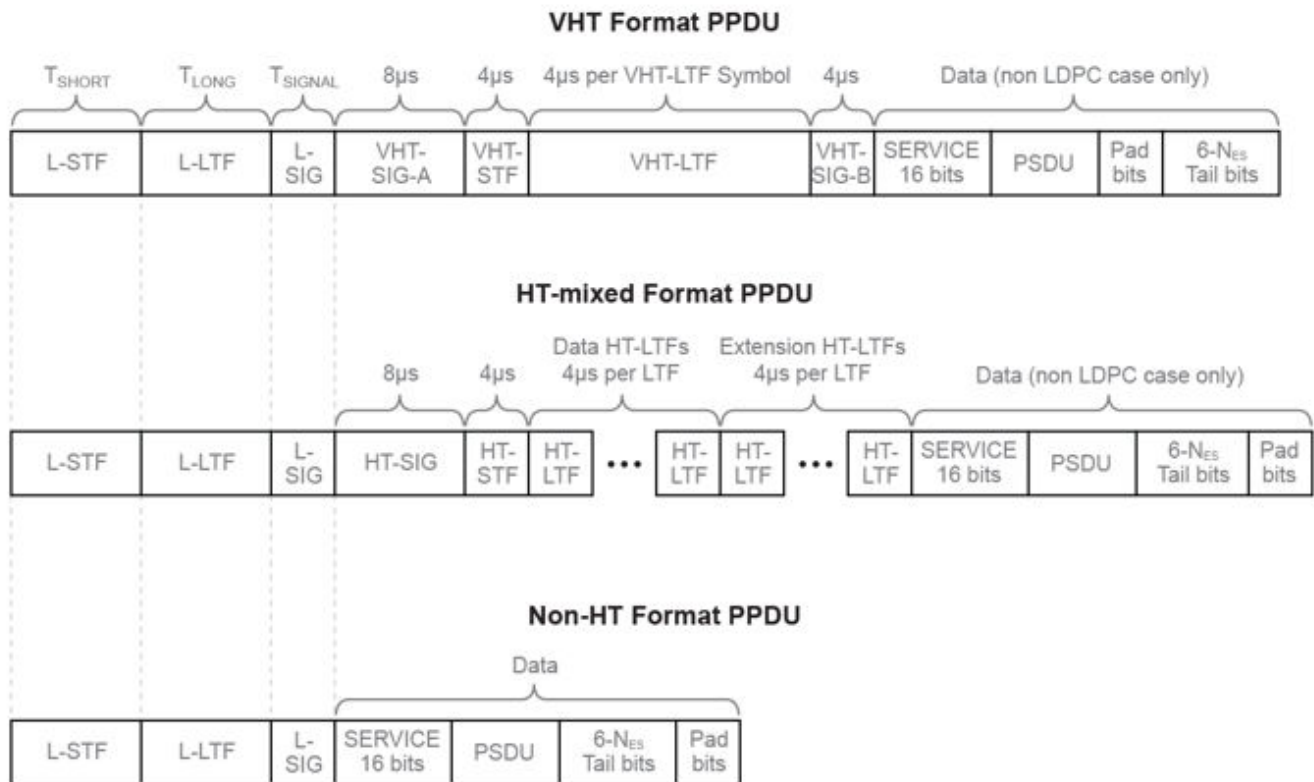

VHT, HT, and non-HT Formats

The VHT, HT, and non-HT PPDU formats consist of preamble and data fields.

VHT Format PPDU

HT-mixed Format PPDU

Non-HT Format PPDU

Use WLAN Toolbox functions to generate a full PPDU waveform or individual PPDU field waveforms.

Generate a full PPDU waveform by using the `wlanWaveformGenerator` function to populate all PPDU fields (preamble and data) in a single call. The `wlanWaveformGenerator` function accepts a bit stream, a format configuration object, and `Name,Value` pairs to configure the waveform.

Generate WLAN Waveforms

Generate HE, DMG, S1G, VHT, HT-mixed, and non-HT format waveforms. Vary configuration parameters and plot the waveforms to highlight differences in waveforms and sample rates.

In each section of this example, you:

- Create a format-specific configuration object.
- Create a vector of information bits for the packet data payload. Internally, the `wlanWaveformGeneration` function loops through the bits vector as many times as needed to generate the specified number of packets.
- Generate the format-specific waveform and plot it. For plotting, because no filtering is applied to the waveform and the oversampling rate is 1, set the sampling rate equal to the channel bandwidth.

Generate Single User HE Format Waveform

Create an HE single-user (HE SU) configuration object and waveform. Using `Name,Value` pairs, specify 4 packets and 15 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgHESU = wlanHESUConfig;
bits = [1;0;0;1;1];
hesuWaveform = wlanWaveformGenerator(bits,cfgHESU, ...
    'NumPackets',4,'IdleTime',15e-6);
```
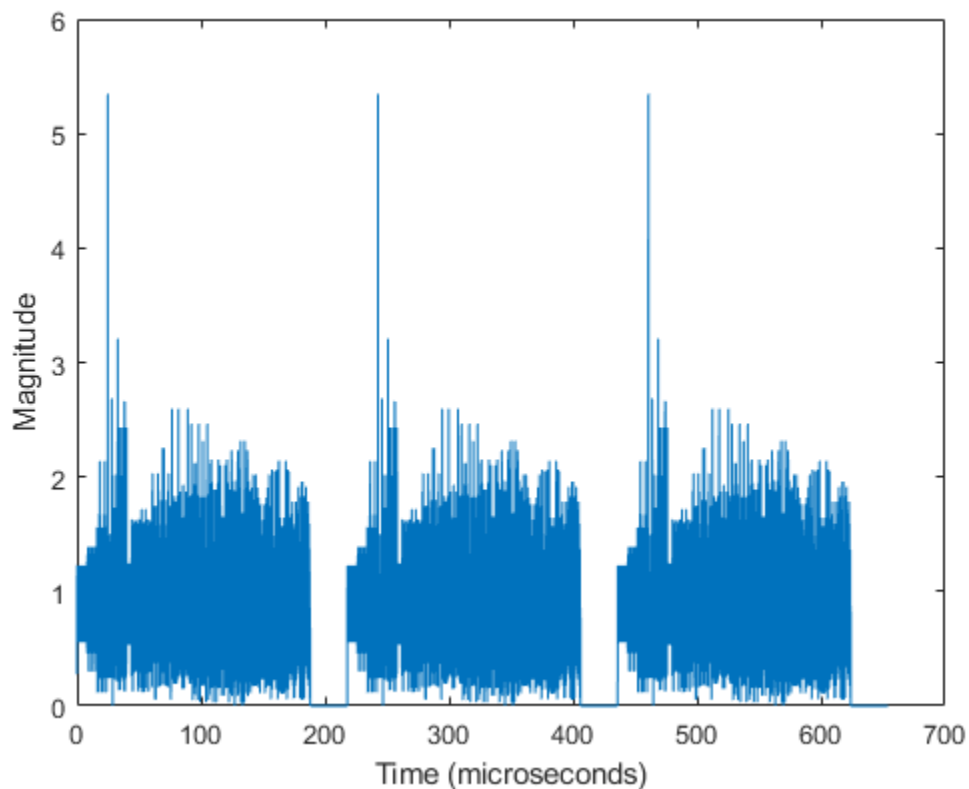
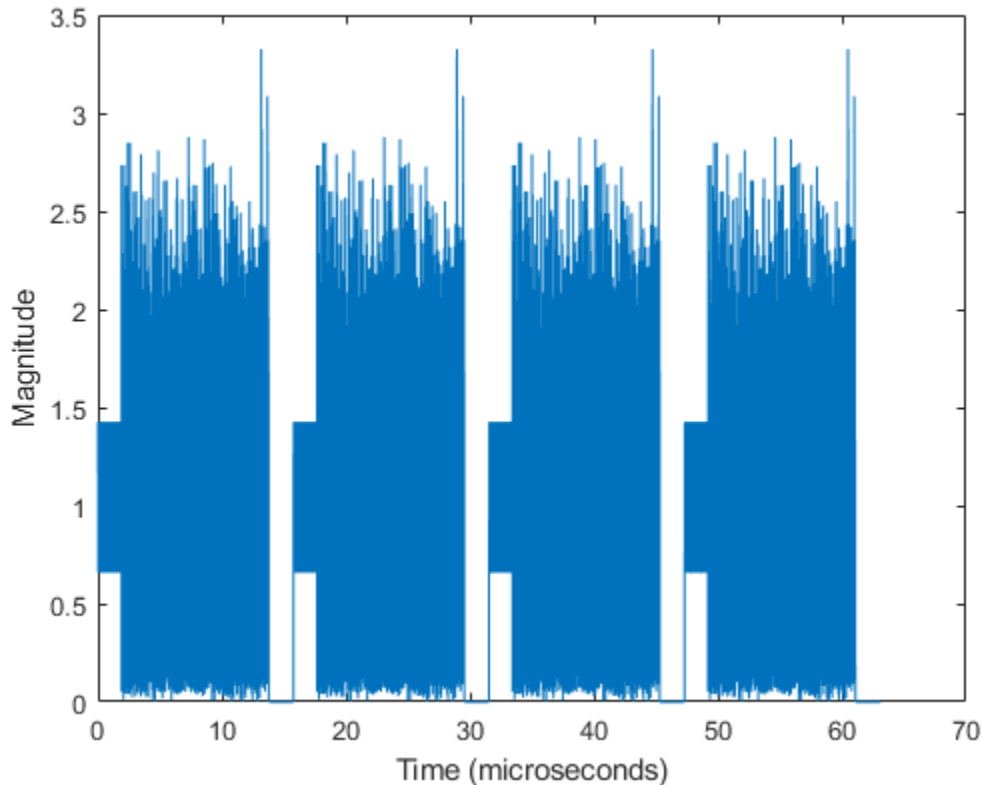Plot the single user HE format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 20e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(hesuWaveform)-1)/fs)*1e6;
plot(time,abs(hesuWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```
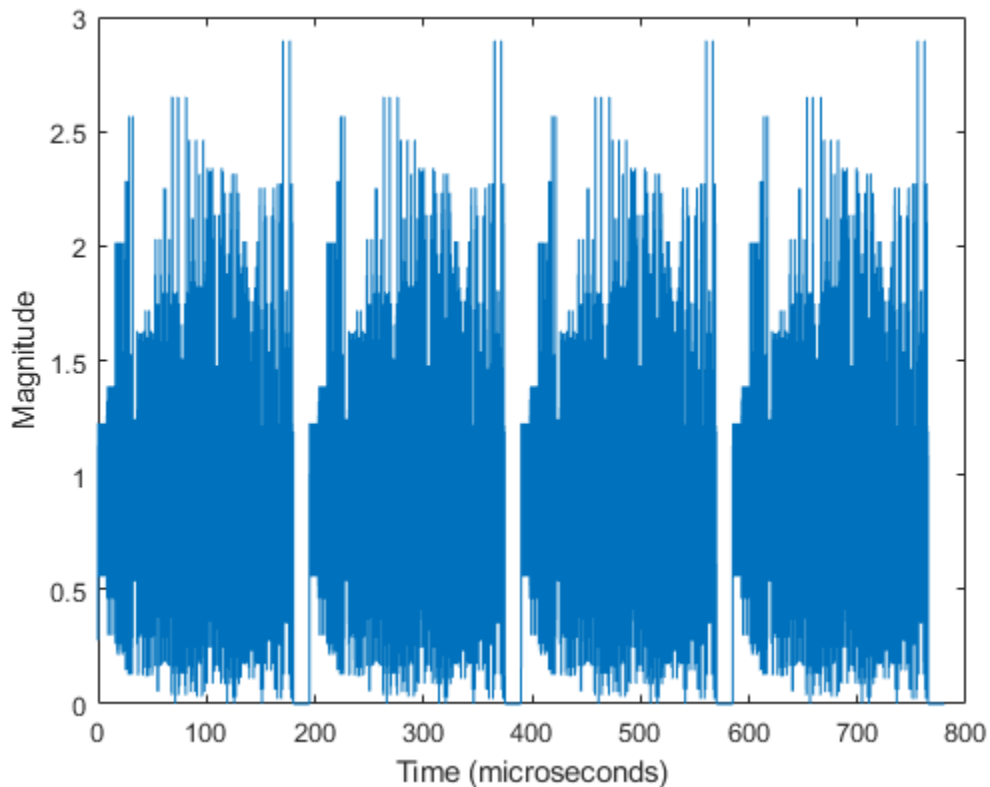

The plot shows four single user HE format packets, with each packet separated by 15 microseconds of idle time.

Generate Multiuser HE Format Waveform

Create an HE multiuser (HE MU) configuration object and waveform. Using `Name,Value` pairs, specify 3 packets and 30 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgHEMU = wlanHEMUConfig(192);
bits = [1;0;0;1;1];
hemuWaveform = wlanWaveformGenerator(bits,cfgHEMU, ...
    'NumPackets',3,'IdleTime',30e-6);
```

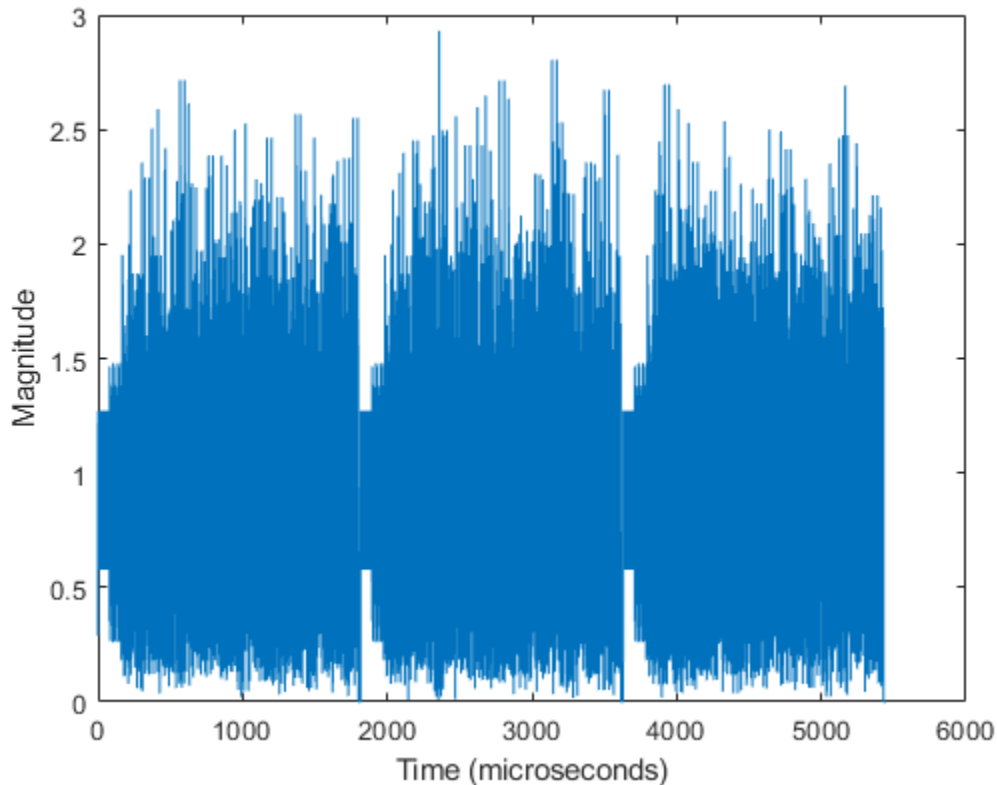Plot the multiuser HE format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 20e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(hemuWaveform)-1)/fs)*1e6;
plot(time,abs(hemuWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```


The plot shows three multiuser HE format packets, with each packet separated by 30 microseconds of idle time.

Generate DMG Format Waveform

Create a DMG configuration object and waveform. Using Name,Value pairs, assign 13 for the MCS which specifies an OFDM waveform, 4 packets, and 2 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgDMG = wlanDMGConfig('MCS',13);
bits = [1;0;0;1;1];
dmgWaveform = wlanWaveformGenerator(bits,cfgDMG, ...
    'NumPackets',4,'IdleTime',2e-6);
```

Plot the DMG format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 2640e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(dmgWaveform)-1)/fs)*1e6;
plot(time,abs(dmgWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```


The plot shows four DMG format packets, with each packet separated by 2 microseconds of idle time.

Generate S1G Format Waveform

Create a sub-1-GHz (S1G) configuration object and waveform. Using `Name,Value` pairs, specify 4 MHz channel bandwidth, 3 packets, and 15 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgS1G = wlanS1GConfig('ChannelBandwidth','CBW4');
bits = [1;0;0;1;1];

s1gWaveform = wlanWaveformGenerator(bits,cfgS1G, ...
    'NumPackets',3,'IdleTime',15e-6);
```

Plot the S1G format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 4e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(s1gWaveform)-1)/fs)*1e6;
plot(time,abs(s1gWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```

The plot shows three S1G format packets, with each packet separated by 15 microseconds of idle time.
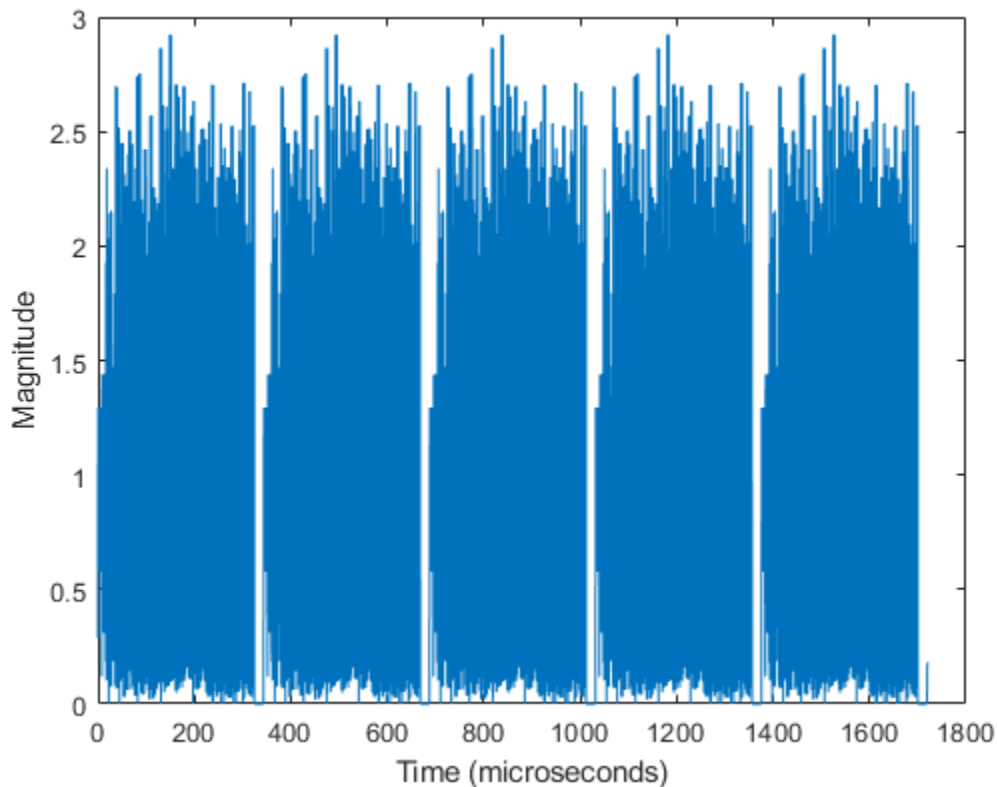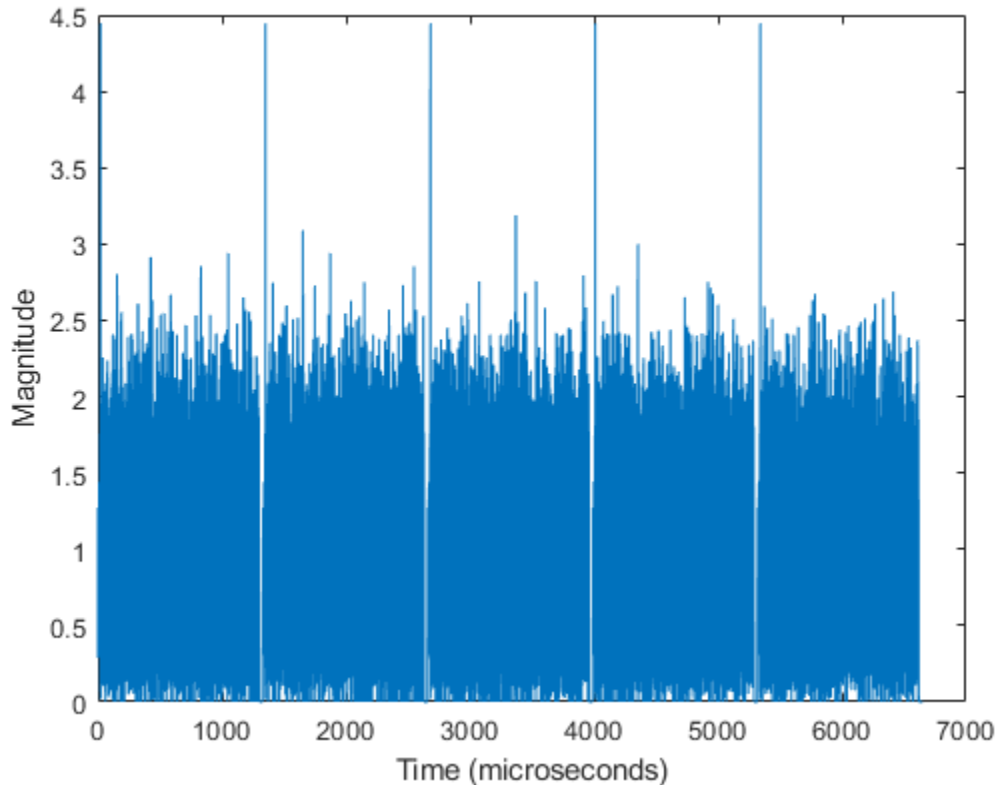
Generate VHT Format Waveform

Create a VHT configuration object and waveform. Using `Name,Value` pairs, specify 5 packets and 20 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgVHT = wlanVHTConfig;
bits = [1;0;0;1;1];
vhtWaveform = wlanWaveformGenerator(bits,cfgVHT, ...
    'NumPackets',5,'IdleTime',20e-6);
```

Plot the VHT format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 80e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(vhtWaveform)-1)/fs)*1e6;
plot(time,abs(vhtWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```

The plot shows five VHT format packets, with each packet separated by 20 microseconds of idle time.

Generate HT Format Waveform

Create an HT configuration object and waveform. Using `Name,Value` pairs, specify 5 packets and 30 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgHT = wlanHTConfig;
bits = [1;0;0;1;1];
htWaveform = wlanWaveformGenerator(bits,cfgHT, ...
    'NumPackets',5,'IdleTime',30e-6);
```

Plot the HT format waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 20e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(htWaveform)-1)/fs)*1e6;
plot(time,abs(htWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```

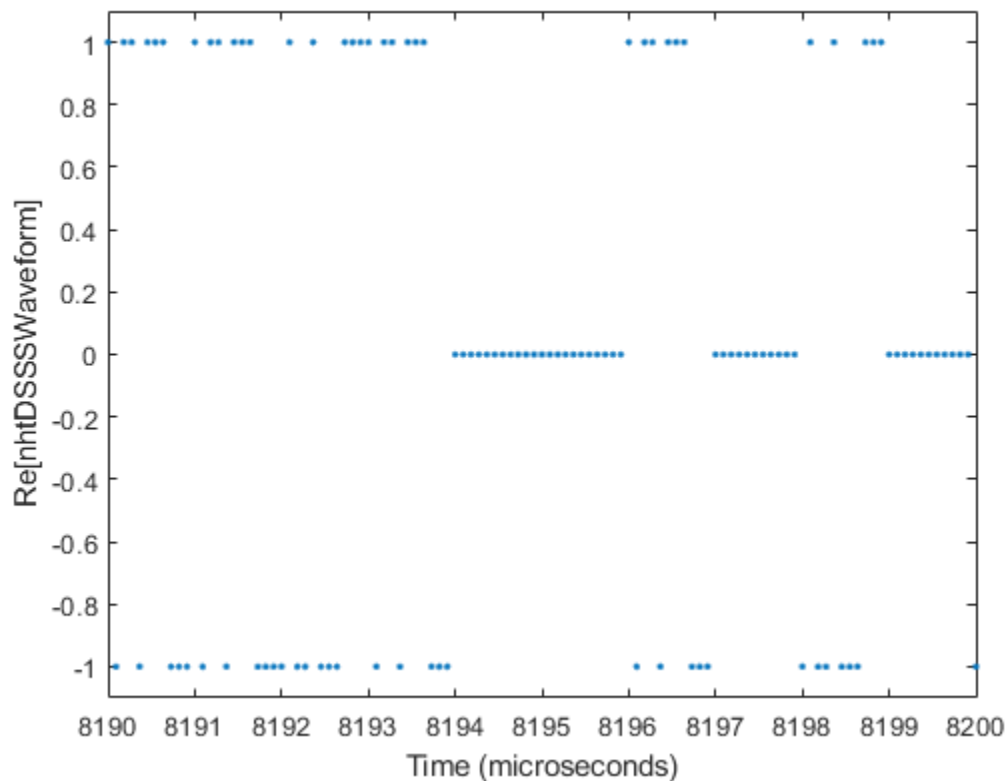The plot shows five HT format packets, with 30 microseconds of idle time separating each packet.

Generate Non-HT Format DSSS Waveform

Create a non-HT configuration object and generate a non-HT format DSSS waveform with a 2 Mbps data rate. Using `Name,Value` pairs, specify 2 packets and 5 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgNonHT = wlanNonHTConfig('Modulation','DSSS','DataRate','2Mbps');
bits = [1;0;0;1;1];
nhtDSSSWaveform = wlanWaveformGenerator(bits,cfgNonHT, ...
    'NumPackets',2,'IdleTime',5e-6);
```

Plot the non-HT Format DSSS waveform, scaling the *x-axis* relative to the channel bandwidth. As specified in IEEE 802.11-2012, Section 17.1.1, the channel bandwidth is 11 MHz for DSSS.

```
fs = 11e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(nhtDSSSWaveform)-1)/fs)*1e6;
plot(time,real(nhtDSSSWaveform),'.')
xlabel ('Time (microseconds)');
ylabel('Re[nhtDSSSWaveform]');
axis([8190,8200,-1.1,1.1])
```

Sample values in DSSS modulation are –1 or 1. The plot shows the real values for a section of the waveform that includes the tail end of the first packet, the 5 microsecond idle period, and the beginning of the second packet for the non-HT format DSSS modulated waveform.
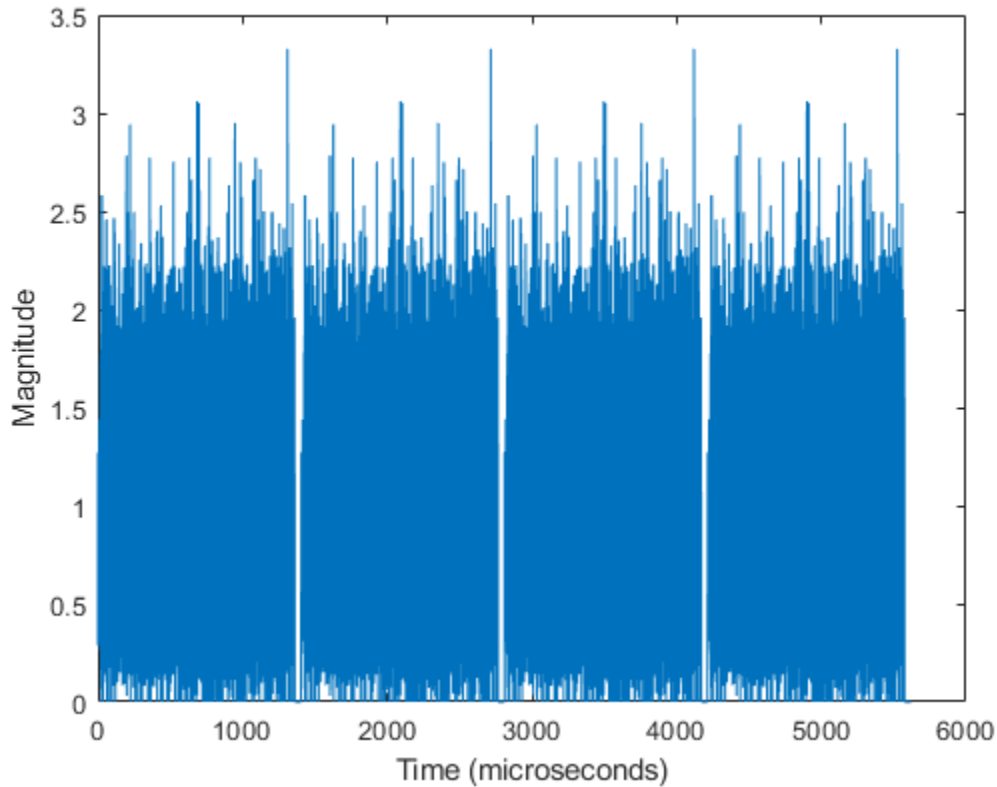
Generate Non-HT Format OFDM Waveform

Create a non-HT configuration object and waveform. Using `Name,Value` pairs, specify 4 packets and 45 microseconds of idle time. Display the configuration object and inspect its properties and settings.

```
cfgNonHT = wlanNonHTConfig;
bits = [1;0;0;1;1];
nhtWaveform = wlanWaveformGenerator(bits,cfgNonHT, ...
    'NumPackets',4,'IdleTime',45e-6);
```

Plot the non-HT format OFDM waveform, scaling the *x-axis* relative to the channel bandwidth.

```
fs = 20e6; % Set sampling frequency equal to the channel bandwidth
time = ((0:length(nhtWaveform)-1)/fs)*1e6;
plot(time,abs(nhtWaveform))
xlabel ('Time (microseconds)');
ylabel('Magnitude');
```

The plot shows four non-HT format OFDM modulated packets, with 45 microseconds of idle time separating each packet.

Waveforms of Individual PPDU Fields

You can also create a VHT, HT, or non-HT PPDU waveform by generating and concatenating waveforms for individual PPDU fields.

PPDU Format	Individual Field Functions
VHT	wlanLSTF, wlanLLTF, wlanLSIG, wlanVHTSTF, wlanVHTLTF, wlanVHTSIGA, wlanVHTSIGB, and wlanVHTData
HT	wlanLSTF, wlanLLTF, wlanLSIG, wlanHTSTF, wlanHTLTF, wlanHTSIG, and wlanHTData
Non-HT for OFDM modulation	wlanLSTF, wlanLLTF, wlanLSIG, and wlanNonHTData

Generating individual PPDU field waveforms, enables you to experiment with the individual fields without generating an entire PPDU.

See Also

wlanHTConfig | wlanNonHTConfig | wlanVHTConfig

More About

- "Create Configuration Objects" on page 3-3
- "WLAN Channel Models" on page 3-41
- "What Is WLAN?" on page 2-2

App-Based WLAN Waveform Generation

This example shows how to generate IEEE® 802.11™ waveforms by using the **WLAN Waveform Generator** app.

Open WLAN Waveform Generator App

On the **Apps** tab of the MATLAB® toolstrip, select the **WLAN Waveform Generator** app icon under **Signal Processing and Communications**. This selection opens the **Wireless Waveform Generator** app configured for WLAN waveform generation.

Select IEEE 802.11 PHY Format

Choose the PHY format of the waveform you want to generate by selecting one of the formats under **WLAN (IEEE 802.11)** in the **Waveform Type** section of the app toolstrip. The app supports these IEEE 802.11 PHY formats.

- `802.11ax`
- `802.11ah`
- `802.11ad`
- `802.11n/ac`
- `802.11p`
- `802.11b/g`
- `802.11a/g/j`

Generate WLAN Waveforms

Set transmission and configuration parameters by specifying options in the **Waveform** tab on the left pane of the app. Add impairments and select visualization tools by specifying options in the **Generation** section of the app toolstrip. To visualize the waveform, click **Generate**. You can export the generated waveform and its parameters by clicking **Export**. You can export the waveform to:

- A MATLAB script with a `.m` extension
- a file with a `.bb` or `.mat` extension
- Your MATLAB workspace as a structure

Generate HE TB Waveform with Default Settings

This image shows the **Time Scope** and **Spectrum Analyzer** visualization results for a high-efficiency trigger-based (HE TB) waveform. The waveform comprises a single packet. The RU size is 52 subcarriers, and the RU index is 3. All other transmission and configuration parameters take their default values.
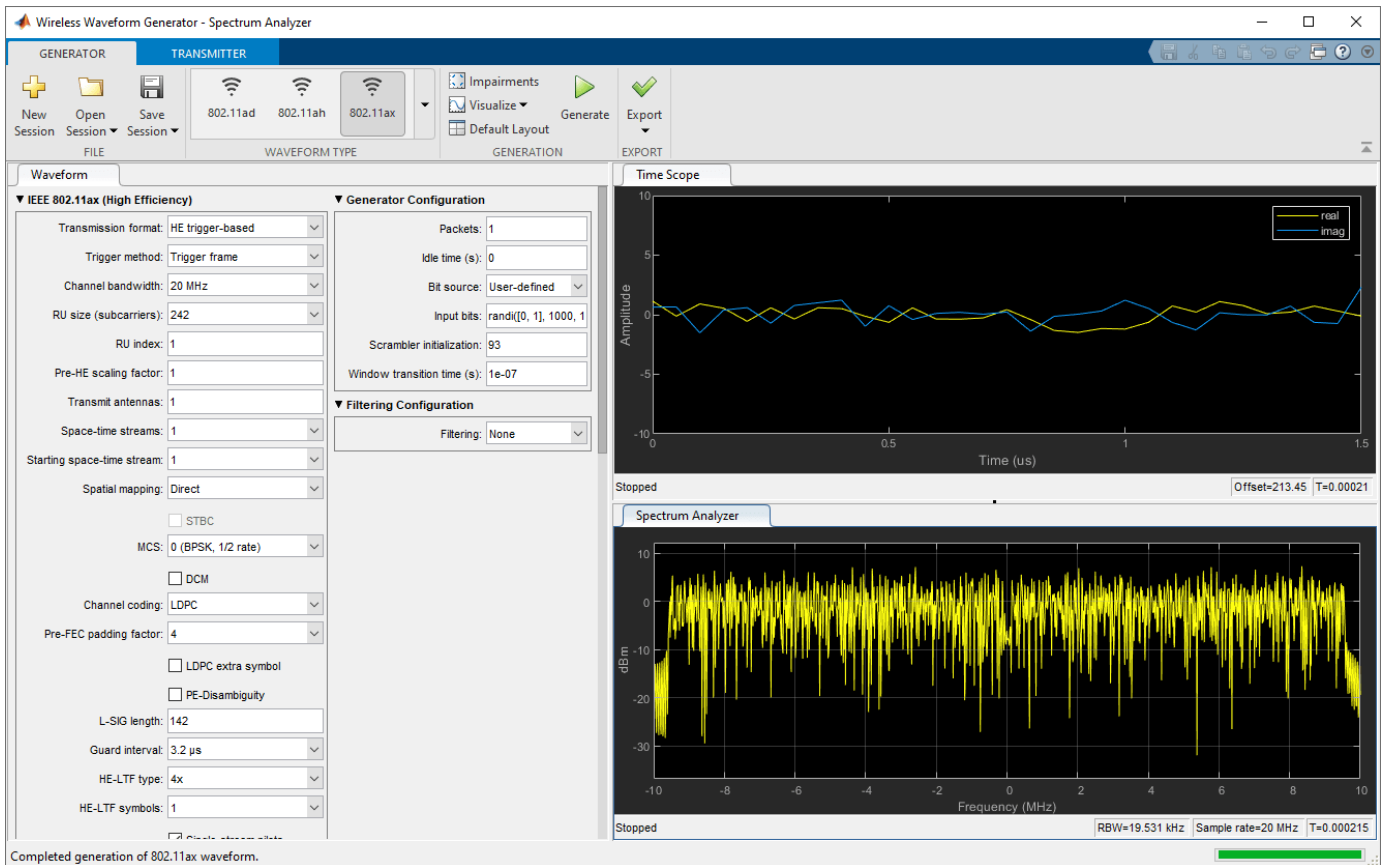
Generate HE ER SU Waveform with Packet Extension

This image shows the **Time Scope** and **Spectrum Analyzer** visualization results for a high-efficiency extended-range single-user (HE ER SU) waveform. The waveform comprises two packets separated by an idle time of 50 microseconds, and contains a nominal packet padding of 8 microseconds for packet extension. The transmission of this waveform uses two antennas and Hadamard spatial mapping. All other format and configuration parameters take their default values.
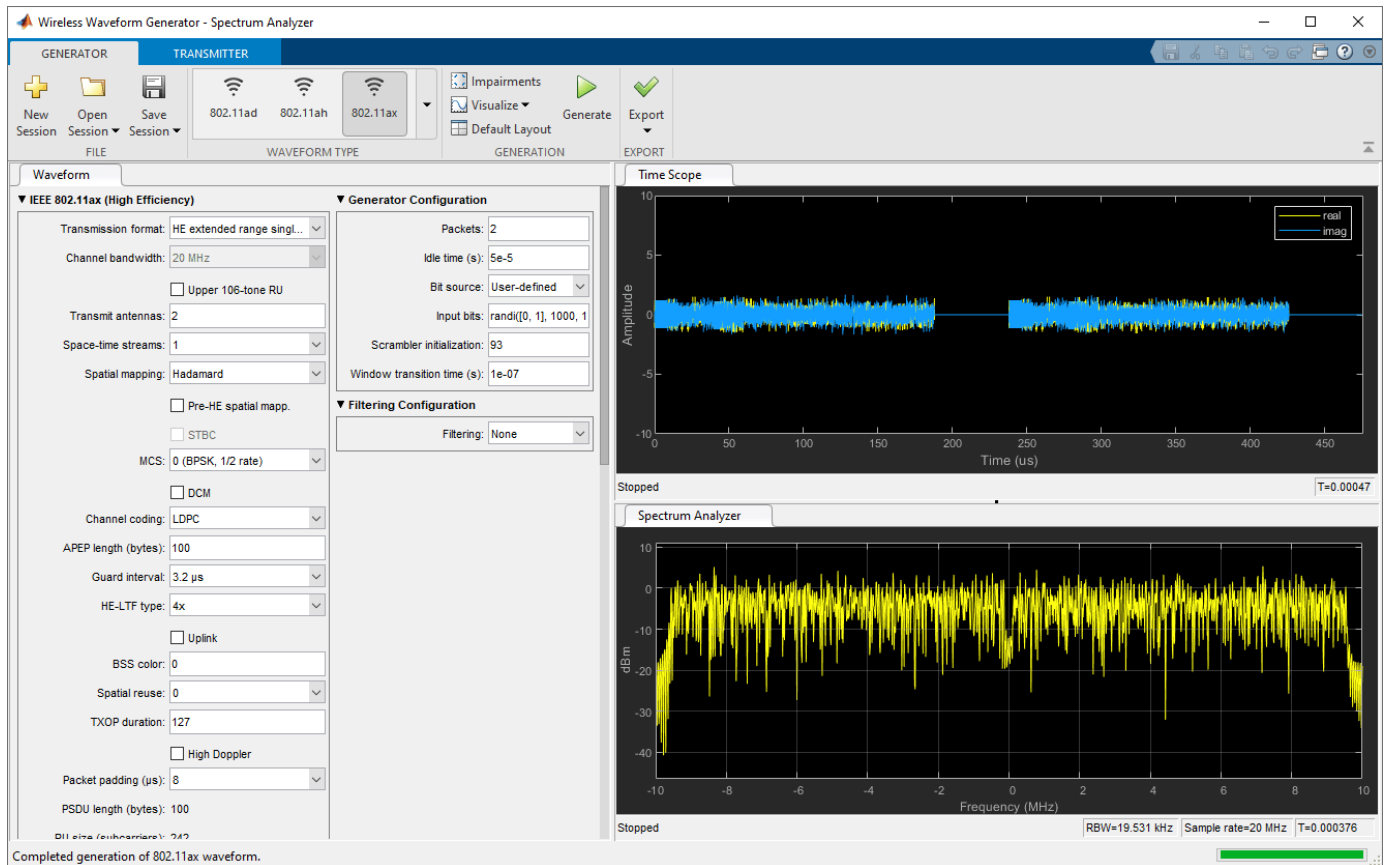
Generate HE MU Waveform with 10 Transmit Antennas

This image shows the **RU & Subcarrier Assignment** visualization results for a high-efficiency multi-user (HE MU) waveform comprising a single-packet. The transmission bandwidth is 80 MHz and the number of antennas is 10, which requires configuration of the **Pre-HE cyclic shifts (ns)** parameter. The transmission includes midamble in the HE-Data field. The allocation indices determine the RU and subcarrier assignment. This image shows RU numbers, sizes, indices, and allocated users, and the result of clicking the first RU in the transmission. All resource units use Fourier spatial mapping. Users specified by indices 1-3 use a modulation and coding scheme (MCS) index of 7, the user specified by index 4 uses an MCS index of 10, and the users specified by indices 5 and 6 use an MCS index of 11. The app displays the resulting PSDU length for each user. All other format and configuration parameters take their default values.

Transmit WLAN Waveform

This feature requires "Instrument Control Toolbox"™ software. 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` (Instrument Control Toolbox) function.

References

[1] IEEE P802.11ax/D4.1. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 1: Enhancements for High Efficiency WLAN." Draft Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[2] IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

[3] IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016 as amended by IEEE Std 802.11ai™-2016). "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation." IEEE Standard for Information technology — Telecommunications and information exchange between systems. Local and metropolitan area networks — Specific requirements.

See Also

Apps
WLAN Waveform Generator

More About

- "Waveform Generation" on page 3-22

External Websites

- "Use Wireless Waveform Generator App"

Generate and Parse WLAN MAC Frames

This example shows how to configure and generate WLAN MAC frames, then recover the payload of MSDUs by parsing the MAC frame.

Introduction

The IEEE® 802.11™ family of standards supports four types of MAC frame: control, data, management, and extension. Within each of these types, the standard defines a range of subtypes, each of which serves a specific purpose in an 802.11™ network.

This example demonstrates how to configure, generate, and parse MPDUs and A-MPDUs by using WLAN Toolbox™ configuration objects and functions.

Generate and Decode MPDU

Create a MAC frame configuration object for a Data frame, specifying a high-efficiency single-user (HE SU) physical layer (PHY) configuration.

```
cfgMPDU = wlanMACFrameConfig('FrameType','Data','FrameFormat','HE-SU');
```

Specify an MSDU as a numeric vector of octets in bit format. You can also specify MSDUs as a character vector or string of octets in hexadecimal format.

```
msdu = randi([0 255],32,1);
```

Generate the MPDU by calling the `wlanMACFrame` function, specifying bits as the output format.

```
[mpdu,mpduLength] = wlanMACFrame(msdu,cfgMPDU,'OutputFormat','bits');
```

Recover the MSDU by calling the `wlanMPDUDecode` function. The function also returns the MAC frame configuration object and the status of the decoding. Check that the decoding operation returns the correct frame format and display the status.

```
[rxCfgMPDU,payload,status] = wlanMPDUDecode(mpdu,wlanHESUConfig);
disp(isequal(cfgMPDU.FrameFormat,rxCfgMPDU.FrameFormat))

    1

disp(status)

    Success
```

Generate and Parse A-MPDU

Create a configuration object for a QoS Data MAC frame, specifying an HE SU PHY configuration. Enable MPDU aggregation and disable MSDU aggregation.

```
cfgAMPDU = wlanMACFrameConfig('FrameType','QoS Data','FrameFormat','HE-SU',...
    'MPDUAggregation',true,'MSDUAggregation',false);
```

Specify a cell array of MSDUs, specifying each MSDU as a numeric vector of octets in bit format. You can also specify MSDUs as a character vector or string of octets in hexadecimal format.

```
msduList = repmat({randi([0 255],32,1)},1,4);
```

Generate the MPDU for a HE SU PHY configuration by calling the `wlanMACFrame` function.

```matlab
cfgPHY = wlanHESUConfig('MCS',5);
[ampdu,ampduLength] = wlanMACFrame(msduList,cfgAMPDU,cfgPHY,'OutputFormat','bits');
```

Deaggregate the A-MPDU to return the MPDU list by calling the `wlanAMPDUDeaggregate` function. The function also returns the result of the delimiter cyclic redundancy check (CRC) and the status of A-MPDU deaggregation.

```matlab
[mpduList,delimiterCRCFailure,status] = wlanAMPDUDeaggregate(ampdu,cfgPHY);
```

Display the number of delimiter CRC failures and the status of deaggregation.

```matlab
disp(nnz(delimiterCRCFailure))

     0
```

```matlab
disp(status)

    Success
```

Obtain the MSDUs by decoding the deaggregated MPDUs with the `wlanMPDUDecode` function and display the status of the decoding process.

```matlab
if strcmp(status,'Success')
    for i = 1:numel(mpduList)
        if ~delimiterCRCFailure(i)
            [cfg,msdu,decodeStatus] = wlanMPDUDecode(mpduList{i},cfgPHY,'DataFormat','octets');
            disp(['MPDU ' num2str(i) ' decoding status: ' char(decodeStatus)])
        end
    end
end

MPDU 1 decoding status: Success
MPDU 2 decoding status: Success
MPDU 3 decoding status: Success
MPDU 4 decoding status: Success
```

See Also

More About

- "802.11ac Waveform Generation with MAC Frames"
- "802.11 MAC Frame Generation"
- "802.11 MAC Frame Decoding"

WLAN Channel Models

This example demonstrates passing WLAN S1G, VHT, HT, and non-HT format waveforms through appropriate fading channel models. When simulating a WLAN communications link, viable options for channel modeling include the TGah,TGn and TGac models from WLAN Toolbox™ and the additive white Gaussian noise (AWGN) and 802.11g models from Communications Toolbox™. In this example, it is sufficient to set the channel model sampling frequency to match the channel bandwidth because no front-end filtering is applied to the signal and the oversampling rate is 1.

In each section of this example, you:

- Create a waveform.
- Transmit it through a fading channel with noise added.
- Use a spectrum analyzer to display the waveform before and after it passes through the noisy fading channel.

Pass S1G Waveform Through TGah SISO Channel

Create a bit stream to use when generating the WLAN S1G format waveform.

```
bits = randi([0 1],1000,1);
```

Create an S1G configuration object, and then generate a 2 MHz S1G waveform. Calculate the signal power.

```
s1g = wlanS1GConfig;
preChS1G = wlanWaveformGenerator(bits,s1g);
```

Pass the signal through a TGah SISO channel with AWGN noise (SNR=10 dB) and a receiver with a 9 dB noise figure. Recall that the channel model sampling frequency is equal to the bandwidth in this example. Set property values by using name-value pairs.

Create a TGah channel object. Set the channel model sampling frequency and channel bandwidth, enable path loss and shadowing, and use the Model-D delay profile.

```
cbw = s1g.ChannelBandwidth;
fs = 2e6; % Channel model sampling frequency equals the channel bandwidth
tgahChan = wlanTGahChannel('SampleRate',fs,'ChannelBandwidth',cbw, ...
    'LargeScaleFadingEffect','Pathloss and shadowing', ...
    'DelayProfile','Model-D');
```

Create an `AWGN Channel` object with SNR = 10 dB. Determine the signal power in Watts, accounting for the TGah large scale fading pathloss.

```
preChSigPwr_dB = 20*log10(mean(abs(preChS1G)));
sigPwr = 10^((preChSigPwr_dB-tgahChan.info.Pathloss)/10);

chNoise = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (SNR)',...
    'SNR',10,'SignalPower', sigPwr);
```

Pass the S1G waveform through a SISO TGah channel and add the AWGN channel noise.

```
postChS1G = chNoise(tgahChan(preChS1G));
```

Create another `AWGN Channel` object to add receiver noise.

```
rxNoise = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Pass the S1G waveform through the receiver. Choose an appropriate noise variance, nVar, to set the receiver noise level. Here, the receiver noise level is based on the noise variance for a receiver with a 9 dB noise figure. nVar = *kTBF*, where *k* is Boltzmann's constant, *T* is the ambient temperature of 290 K, *B* is the bandwidth, and *F* is the receiver noise figure.

```
nVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);
```

```
rxS1G = rxNoise(postChS1G,nVar);
```

Display a spectrum analyzer with before-channel and after-channel waveforms. Use SpectralAverages = 10 to reduce noise in the plotted signals.

```
title = '2 MHz S1G Waveform Before and After TGah Channel';
saScope = dsp.SpectrumAnalyzer('SampleRate',fs,'ShowLegend',true,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99,'Title',title,...
    'ChannelNames',{'Before','After'});
saScope([preChS1G,rxS1G])
```
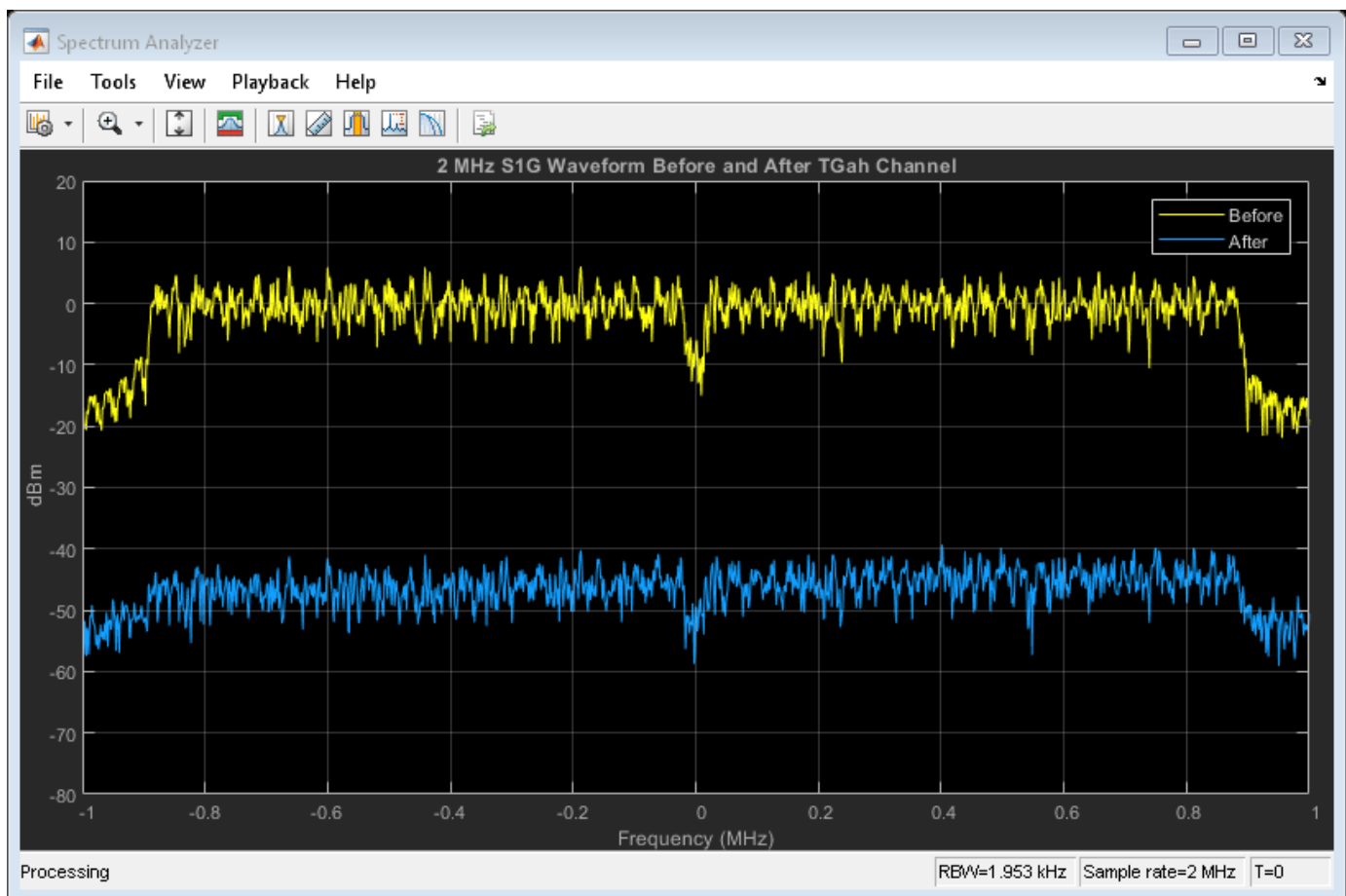

Path loss accounts for the roughly 50 dB of separation between the waveform before and after it passes through the TGah channel. The path loss results from the default transmitter-to-receiver distance of 3 meters, and from shadowing effects. The signal level variation shows the frequency selectivity of the delay profile across the frequency spectrum.

Pass VHT Waveform Through TGac SISO Channel

Create a bit stream to use when generating the WLAN VHT format waveform.

```
bits = randi([0 1],1000,1);
```

Create a VHT configuration object, and generate an 80 MHz VHT waveform. Calculate the signal power.

```
vht = wlanVHTConfig;
preChVHT = wlanWaveformGenerator(bits,vht);
```

Pass the signal through a TGac SISO channel with AWGN noise (SNR=10 dB) and a receiver with a 9 dB noise figure. Recall that the channel model sampling frequency is equal to the bandwidth in this example. Set parameters using `Name,Value` pairs.

Create a TGac channel object. Set the channel model sampling frequency and channel bandwidth, enable path loss and shadowing, and use the Model-D delay profile.

```
cbw = vht.ChannelBandwidth;
fs = 80e6; % Channel model sampling frequency equals the channel bandwidth
tgacChan = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',cbw, ...
    'LargeScaleFadingEffect','Pathloss and shadowing', ...
    'DelayProfile','Model-D');
```

Create an `AWGN Channel` object with SNR = 10 dB. Determine the signal power in Watts, accounting for the TGac large scale fading pathloss.

```
preChSigPwr_dB = 20*log10(mean(abs(preChVHT)));
sigPwr = 10^((preChSigPwr_dB-tgacChan.info.Pathloss)/10);

chNoise = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (SNR)',...
    'SNR',10,'SignalPower', sigPwr);
```

Pass the VHT waveform through a SISO TGac channel and add the AWGN channel noise.

```
postChVHT = chNoise(tgacChan(preChVHT));
```

Create another `AWGN Channel` object to add receiver noise.

```
rxNoise = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Pass the VHT waveform through the receiver. Choose an appropriate noise variance, nVar, to set the receiver noise level. Here, the receiver noise level is based on the noise variance for a receiver with a 9 dB noise figure. `nVar` = $kTBF$, where k is Boltzmann's constant, T is the ambient temperature of 290 K, B is the bandwidth, and F is the receiver noise figure.

```
nVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);

rxVHT = rxNoise(postChVHT,nVar);
```

Display a spectrum analyzer with before-channel and after-channel waveforms. Use `SpectralAverages` = 10 to reduce noise in the plotted signals.

```
title = '80 MHz VHT Waveform Before and After TGac Channel';
saScope = dsp.SpectrumAnalyzer('SampleRate',fs,'ShowLegend',true,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99,'Title',title,...
```

```
              'ChannelNames',{'Before','After'});
       saScope([preChVHT,rxVHT])
```
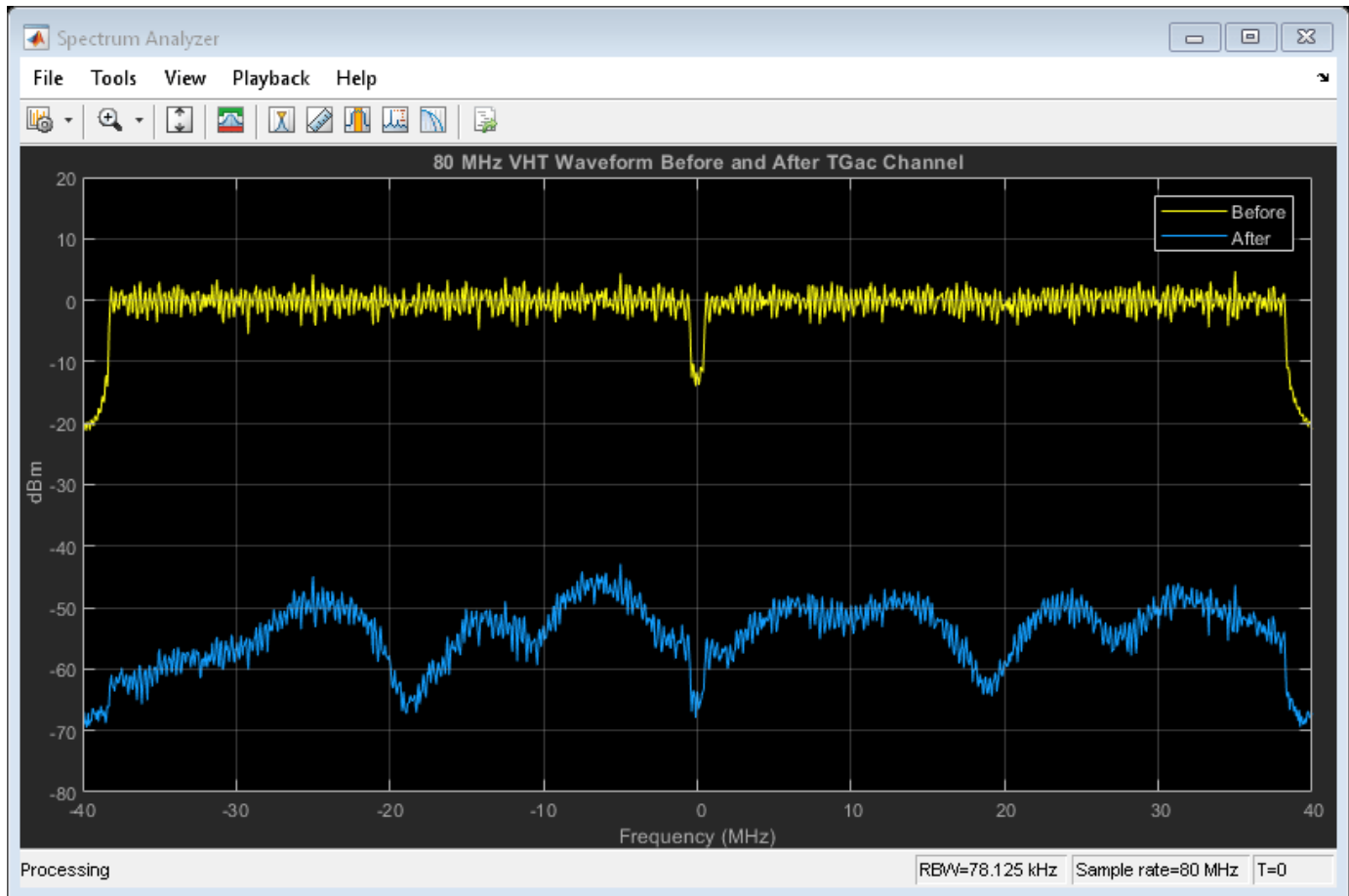

Path loss accounts for the roughly 50 to 60 dB of separation between the waveform before and after it passes through the TGac channel. The path loss results from the default transmitter-to-receiver distance of 3 meters, and from shadowing effects. The signal level variation shows the frequency selectivity of the delay profile across the frequency spectrum.

Pass HT Waveform Through TGn SISO Channel

Create a bit stream to use when generating the WLAN HT format waveform.

```
bits = randi([0 1],1000,1);
```

Create an HT configuration object, and generate an HT waveform.

```
ht = wlanHTConfig;
preChHT = wlanWaveformGenerator(bits,ht);
```

Pass the signal through a TGn SISO channel with AWGN noise (SNR=10 dB) and a receiver with a 9 dB noise figure. Recall that the channel model sampling frequency is equal to the bandwidth in this example. Set parameters using `Name,Value` pairs.

Create a TGn channel object. Set the channel model sampling frequency and channel bandwidth, enable path loss and shadowing, and use the Model-F delay profile.

```
fs = 20e6; % Channel model sampling frequency equals the channel bandwidth
tgnChan = wlanTGnChannel('SampleRate',fs,'LargeScaleFadingEffect', ...
    'Pathloss and shadowing','DelayProfile','Model-F');
```

Pass the HT waveform through a TGn channel. Use the `awgn` function to add channel noise at an SNR level of 10 dB.

```
postChHT = awgn(tgnChan(preChHT),10,'measured');
```

Create an `AWGN Channel` object to add receiver noise.

```
rxNoise = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Pass the HT waveform through the receiver. Choose an appropriate noise variance, nVar, for setting the receiver noise level. Here, the receiver noise is based on the noise variance for a receiver with a 9 dB noise figure. `nVar` = *kTBF,* where *k* is Boltzmann's constant, *T* is the ambient temperature of 290 K, *B* is the bandwidth, and *F* is the receiver noise figure.

```
nVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);
```

```
rxHT = rxNoise(postChHT, nVar);
```

Display a spectrum analyzer with before-channel and after-channel waveforms. Use `SpectralAverages` = 10 to reduce noise in the plotted signals.

```
title = '20 MHz HT Waveform Before and After TGn Channel';
saScope = dsp.SpectrumAnalyzer('SampleRate',fs,'ShowLegend',true,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99,'Title',title,...
    'ChannelNames',{'Before','After'});
saScope([preChHT,postChHT])
```
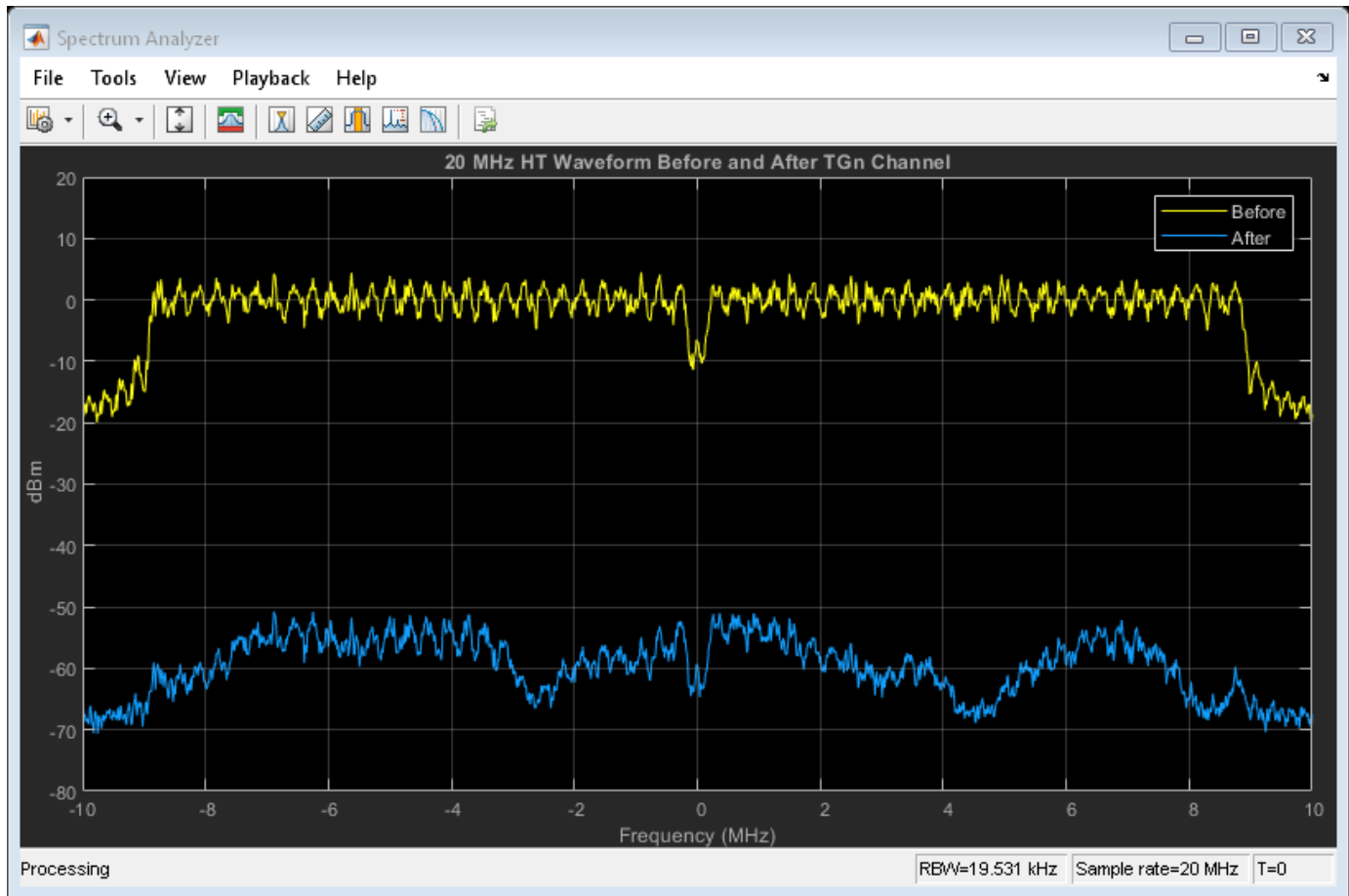
Path loss accounts for the roughly 50 to 60 dB of separation between the waveform before and after it passes through the TGn channel. The path loss results from the default transmitter-to-receiver distance of 3 meters, and from shadowing effects. The signal level variation shows the frequency selectivity of the delay profile across the frequency spectrum.

Pass Non-HT Waveform Through 802.11g Channel

Create a bit stream to use when generating the WLAN Non-HT format waveform.

```
bits = randi([0 1],1000,1);
```

Create a non-HT configuration object, and generate a non-HT waveform.

```
nht = wlanNonHTConfig;
preChNonHT = wlanWaveformGenerator(bits,nht);
```

Calculate free-space path loss for a transmitter-to-receiver separation distance of 3 meters. Create an 802.11g channel object with a 3 Hz maximum Doppler shift and an RMS path delay equal to two times the sample time. Recall that the channel model sampling frequency is equal to the bandwidth in this example. Create an AWGN channel object.

```
dist = 3;
fc = 2.4e9;
pathLoss = 10^(-log10(4*pi*dist*(fc/3e8)));
fs = 20e6; % Channel model sampling frequency equals the channel bandwidth
```

```
maxDoppShift = 3;
trms = 2/fs;
ch802 = comm.RayleighChannel('SampleRate',fs,'MaximumDopplerShift',maxDoppShift,'PathDelays',trms
```

Pass the non-HT waveform through an 802.11g channel. Use the `awgn` function to add channel noise at an SNR level of 10 dB.

```
postChNonHT = awgn(ch802(preChNonHT),10,'measured');
```

Create an `AWGN Channel` object to add receiver noise.

```
rxNoise = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Pass the non-HT waveform through the receiver. Choose an appropriate noise variance, `nVar`, for setting the receiver noise level. Here, the receiver noise is based on the noise variance for a receiver with a 9 dB noise figure. `nVar` = *kTBF*, where *k* is Boltzmann's constant, *T* is the ambient temperature of 290 K, *B* is the bandwidth, and *F* is the receiver noise figure.

```
nVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);

rxNonHT = rxNoise(postChNonHT, nVar)* pathLoss;
```

Display a spectrum analyzer with before-channel and after-channel waveforms. Use `SpectralAverages` = 10 to reduce noise in the plotted signals.

```
title = '20 MHz Non-HT Waveform Before and After 802.11g Channel';
saScope = dsp.SpectrumAnalyzer('SampleRate',fs,'ShowLegend',true,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99,'Title',title,...
    'ChannelNames',{'Before','After'});
saScope([preChNonHT,rxNonHT])
```
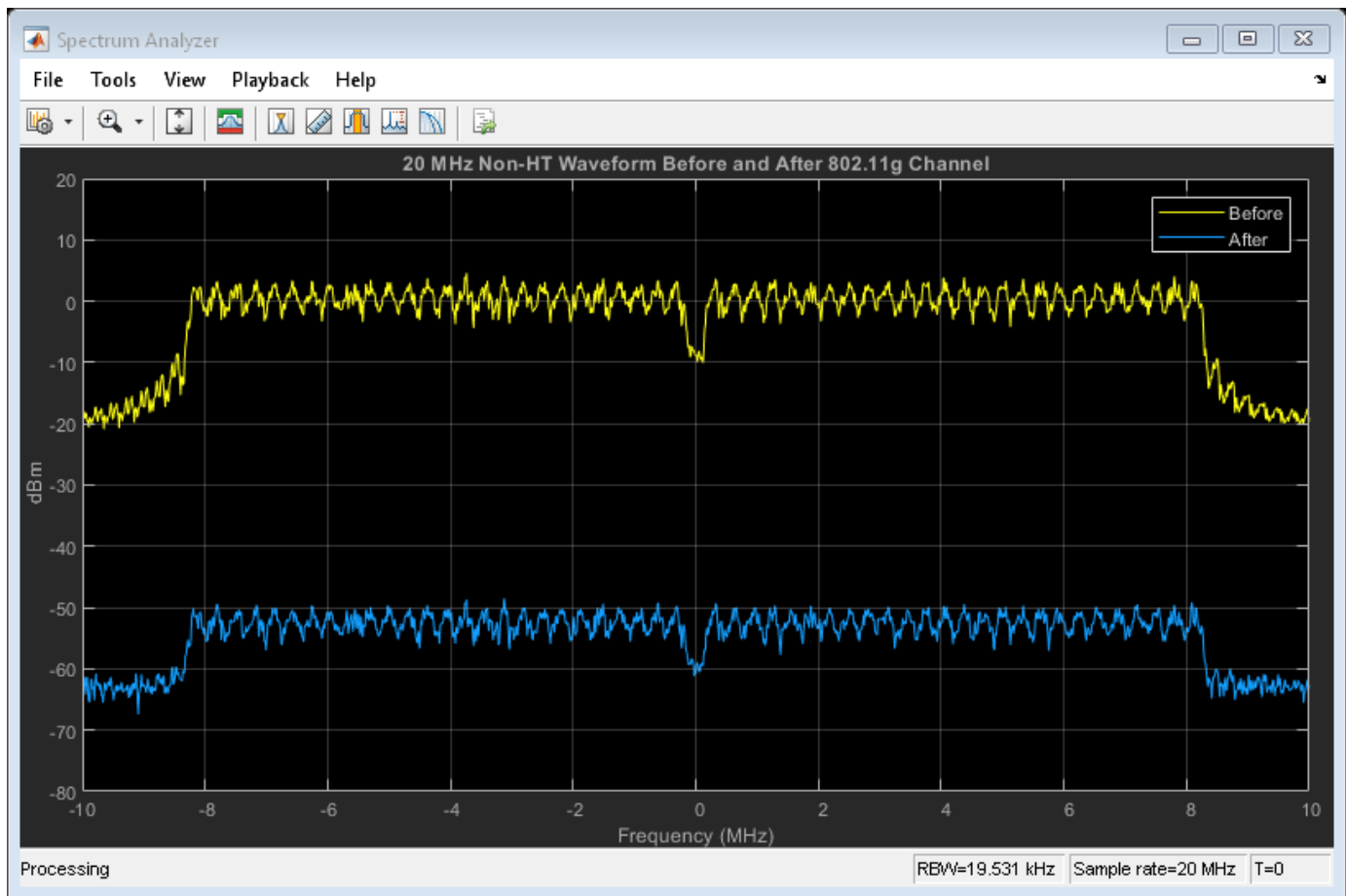
Free-space path loss accounts for the roughly 50 to 60 dB of separation between the waveform before and after it passes through the 802.11g channel. The path loss results from the specified transmitter-to-receiver distance of 3 meters, and from shadowing effects. The signal level variation shows the frequency selectivity of the delay profile across the frequency spectrum.

Pass VHT Waveform Through TGac MIMO Channel

Create a bit stream to use when generating the WLAN VHT format waveform.

```
bits = randi([0 1],1000,1);
```

Create a multi-user VHT configuration object, and generate a VHT waveform. Set the number of transmit antennas to four. Set the number of space-time streams and the number of receive antennas to 3. Because the number of transmit antennas is not equal to the number of space-time streams, the spatial mapping is not direct. Set the spatial mapping to Hadamard.

```
ntx = 4;
nsts = 3;
nrx = 3;
vht = wlanVHTConfig('NumTransmitAntennas',ntx, ...
    'NumSpaceTimeStreams',nsts,'SpatialMapping','Hadamard');
preChVHT = wlanWaveformGenerator(bits,vht);
```

Create TGac MIMO channel and AWGN channel objects. Recall that the channel model sampling frequency is equal to the bandwidth in this example. Disable large-scale fading effects.

```
cbw = vht.ChannelBandwidth;
fs = 80e6; % Channel model sampling frequency equals the channel bandwidth
tgacChan = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',cbw,...
    'NumTransmitAntennas',ntx,'NumReceiveAntennas',nrx);
tgacChan.LargeScaleFadingEffect = 'None';
```

Pass the VHT waveform through a TGac channel. Use the `awgn` function to add channel noise at an SNR level of 10 dB.

```
postChVHT = awgn(tgacChan(preChVHT),10,'measured');
```

Create an `AWGN Channel` object to add receiver noise.

```
rxNoise = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Pass the multi-user VHT waveform through a noisy TGac channel. Choose an appropriate noise variance, nVar, for setting the AWGN level. Here, the AWGN level is based on the noise variance for a receiver with a 9 dB noise figure. `nVar` = $kTBF$, where k is Boltzmann's constant, T is the ambient temperature of 290 K, B is the bandwidth, and F is the receiver noise figure.

```
nVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);
```

```
rxVHT = rxNoise(postChVHT,nVar);
```

Display a spectrum analyzer showing the multiple streams after the channel effects have been added. Use `SpectralAverages` = 10 to reduce noise in the plotted signals.

```
title = '80 MHz VHT 4x3 MIMO Waveform After TGac Channel';
saScope = dsp.SpectrumAnalyzer('SampleRate',fs,'ShowLegend',true,...
    'AveragingMethod','Exponential','ForgettingFactor',0.99,'Title',title,...
    'ChannelNames',{'RX1','RX2','RX3'});
saScope(rxVHT)
```
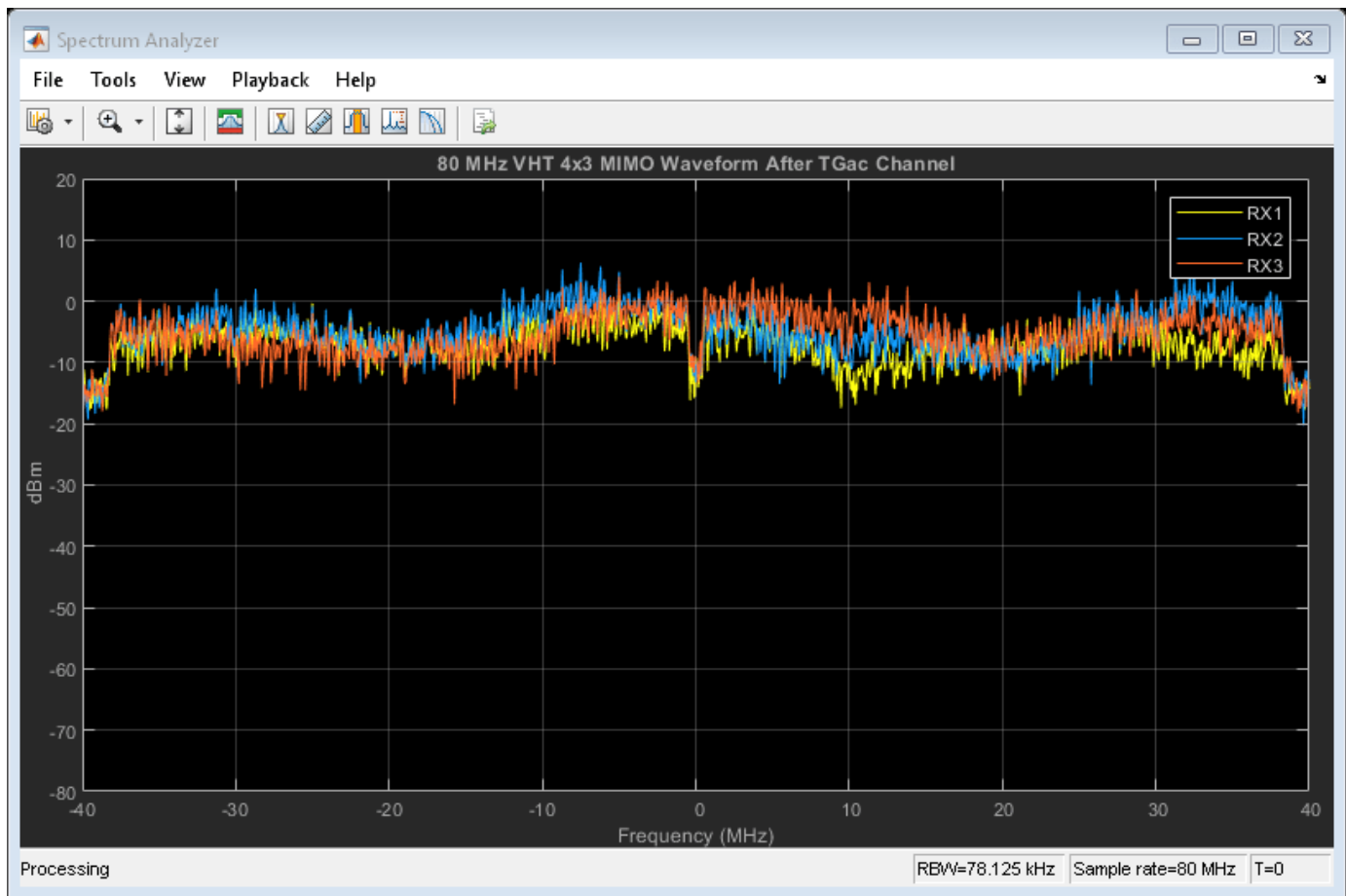
The overlaid signals show the TGac channel variation between the received streams.

References

[1] Erceg, V., L. Schumacher, P. Kyritsi, et al. *TGn Channel Models*. Version 4. IEEE 802.11-03/940r4, May 2004.

[2] Breit, G., H. Sampath, S. Vermani, et al. *TGac Channel Model Addendum*. Version 12. IEEE 802.11-09/0308r12, March 2010.

See Also
`wlanHTConfig` | `wlanNonHTConfig` | `wlanTGacChannel` | `wlanTGnChannel` | `wlanVHTConfig`

Related Examples

Packet Recovery

Received packets are degraded due to radio and channel impairments. Recovery of packet contents requires symbol timing and frequency offset correction, channel estimation, and demodulation and recovery of the preamble and payload. WLAN Toolbox functions perform these operations on VHT, HT-mixed, and non-HT PPDU fields.

VHT Packet Recovery

This example shows how to recover contents from a VHT format waveform.

Generate 80 MHz VHT Waveform

Create a VHT configuration object. Set `APEPLength` to `3200` and `MCS` to `5`. Create a transmission bit stream for the data field. For a VHT waveform, the data field contains `PSDULength*8` bits.

```
cfgVHT = wlanVHTConfig('APEPLength',3200,'MCS',5);
txBits = randi([0 1],cfgVHT.PSDULength*8,1);
```

Create the PPDU fields individually. Create L-STF, L-LTF, L-SIG, VHT-SIG-A, VHT-STF, VHT-LTF, and VHT-SIG-B preamble fields and the VHT-Data field.

```
lstf = wlanLSTF(cfgVHT);
lltf = wlanLLTF(cfgVHT);
lsig = wlanLSIG(cfgVHT);
vhtSigA = wlanVHTSIGA(cfgVHT);
vhtstf = wlanVHTSTF(cfgVHT);
vhtltf = wlanVHTLTF(cfgVHT);
vhtSigB = wlanVHTSIGB(cfgVHT);
vhtData = wlanVHTData(txBits,cfgVHT);
```

Concatenate the individual fields to create a single PPDU waveform.

```
txPPDU = [lstf; lltf; lsig; vhtSigA; vhtstf; vhtltf; vhtSigB; vhtData];
```

Pass VHT Waveform Through TGac SISO Channel

Create TGac SISO and AWGN channel objects.

```
chBW = cfgVHT.ChannelBandwidth;
fs = 80e6;
tgac = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',chBW,...
    'LargeScaleFadingEffect','Pathloss and shadowing');
awgnChan = comm.AWGNChannel('NoiseMethod','Variance','VarianceSource','Input port');
```

Calculate the noise variance for a receiver with a 9 dB noise figure. The noise variance, `noiseVar`, is equal to kTBF, where k is Boltzmann's constant, T is the ambient temperature of 290 K, B is the bandwidth (sample rate), and F is the receiver noise figure. Pass the transmitted waveform through the noisy TGac channel.

```
noiseVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10)
```

```
noiseVar = 2.5438e-12
```

```
rxPPDU = awgnChan(tgac(txPPDU),noiseVar);
```

Recover VHT Preamble Contents from PPDU

In general, the L-STF and L-LTF are processed to perform frequency offset estimation and correction, and symbol timing. For this example, the carrier frequency is not offset and the packet timing is 'on-time'. Therefore, for accurate demodulation, determination of carrier frequency offset and symbol timing is not required.

Find the start and stop indices for the PPDU fields.

```
fieldInd = wlanFieldIndices(cfgVHT)
```

```
fieldInd = struct with fields:
      LSTF: [1 640]
      LLTF: [641 1280]
      LSIG: [1281 1600]
   VHTSIGA: [1601 2240]
    VHTSTF: [2241 2560]
    VHTLTF: [2561 2880]
   VHTSIGB: [2881 3200]
    VHTData: [3201 12160]
```

The stop index of VHT-SIG-B indicates the preamble length in samples.

```
numSamples = fieldInd.VHTSIGB(2);
```

Plot the preamble and the beginning of the packet data. Add markers to and plot to delineate the packet field boundaries.

```
time = ([0:double(numSamples)-1]/fs)*1e6;
peak = 1.2*max(abs(rxPPDU(1:numSamples)));
fieldMarkers = zeros(numSamples,1);
fieldMarkers(fieldInd.LSTF(2)-1,1) = peak;
fieldMarkers(fieldInd.LLTF(2)-1,1) = peak;
fieldMarkers(fieldInd.LSIG(2)-1,1) = peak;
fieldMarkers(fieldInd.VHTSIGA(2)-1,1) = peak;
fieldMarkers(fieldInd.VHTSTF(2)-1,1) = peak;
fieldMarkers(fieldInd.VHTLTF(2)-1,1) = peak;
fieldMarkers(fieldInd.VHTSIGB(2)-1,1) = peak;
plot(time,abs(rxPPDU(1:numSamples)),time,fieldMarkers)
xlabel ('Time (microseconds)')
ylabel('Magnitude')
title('VHT Preamble')
```
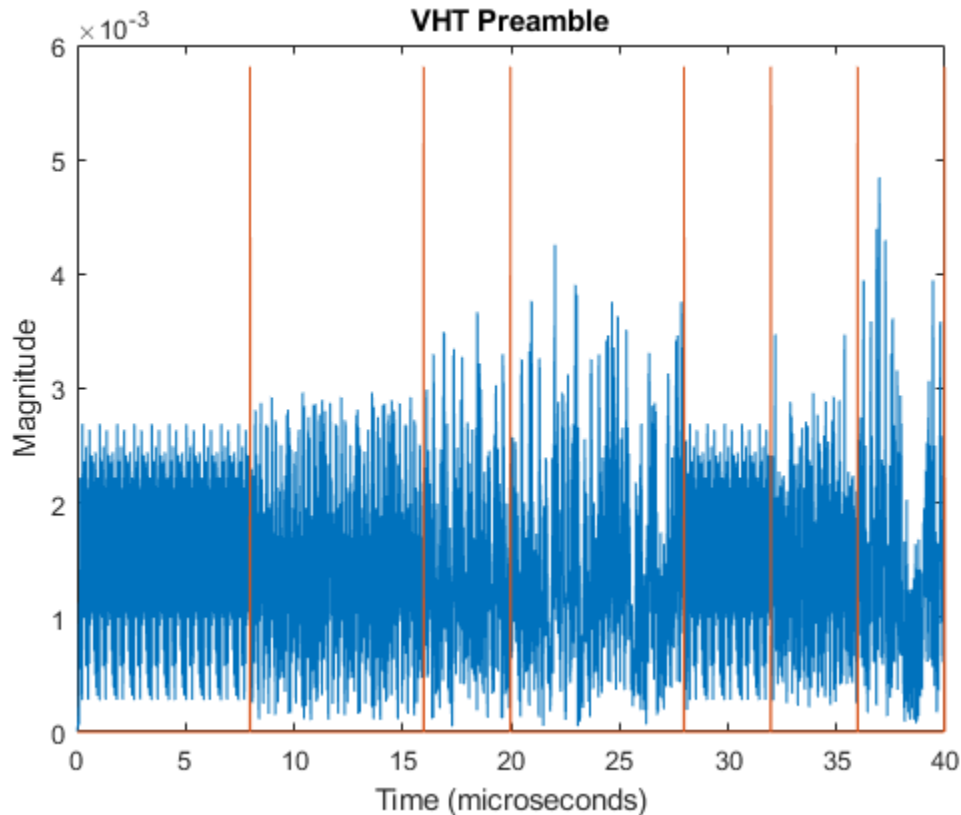
Demodulate the L-LTF and estimate the channel.

```
rxLLTF = rxPPDU(fieldInd.LLTF(1):fieldInd.LLTF(2),:);
demodLLTF = wlanLLTFDemodulate(rxLLTF,cfgVHT);
chEstLLTF = wlanLLTFChannelEstimate(demodLLTF,cfgVHT);
```

Extract the L-SIG field from the received PPDU, recover its information bits and check the CRC.

```
rxLSIG = rxPPDU(fieldInd.LSIG(1):fieldInd.LSIG(2),:);
[recLSIG,failCRC] = wlanLSIGRecover(rxLSIG,chEstLLTF,noiseVar,chBW);
failCRC
```

```
failCRC = logical
    0
```

`failCRC = 0` indicates that CRC passed.

For the VHT format, the L-SIG rate bits are constant and set to `[1 1 0 1]`. Inspect the L-SIG rate information and confirm that this constant sequence is recovered. For the VHT format, the MCS setting in VHT-SIG-A2 determines the actual data rate.

```
rate = recLSIG(1:4)'
```

```
rate = 1x4 int8 row vector

   1   1   0   1
```

Extract the VHT-SIG-A and confirm that the CRC check passed.

```
rxVHTSIGA = rxPPDU(fieldInd.VHTSIGA(1):fieldInd.VHTSIGA(2),:);
[recVHTSIGA,failCRC] = wlanVHTSIGARecover(rxVHTSIGA, ...
    chEstLLTF,noiseVar,chBW);
failCRC
```

```
failCRC = logical
    0
```

Extract the MCS setting from the VHT-SIG-A. For single user VHT, the MCS is located in VHT-SIG-A2 bits 4 through 7.

```
recMCSbits = (recVHTSIGA(29:32))';
recMCS = bi2de(double(recMCSbits))
```

```
recMCS = 5
```

```
isequal(recMCS,cfgVHT.MCS)
```

```
ans = logical
    1
```

The recovered MCS setting matches the MCS value in the configuration object.

Extract and demodulate the VHT-LTF. Use the demodulated signal to perform channel estimation. Use the channel estimate to recover the VHT-SIG-B and VHT-Data fields.

```
rxVHTLTF = rxPPDU(fieldInd.VHTLTF(1):fieldInd.VHTLTF(2),:);
demodVHTLTF = wlanVHTLTFDemodulate(rxVHTLTF,cfgVHT);
chEstVHTLTF = wlanVHTLTFChannelEstimate(demodVHTLTF,cfgVHT);
```

Extract and recover the VHT-SIG-B.

```
rxVHTSIGB = rxPPDU(fieldInd.VHTSIGB(1):fieldInd.VHTSIGB(2),:);
recVHTSIGB = wlanVHTSIGBRecover(rxVHTSIGB,chEstVHTLTF,noiseVar,chBW);
```

As described in IEEE Std 802.11ac-2013, Table 22-1, the value in the VHT-SIG-B Length field multiplied by 4 is the recovered APEP length for packets carrying data. Verify that the APEP length, contained in the first 19 bits of the VHT-SIG-B, corresponds to the specified APEP length.

```
sigbAPEPbits = recVHTSIGB(1:19)';
sigbAPEPlength = bi2de(double(sigbAPEPbits))*4
```

```
sigbAPEPlength = 3200
```

```
isequal(sigbAPEPlength,cfgVHT.APEPLength)
```

```
ans = logical
    1
```

The recovered value matches the configured APEP Length.

Recover equalized symbols using channel estimates from the VHT-LTF.

```
recPSDU = wlanVHTDataRecover(rxPPDU(fieldInd.VHTData(1):fieldInd.VHTData(2),:),...
    chEstVHTLTF,noiseVar,cfgVHT);
```

Compare transmission and receive PSDU bits.

```
numErr = biterr(txBits,recPSDU)
```

```
numErr = 0
```

The number of bit errors is zero.

HT Packet Recovery

This example shows how to recover content from an HT-format waveform.

Generate 20 MHz HT Waveform

Create an HT configuration object and transmission PSDU. Set `MCS` to 2. For an HT waveform, the data field is `PSDULength*8` bits.

```
cfgHT = wlanHTConfig('MCS',2);
txPSDU = randi([0 1],cfgHT.PSDULength*8,1);
```

Create the PPDU fields individually. Create L-STF, L-LTF, L-SIG, HT-SIG, HT-STF, and HT-LTF preamble fields and the HT-Data field.

```
lstf = wlanLSTF(cfgHT);
lltf = wlanLLTF(cfgHT);
lsig = wlanLSIG(cfgHT);
htsig = wlanHTSIG(cfgHT);
htstf = wlanHTSTF(cfgHT);
htltf = wlanHTLTF(cfgHT);
htData = wlanHTData(txPSDU,cfgHT);
```

Concatenate the individual fields to create a single PPDU waveform.

```
txPPDU = [lstf; lltf; lsig; htsig; htstf; htltf; htData];
```

Pass HT Waveform Through TGn SISO Channel

Create TGn SISO channel and AWGN channel objects.

```
fs = 20e6;
tgnChan = wlanTGnChannel('SampleRate',fs,'LargeScaleFadingEffect','Pathloss and shadowing');
awgnChan = comm.AWGNChannel('NoiseMethod','Variance','VarianceSource','Input port');
```

Calculate the noise variance for a receiver with a 9 dB noise figure. The noise variance, `noiseVar`, is equal to kTBF, where k is Boltzmann's constant, T is the ambient temperature of 290 K, B is the bandwidth (sample rate), and F is the receiver noise figure. Pass the transmitted waveform through the noisy TGn channel.

```
noiseVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);
rxPPDU = awgnChan(tgnChan(txPPDU),noiseVar);
```

Recover HT Preamble Contents from PPDU

In general, the L-STF and L-LTF are processed to perform frequency offset estimation and correction, and symbol timing. For this example, the carrier frequency is not offset and the packet timing is 'on-time'. Therefore, for accurate demodulation, determination of carrier frequency offset and symbol timing is not required.

Find the start and stop indices for the PPDU fields.

```
fieldInd = wlanFieldIndices(cfgHT)

fieldInd = struct with fields:
      LSTF: [1 160]
      LLTF: [161 320]
      LSIG: [321 400]
     HTSIG: [401 560]
     HTSTF: [561 640]
     HTLTF: [641 720]
    HTData: [721 9200]
```

The stop index of HT-LTF indicates the preamble length in samples.

```
numSamples = fieldInd.HTLTF(2);
```

Plot the preamble and the beginning of the packet data. Add markers to and plot to delineate the packet field boundaries.

```
time = ([0:double(numSamples)-1]/fs)*1e6;
peak = 1.2*max(abs(rxPPDU(1:numSamples)));
fieldMarkers = zeros(numSamples,1);
fieldMarkers(fieldInd.LSTF(2)-1,1) = peak;
fieldMarkers(fieldInd.LLTF(2)-1,1) = peak;
fieldMarkers(fieldInd.LSIG(2)-1,1) = peak;
fieldMarkers(fieldInd.HTSIG(2)-1,1) = peak;
fieldMarkers(fieldInd.HTSTF(2)-1,1) = peak;
fieldMarkers(fieldInd.HTLTF(2)-1,1) = peak;
plot(time,abs(rxPPDU(1:numSamples)),time,fieldMarkers)
xlabel ('Time (microseconds)')
ylabel('Magnitude')
title('HT Format Preamble')
```
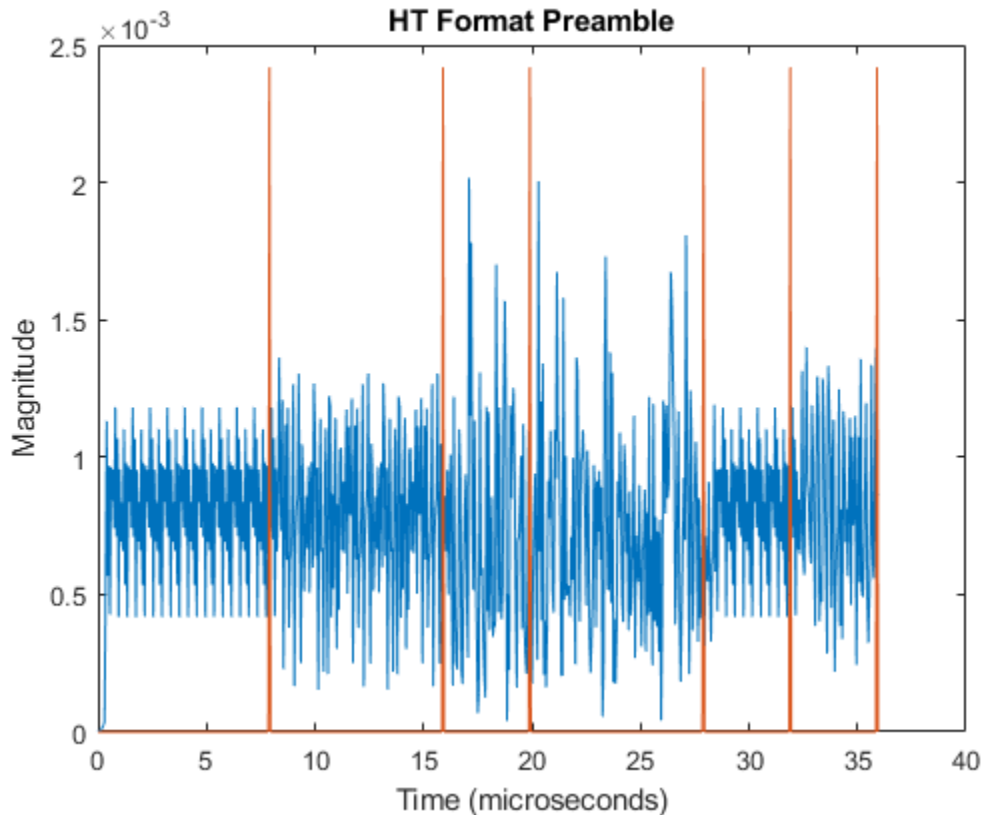
Demodulate the L-LTF and estimate the channel.

```
rxLLTF = rxPPDU(fieldInd.LLTF(1):fieldInd.LLTF(2),:);
demodLLTF = wlanLLTFDemodulate(rxLLTF,cfgHT);
chEstLLTF = wlanLLTFChannelEstimate(demodLLTF,cfgHT);
```

Extract the L-SIG field from the received PPDU and recover its information bits.

```
rxLSIG = rxPPDU(fieldInd.LSIG(1):fieldInd.LSIG(2),:);
[recLSIG,failCRC] = wlanLSIGRecover(rxLSIG,chEstLLTF,noiseVar,cfgHT.ChannelBandwidth);
failCRC
```

```
failCRC = logical
   0
```

`failCRC = 0` indicates that CRC passed.

For the HT format, the L-SIG rate bits are constant and set to `[1 1 0 1]`. Inspect the L-SIG rate information and confirm that this constant sequence is recovered. For the HT format, the MCS setting in HT-SIG determines the actual data rate.

```
rate = recLSIG(1:4)'
```

```
rate = 1x4 int8 row vector

   1   1   0   1
```

Extract the HT-SIG and confirm that the CRC check passed.

```
recHTSIG = rxPPDU(fieldInd.HTSIG(1):fieldInd.HTSIG(2),:);
[recHTSIG,failCRC] = wlanHTSIGRecover(recHTSIG,chEstLLTF,noiseVar,cfgHT.ChannelBandwidth);
failCRC
```

```
failCRC = logical
   0
```

Extract the MCS setting from the HT-SIG. For HT, the MCS is located in HT-SIG bits 0 through 6.

```
recMCSbits = (recHTSIG(1:7))';
recMCS = bi2de(double(recMCSbits))
```

```
recMCS = 2
```

```
isequal(recMCS,cfgHT.MCS)
```

```
ans = logical
   1
```

The recovered MCS setting matches the MCS value in the configuration object.

Extract and demodulate the HT-LTF. Use the demodulated signal to perform channel estimation. Use the channel estimate to recover the HT-Data field.

```
rxHTLTF = rxPPDU(fieldInd.HTLTF(1):fieldInd.HTLTF(2),:);
demodHTLTF = wlanHTLTFDemodulate(rxHTLTF,cfgHT);
chEstHTLTF = wlanHTLTFChannelEstimate(demodHTLTF,cfgHT);
```

Recover HT-Data Contents from PPDU

Recover the received equalized symbols using channel estimates from the HT-LTF.

```
[recPSDU] = wlanHTDataRecover(rxPPDU(fieldInd.HTData(1):fieldInd.HTData(2),:),...
    chEstHTLTF,noiseVar,cfgHT);
```

Compare the transmitted and received PSDU bits, and confirm that the number of bit errors is zero.

```
numErr = biterr(txPSDU,recPSDU)
```

```
numErr = 0
```

Non-HT Packet Recovery

This example steps through recovery of non-HT-format waveform content.

Generate 20 MHz Non-HT Waveform

Create a non-HT configuration object and transmission PSDU. Set MCS to 4.For a non-HT waveform, the data field is PSDULength*8 bits.

```
cfgNonHT = wlanNonHTConfig('MCS',4);
txPSDU = randi([0 1],cfgNonHT.PSDULength*8,1);
```

Create the PPDU fields individually. Use the non-HT-Data contents to check the bit error rate after recovery. Create L-STF, L-LTF, and L-SIG preamble fields and non-HT data field.

```
lstf = wlanLSTF(cfgNonHT);
lltf = wlanLLTF(cfgNonHT);
lsig = wlanLSIG(cfgNonHT);
nhtData = wlanNonHTData(txPSDU,cfgNonHT);
```

Concatenate the individual fields to create a single PPDU waveform.

```
txPPDU = [lstf; lltf; lsig; nhtData];
```

Pass Non-HT Waveform Through 802.11g SISO Channel

Calculate the free-space path loss for a transmitter-to-receiver separation distance of 3 meters. Create an 802.11g channel with a 3 Hz maximum Doppler shift and an RMS path delay equal to two times the sample time. Create an AWGN channel.

```
dist = 3;
pathLoss = 10^(-log10(4*pi*dist*(2.4e9/3e8)));
fs = 20e6;
trms = 2/fs;
maxDoppShift = 3;
ch802 = comm.RayleighChannel('SampleRate',fs,'MaximumDopplerShift',maxDoppShift,'PathDelays',trms
awgnChan = comm.AWGNChannel('NoiseMethod','Variance','VarianceSource','Input port');
```

Calculate the noise variance for a receiver with a 9 dB noise figure. The noise variance, `noiseVar`, is equal to kTBF, where k is Boltzmann's constant, T is the ambient temperature of 290 K, B is the bandwidth (sample rate), and F is the receiver noise figure. Pass the transmitted waveform through the noisy, lossy 802.11g channel.

```
noiseVar = 10^((-228.6 + 10*log10(290) + 10*log10(fs) + 9)/10);
rxPPDU = awgnChan(ch802(txPPDU),noiseVar) * pathLoss;
```

Recover Non-HT Preamble Contents from PPDU

In general, the L-STF and L-LTF are processed to perform frequency offset estimation and correction, and symbol timing. For this example, the carrier frequency is not offset and the packet timing is 'on-time'. Therefore, for accurate demodulation, determination of carrier frequency offset and symbol timing is not required.

Find the start and stop indices for the PPDU fields.

```
fieldInd = wlanFieldIndices(cfgNonHT)
```

```
fieldInd = struct with fields:
         LSTF: [1 160]
         LLTF: [161 320]
         LSIG: [321 400]
    NonHTData: [401 7120]
```

The stop index of the L-SIG field indicates the preamble length in samples.

```
numSamples = fieldInd.LSIG(2);
```

Plot the preamble and the beginning of the packet data. Add markers to and plot to delineate the packet field boundaries.
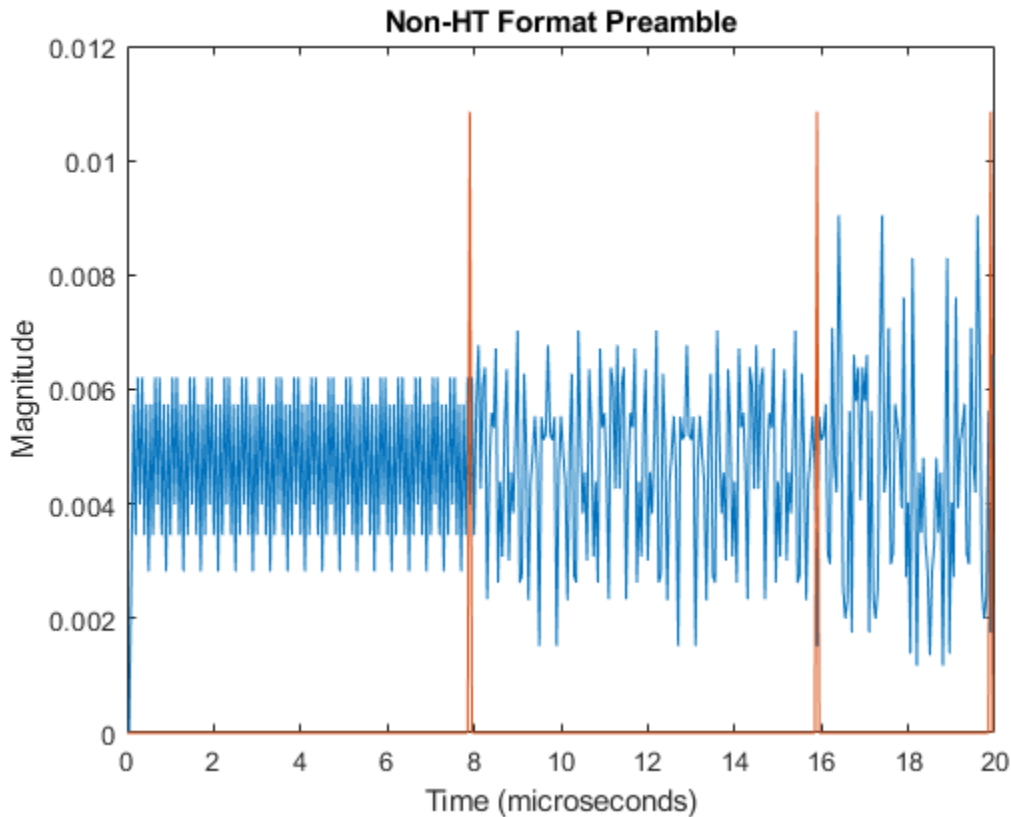
```
time = ((0:double(numSamples)-1)/fs)*1e6;
peak = 1.2*max(abs(rxPPDU(1:numSamples)));
```

```
fieldMarkers = zeros(numSamples,1);
fieldMarkers(fieldInd.LSTF(2)-1,1)  = peak;
fieldMarkers(fieldInd.LLTF(2)-1,1) = peak;
fieldMarkers(fieldInd.LSIG(2)-1,1) = peak;
plot(time,abs(rxPPDU(1:numSamples)),time,fieldMarkers)
xlabel ('Time (microseconds)')
ylabel('Magnitude')
title('Non-HT Format Preamble')
```


Demodulate the L-LTF and estimate the channel.

```
rxLLTF = rxPPDU(fieldInd.LLTF(1):fieldInd.LLTF(2),:);
demodLLTF = wlanLLTFDemodulate(rxLLTF,cfgNonHT);
chEstLLTF = wlanLLTFChannelEstimate(demodLLTF,cfgNonHT);
```

Extract the L-SIG field from the received PPDU and recover its information bits.

```
rxLSIG = rxPPDU(fieldInd.LSIG(1):fieldInd.LSIG(2),:);
recLSIG = wlanLSIGRecover(rxLSIG,chEstLLTF,noiseVar,'CBW20');
```

The first four bits of the L-SIG field, bits 0 through 3, contain the rate information. Confirm that the sequence [1 0 0 1] is recovered. This sequence corresponds to the 24 MHz data rate for the non-HT MCS setting of 4.

```
rate = recLSIG(1:4)'
```

rate = 1x4 int8 row vector

```
1  0  0  1
```

Extract and demodulate the L-LTF. Use the demodulated signal to perform channel estimation. Use the channel estimate to recover the non-HT-Data field.

```
rxLLTF = rxPPDU(fieldInd.LLTF(1):fieldInd.LLTF(2),:);
demodLLTF = wlanLLTFDemodulate(rxLLTF,cfgNonHT);
chEstLLTF = wlanLLTFChannelEstimate(demodLLTF,cfgNonHT);
```

Recover Non-HT-Data Contents from PPDU

Recover equalized symbols using channel estimates from HT-LTF, specifying a zero-forcing equalization method.

```
rxPSDU = rxPPDU(fieldInd.NonHTData(1):fieldInd.NonHTData(2),:);
[recPSDU,~,eqSym] = wlanNonHTDataRecover(rxPSDU,chEstLLTF,noiseVar,cfgNonHT,'EqualizationMethod'
```

Compare the transmitted and received PSDU bits, and confirm that the number of bit errors is zero.

```
numErr = biterr(txPSDU,recPSDU)
```

```
numErr = 0
```

See Also
wlanHTConfig | wlanNonHTConfig | wlanVHTConfig

Related Examples
- "WLAN Channel Models" on page 3-41
- "What Is WLAN?" on page 2-2
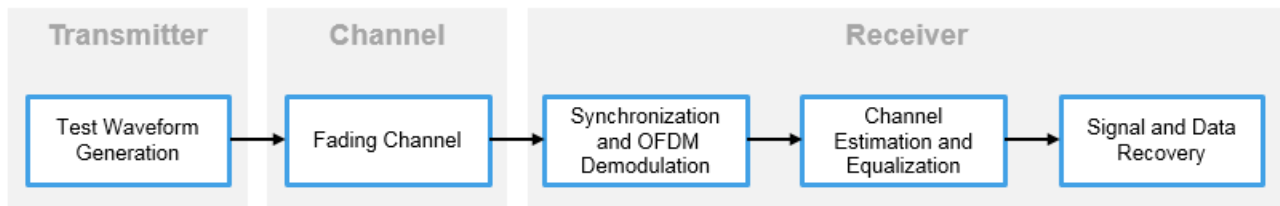- "Build VHT PPDU"
- "Build HT PPDU"
- "Build Non-HT PPDU"

Transmit-Receive Chain

In this section...
"Transmit Processing Chain" on page 3-62
"Receiver Processing Chain" on page 3-65

WLAN Toolbox functionality includes elements of a standard transmitter–channel–receiver processing chain.

- Transmitter functions enable simulation of the various IEEE 802.11 [3] formats. The simulated waveform includes preamble and data fields of the PPDU. You can use this waveform in link-level simulations. You can also use it as a test signal for test devices and equipment.
- Channel functions model various types of AWGN, fading, or moving channel environmental effects.
- Receiver functions recover the transmitted signal.

Transmit Processing Chain

WLAN Toolbox functions enable you to generate waveforms for a complete PPDU or for the individual fields of VHT, HT-mixed, and non-HT format PPDUs.

VHT Data Transmit Processing Chain

As described in IEEE 802.11ac-2013 [4], Section 22 specifies the PHY entity for a very high throughput (VHT) orthogonal frequency division multiplexing (OFDM) system. A VHT STA must be capable of transmitting and receiving HT-PHY and non-HT-PHY-compliant PPDUs. Specifically, the VHT PHY is based on the HT PHY defined in Section 20, which in turn is based on the OFDM PHY defined in Section 18. The VHT PHY extends the maximum number of space-time streams supported to eight and provides support for downlink multi-user (MU) transmissions. A downlink MU transmission supports up to four users with up to four space-time streams per user, with the total number of space-time streams not exceeding eight.

IEEE Std 802.11ac-2013 [4], Section 22 defines requirements for physical layer processing associated with each PPDU field for the VHT format.

---

3. IEEE Std 802.11-2016 Adapted and reprinted with permission from IEEE. Copyright IEEE 2016. All rights reserved.
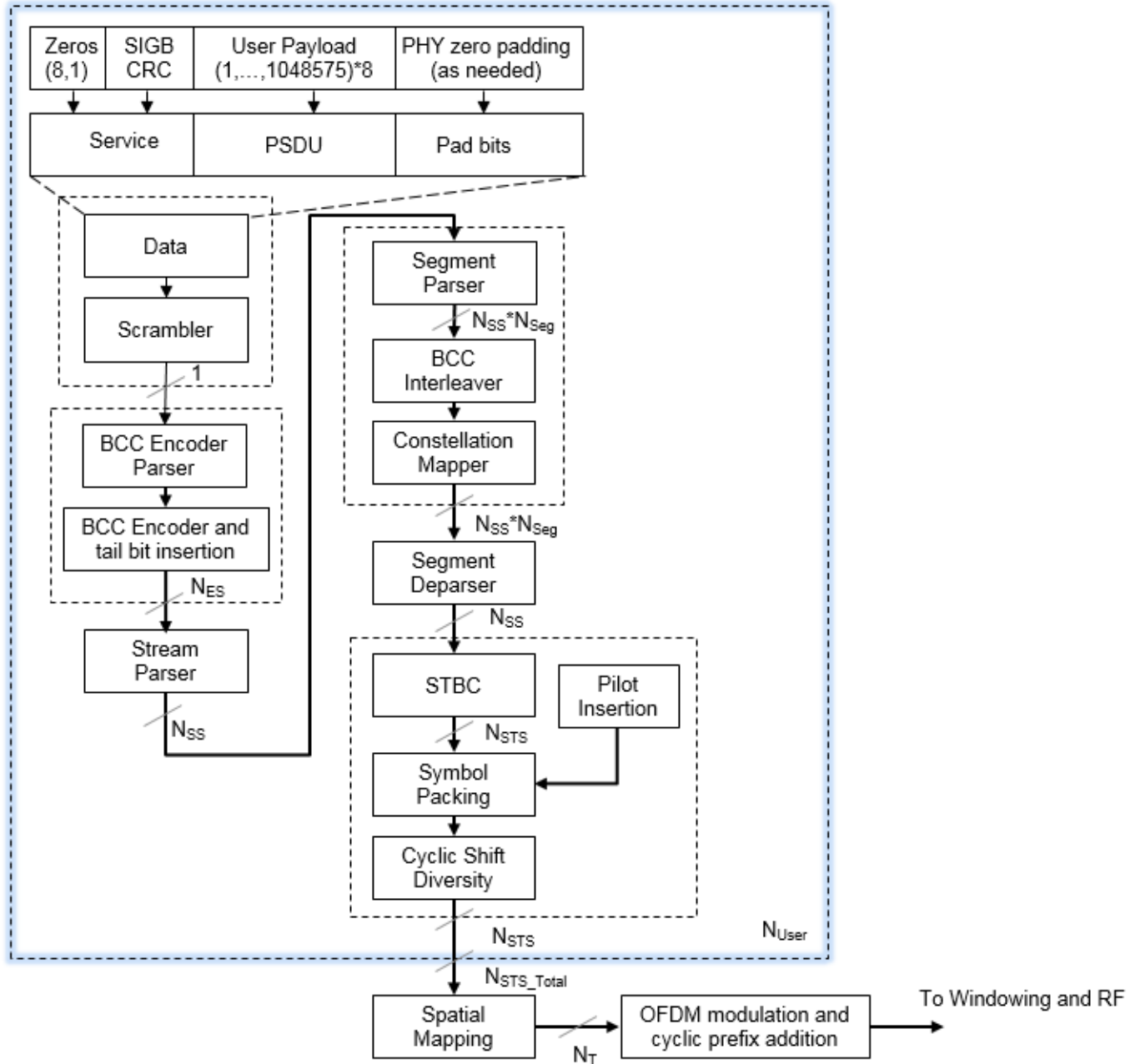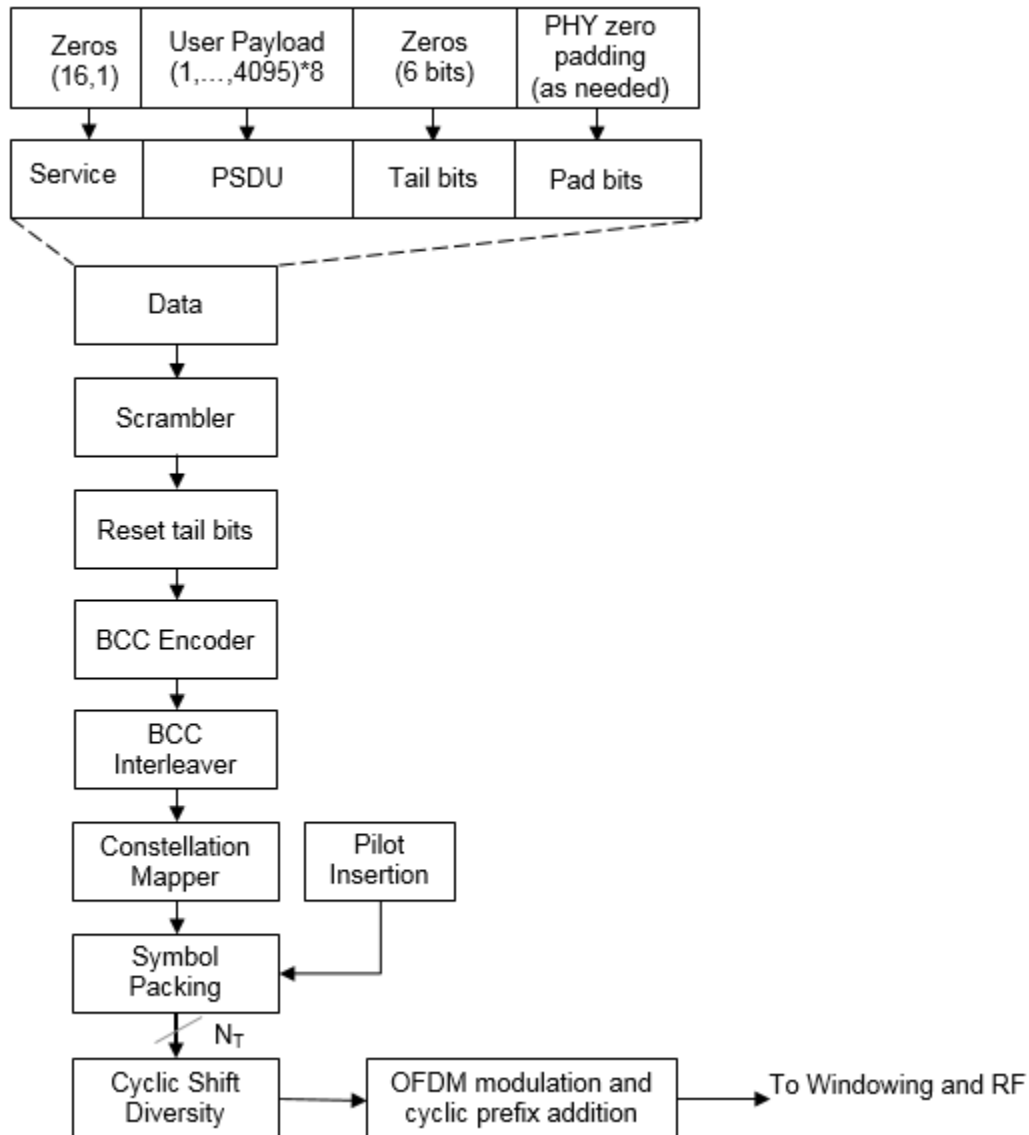
HT Data Transmit Processing Chain

IEEE 802.11-2012 [3], Section 20 defines requirements for physical layer processing associated with each PPDU field for the HT-mixed format.
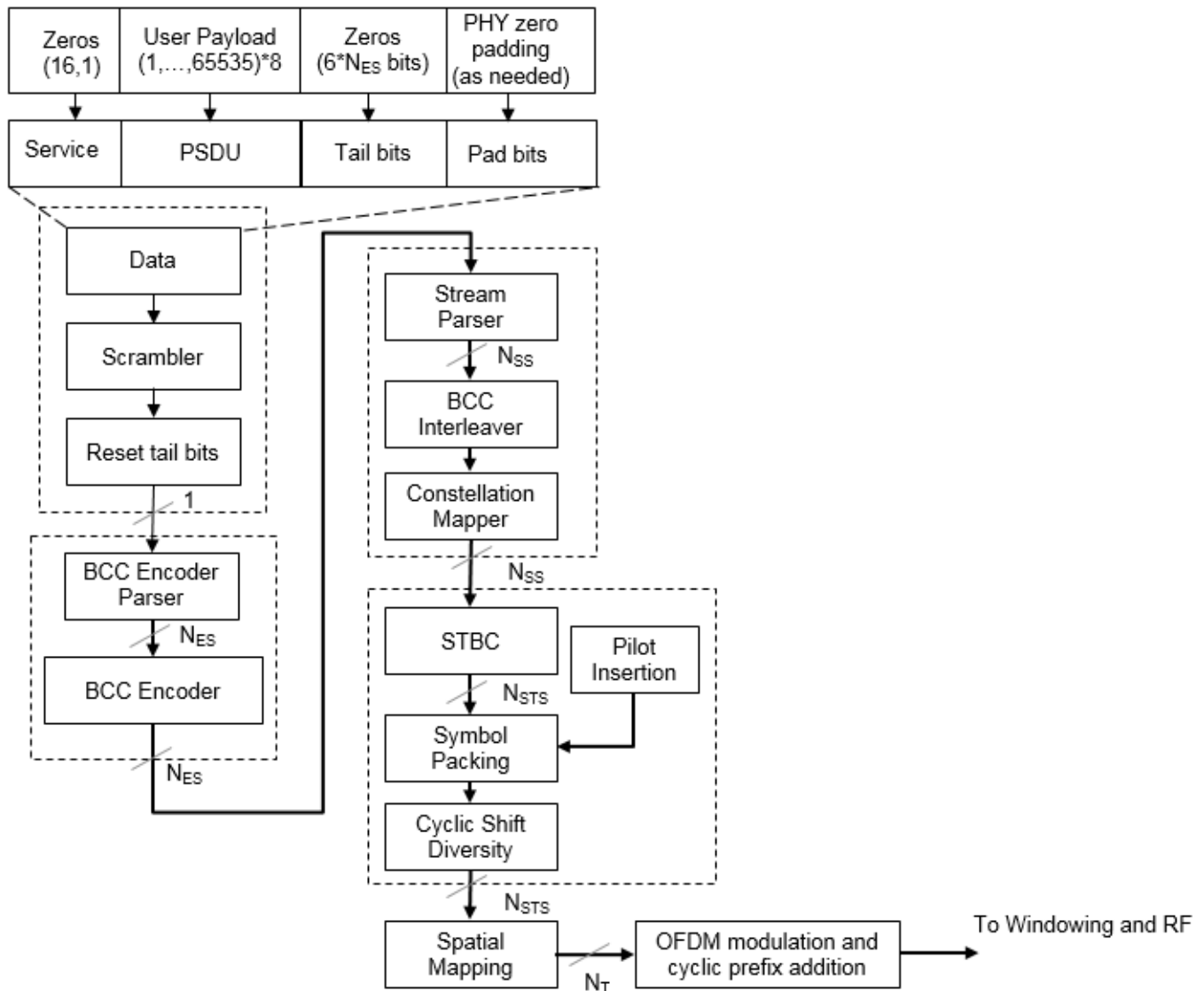
Non-HT Data Transmit Processing Chain

IEEE 802.11-2012 [3], Section 18 defines requirements for physical layer processing associated with each PPDU field for the OFDM modulation scheme. IEEE 802.11-2012 [3], Section 17, and Section 19 define requirements for physical layer processing associated with each PPDU field for the DSSS modulation scheme.
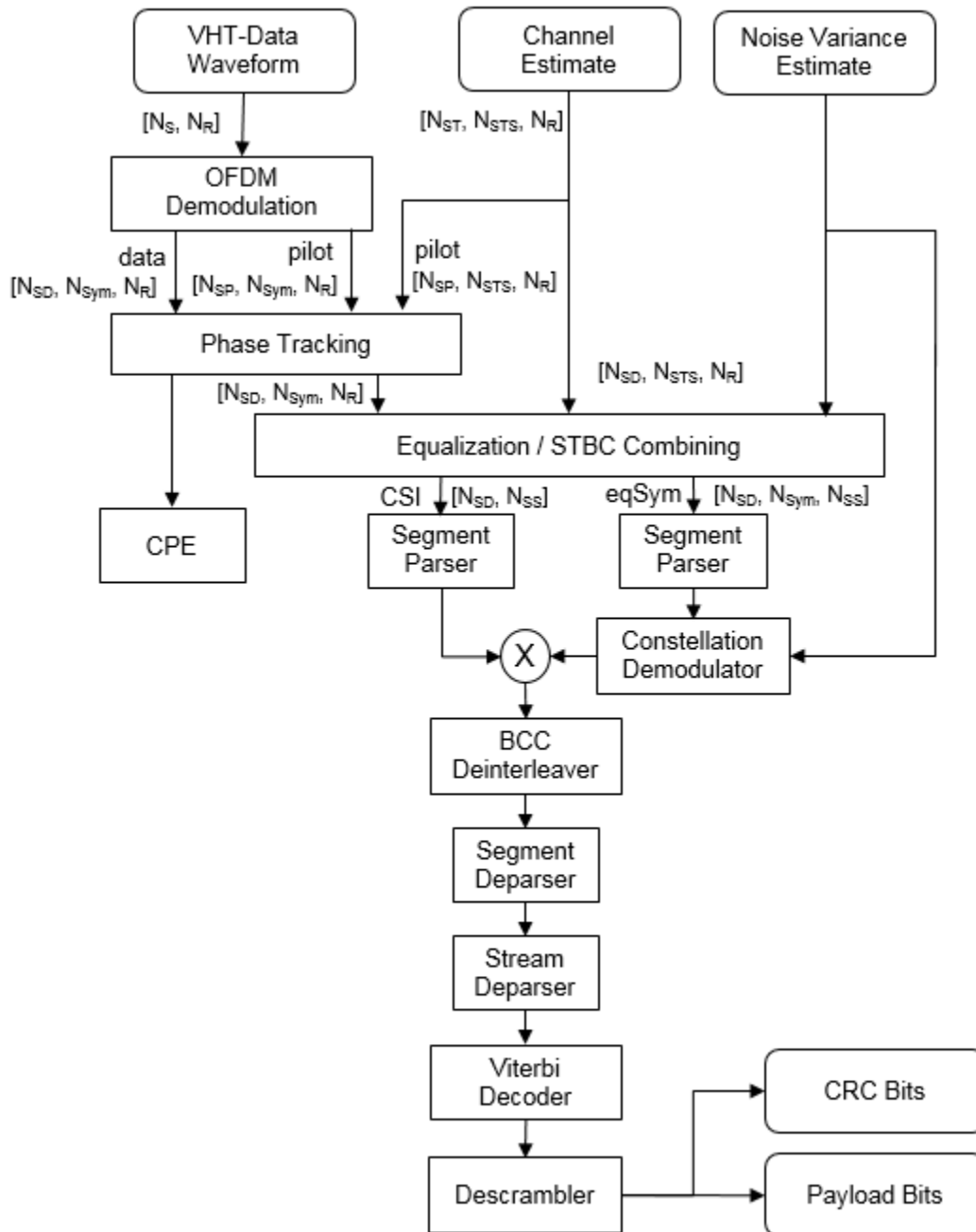
Receiver Processing Chain

WLAN Toolbox functions enable you to recover transmitted VHT, HT-mixed, and non-HT format PPDUs. The receive processing chain includes synchronization, OFDM demodulation, channel estimation, equalization, and signal and data recovery.
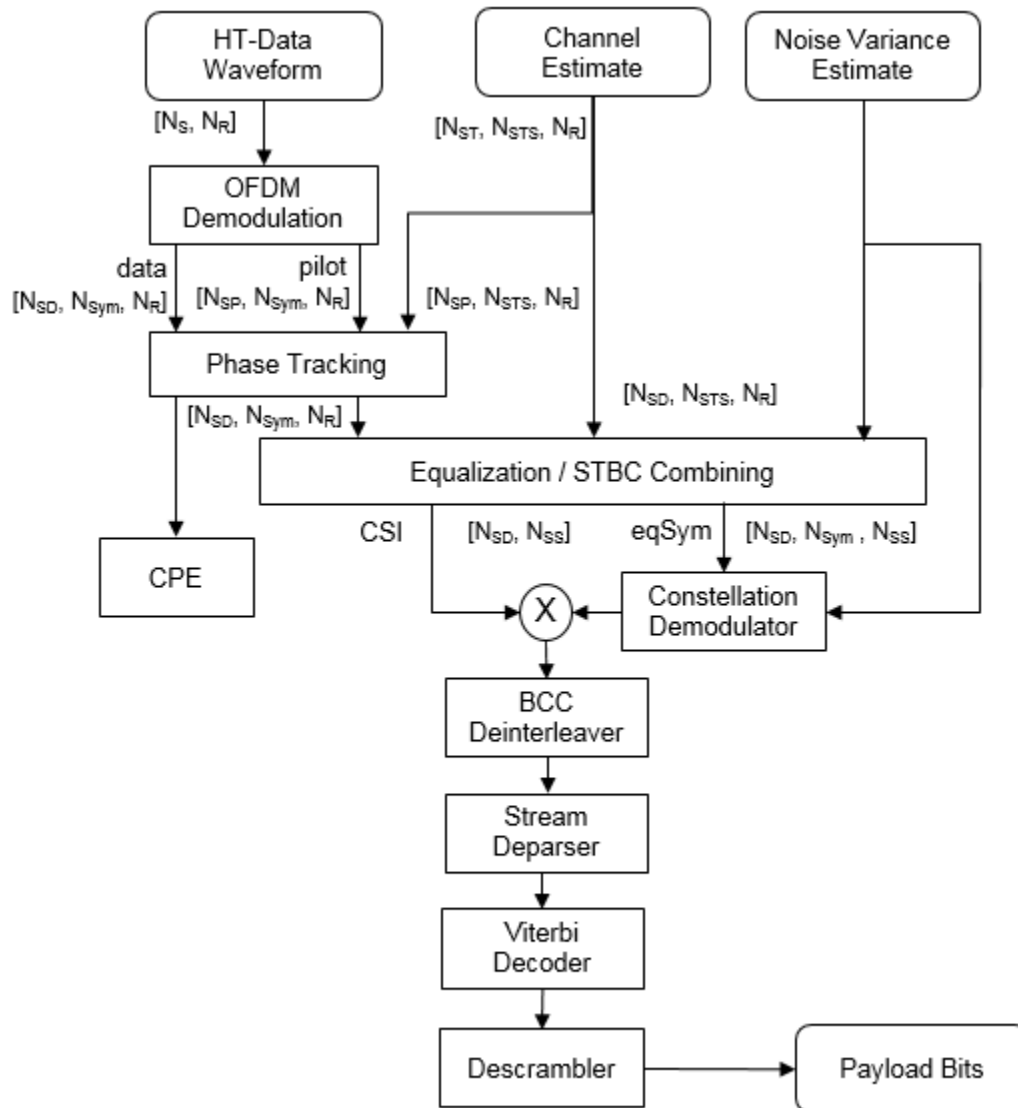
VHT Data Receive Processing Chain

This figure shows the receiver elements used to process the VHT Data field. The "Signal Reception" category includes a list of all receiver functions in the WLAN Toolbox.

HT Data Receive Processing Chain

This figure shows the receiver elements used to process the HT Data field. The "Signal Reception" category includes a list of all receiver functions in the WLAN Toolbox.

Non-HT Data Receive Processing Chain

This figure shows the receiver elements used to process the non-HT Data field. The "Signal Reception" category includes a list of all receiver functions in the WLAN Toolbox.
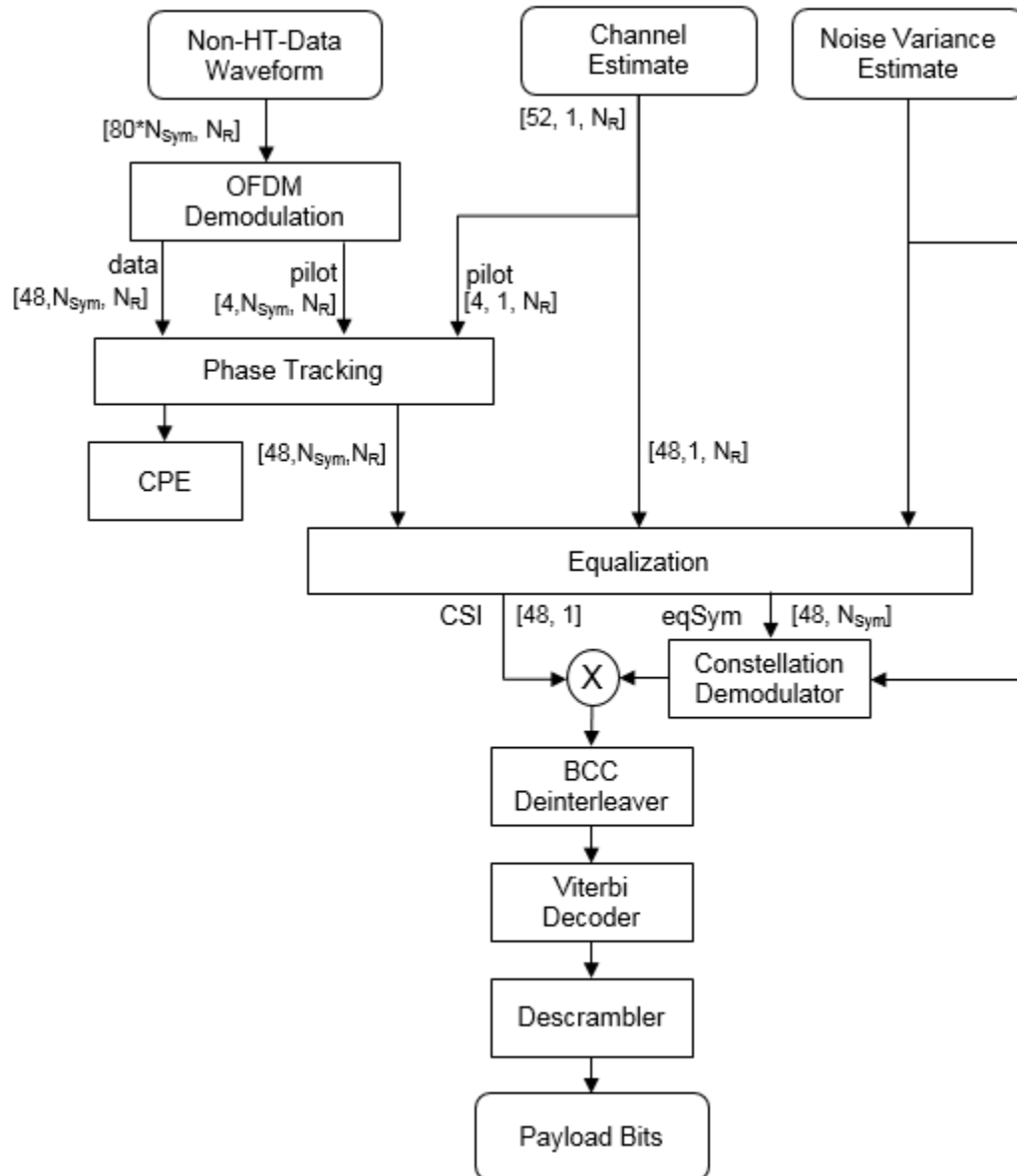
Non-HT-Data Waveform

$[80 \cdot N_{Sym}, N_R]$

OFDM Demodulation

data
$[48, N_{Sym}, N_R]$

pilot
$[4, N_{Sym}, N_R]$

Channel Estimate

$[52, 1, N_R]$

Noise Variance Estimate

pilot
$[4, 1, N_R]$

Phase Tracking

CPE

$[48, N_{Sym}, N_R]$

$[48, 1, N_R]$

Equalization

CSI $[48, 1]$

eqSym $[48, N_{Sym}]$

X

Constellation Demodulator

BCC Deinterleaver

Viterbi Decoder

Descrambler

Payload Bits

References

[1] IEEE 802.11™: Wireless LANs. http://standards.ieee.org/about/get/802/802.11.html

[2] IEEE Std 802.11™-2016 IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[3] IEEE Std 802.11™-2012 IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific

requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[4] IEEE Std 802.11ac™-2013 IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications — Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz.

[5] IEEE Std 802.11ad™-2012 IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications — Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band.

[6] Perahia, E., and R. Stacey. *Next Generation Wireless LANs: 802.11n and 802.11ac*. 2nd Edition. United Kingdom: Cambridge University Press, 2013.

See Also
"Transmit and Recover L-SIG, VHT-SIG-A, VHT-SIG-B in Fading Channel" | "End-to-End VHT Simulation with Frequency Correction"