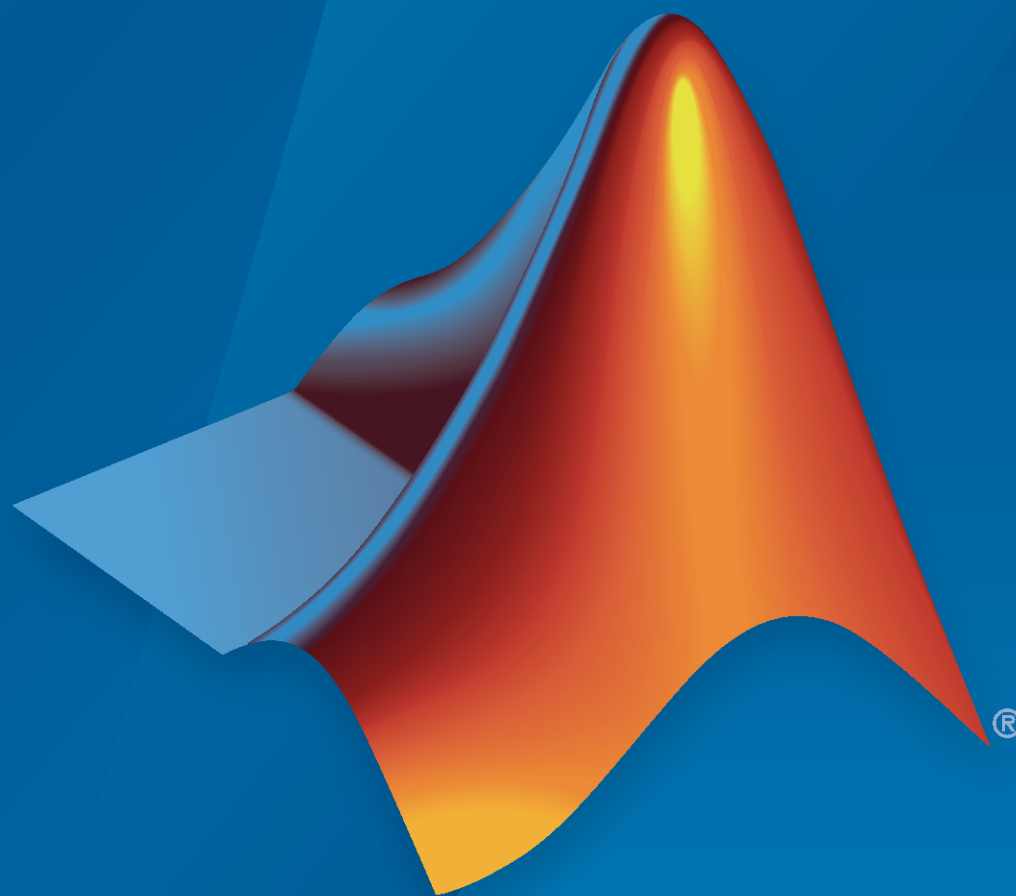# WLAN Toolbox™

User's 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)

Contents

1

Tutorials

WLAN Parameterization

WLAN Toolbox configuration objects initialize, store, and validate configuration properties. The functions in the toolbox use these properties to initialize parameter settings that define the characteristics of waveforms generated and to define the signal recovery process.

Configuration Objects in WLAN Toolbox

The configuration objects are designed specifically as containers to store properties. They also provide some level of data validation for the function inputs that they maintain. Functions perform further data validation across input settings based on run-time conditions.

The configuration objects are optimized for iterative computations that process large streams of data, such as communications systems.

WLAN Toolbox configuration objects define and configure format-specific and function-specific properties. The reference page for each object contains descriptions, valid settings, ranges, and other information about the object properties.

- `wlanHESUConfig` — This high-efficiency single-user (HE SU) configuration object defines and configures HE SU and HE extended-range single-user (HE ER SU) transmission PPDUs.
- `wlanHEMUConfig` — This HE multiuser (HE MU) configuration object defines and configures HE MU transmission PPDUs.
- `wlanDMGConfig` — This directional multi-gigabit (DMG) configuration object defines and configures DMG transmission PPDUs.
- `wlanS1GConfig` — This sub-1-GHz (S1G) configuration object defines and configures S1G transmission PPDUs.
- `wlanVHTConfig` — This very-high-throughput (VHT) configuration object defines and configures VHT transmission PPDUs.
- `wlanHTConfig` — This high-throughput HT configuration object defines and configures HT transmission PPDUs.
- `wlanNonHTConfig` — This non-high-throughput (non-HT) configuration object defines and configures non-HT transmission PPDUs.
- `wlanHERecoveryConfig` — This HE recovery configuration object defines and configures information recovery characteristics for received HE SU, HE ER SU, and HE MU PPDUs

See Also
"WLAN PPDU Structure" on page 1-3 | "Mapping 802.11 Standards to WLAN Toolbox Configuration Objects" on page 1-30 | "Create Configuration Objects" | "What Is WLAN?" | "Multiuser HE Transmission" on page 1-23

WLAN PPDU Structure

Physical Layer Convergence Procedure Protocol Data Unit

IEEE® 802.11™[1] is a packet-based protocol. Each physical layer conformance procedure (PLCP) 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 packet structure vary between 802.11 versions. The transmission vector (*TXVECTOR*) format parameter is classified as:

- *HE* to specify a high-efficiency 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], determines the structure of PPDUs transmitted by an HE STA.

- *DMG* to specify a directional multi-gigabit 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 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 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 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 Format 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 format preamble fields include additional format-specific signaling fields. Each format defines a data field for transmission of user payload data.

HE SU Format PPDU

HE Extended Range SU Format PPDU

HE TB Format PPDU

HE MU Format PPDU
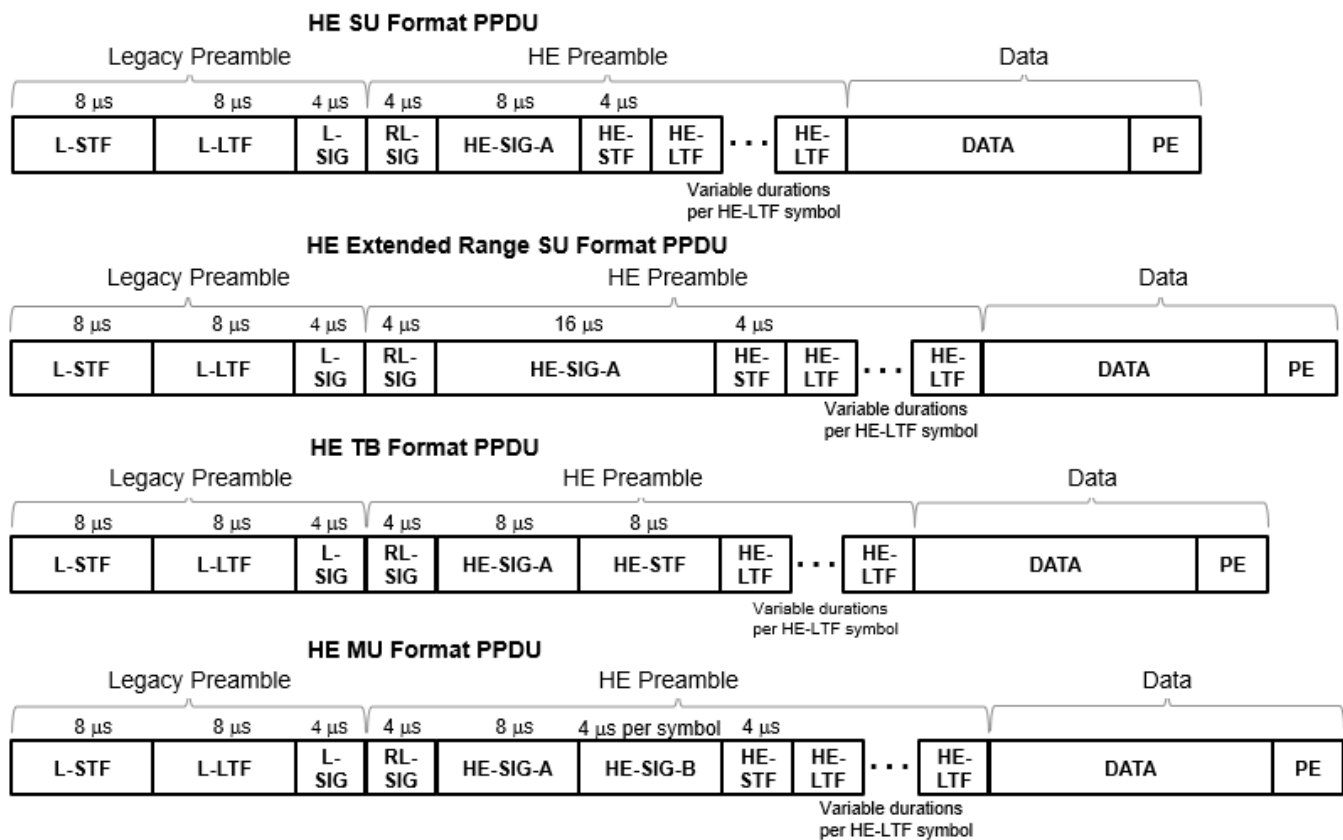
PPDU Field Abbreviation	Description
L-STF	Non-HT Short Training field
L-LTF	Non-HT Long Training field
L-SIG	Non-HT Signal field
RL-SIG	Repeated Non-HT Signal field
HE-SIG-A	HE Signal A field
HE-SIG-B	HE Signal B field
HE-STF	HE Short Training field
HE-LTF	HE Long Training field
Data	Data field carrying 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.
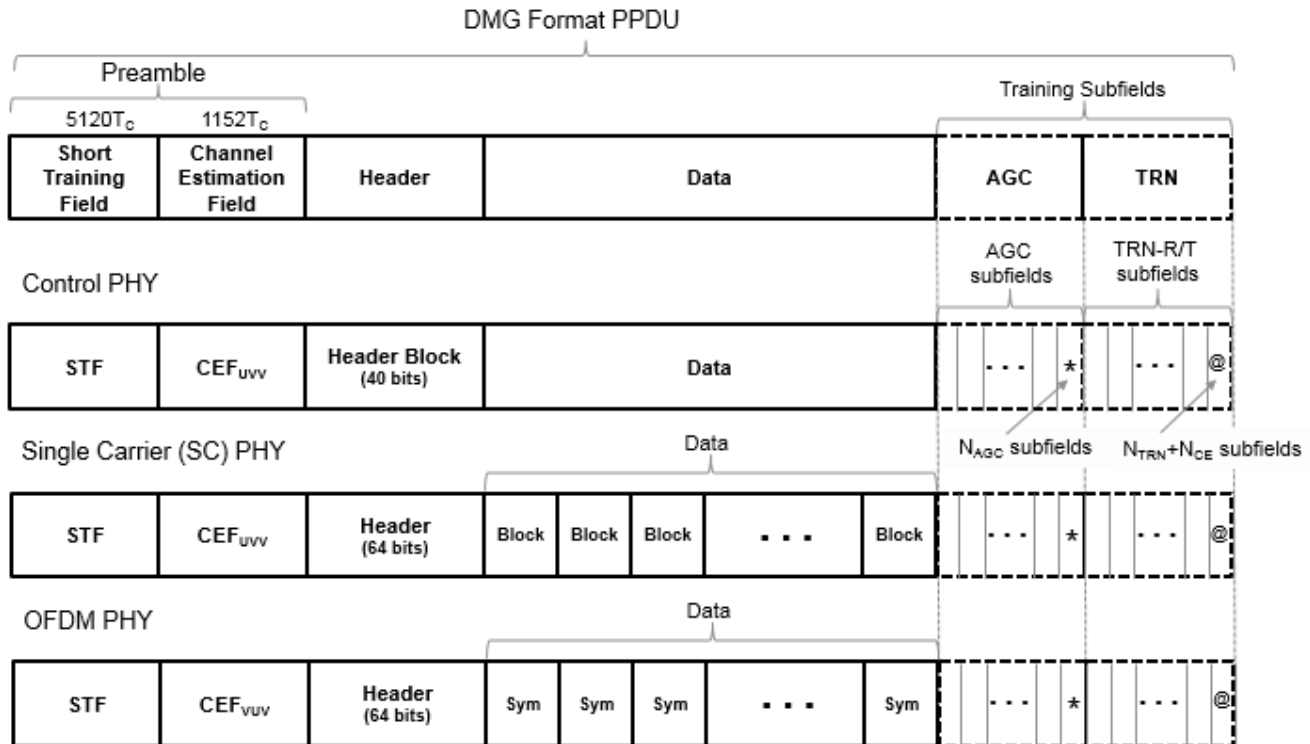
DMG Format PPDU

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 multiuser transmissions. The long preamble PPDU consists of two portions; the omni-directional portion and the beam-changeable portion.

S1G_LONG Format PPDU
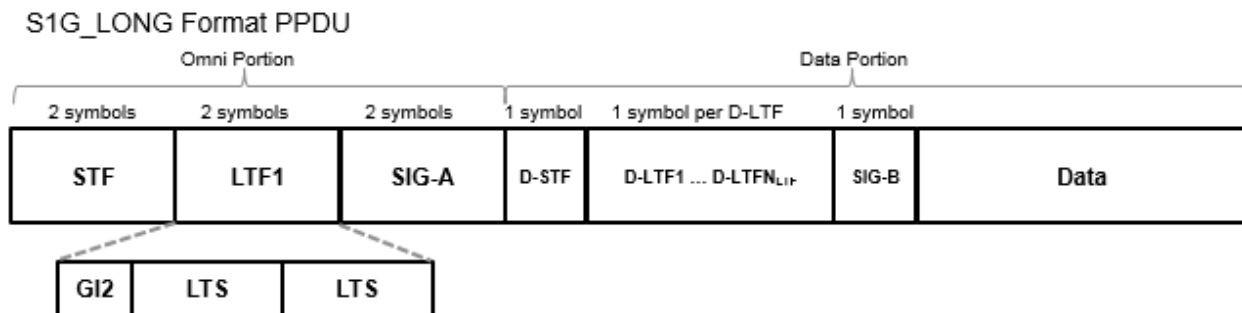
- The omni-directional portion is transmitted to all users without beamforming. It consists of three fields:

  - 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 multiuser 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
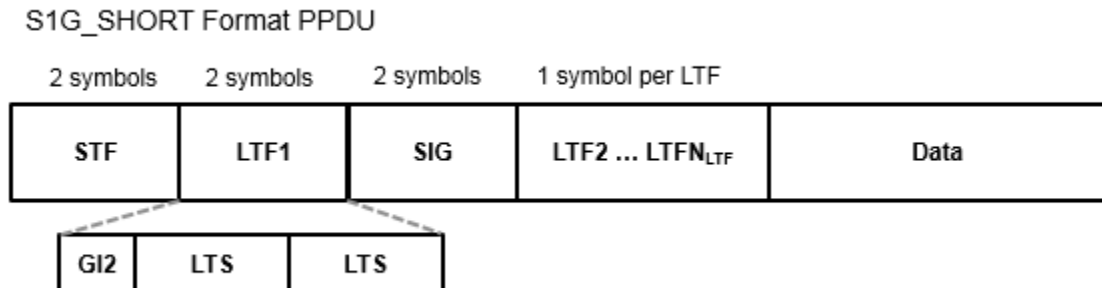
The PPDU consists of these five fields:

- 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. $N_{SYMBOLS} = 1$ 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
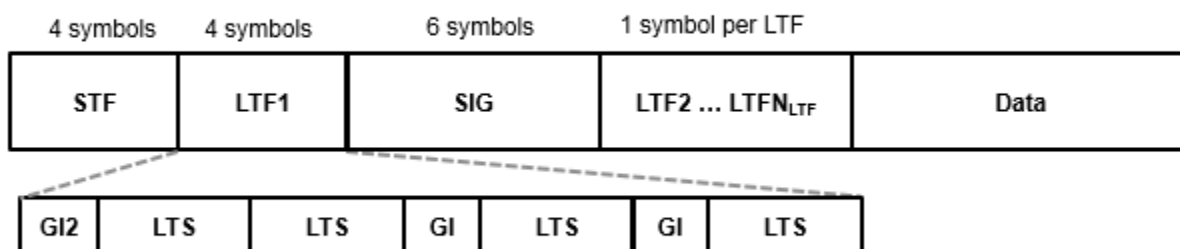
VHT, HT-Mixed, and Non-HT Format PPDU Field Structures

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.
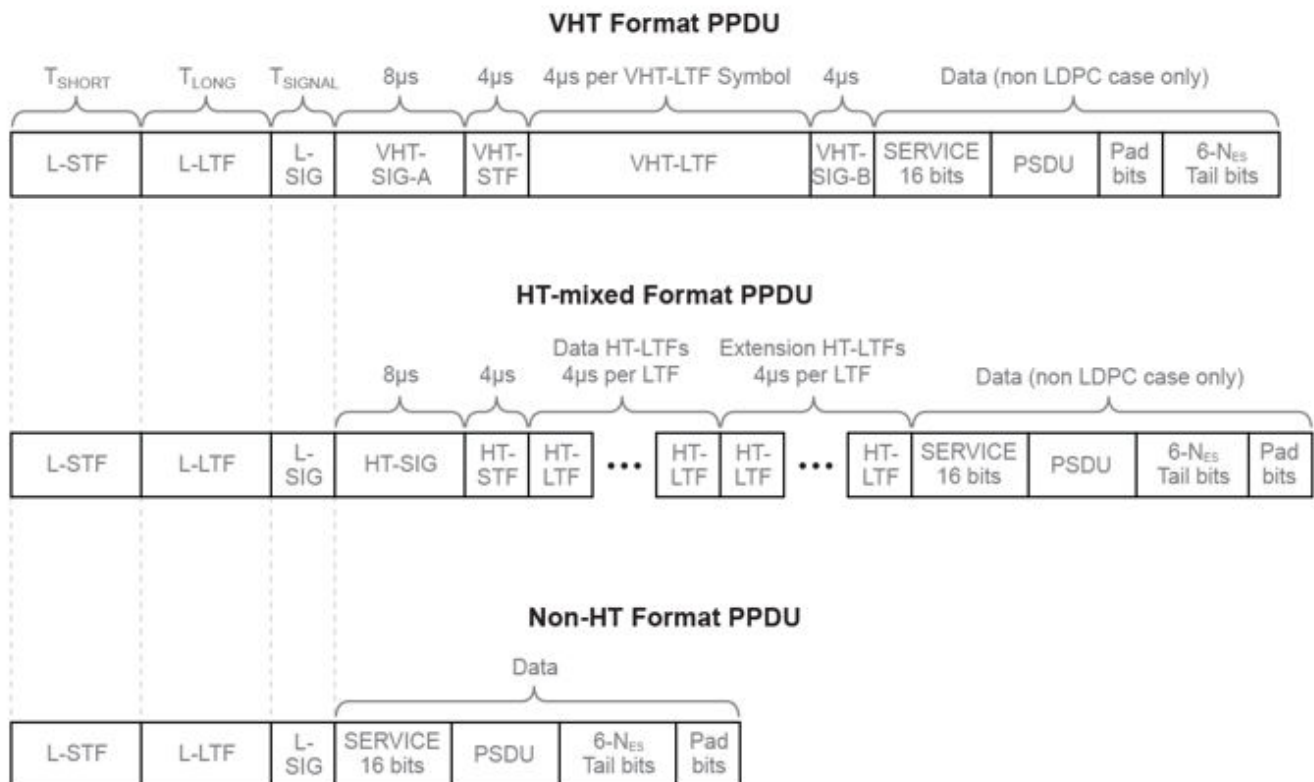
VHT Format PPDU

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.
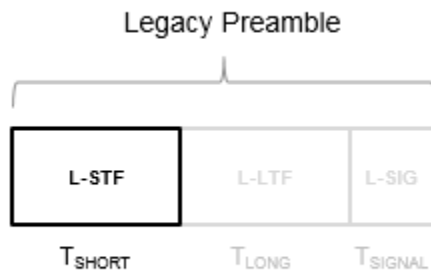
The L-STF duration varies with channel bandwidth.

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}
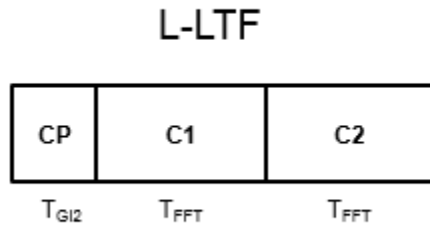
The L-LTF duration varies with channel bandwidth.

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}
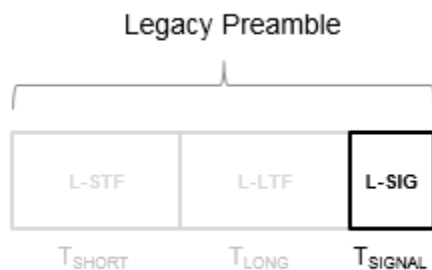
The L-SIG is one OFDM symbol with a duration that varies with channel bandwidth.

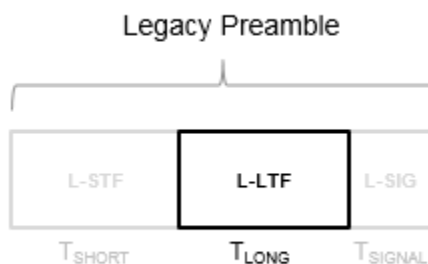| Channel Bandwidth (MHz)	Subcarrier frequency spacing, Δ_F (kHz)	Fast Fourier Transform (FFT) period $(T_{FFT} = 1 / \Delta_F)$	Guard Interval (GI) Duration $(T_{GI} = T_{FFT} / 4)$	L-SIG duration $(T_{SIGNAL} = T_{GI} + T_{FFT})$
20, 40, 80, and 160	312.5	3.2 µs	0.8 µs	4 µs
10	156.25	6.4 µs	1.6 µs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

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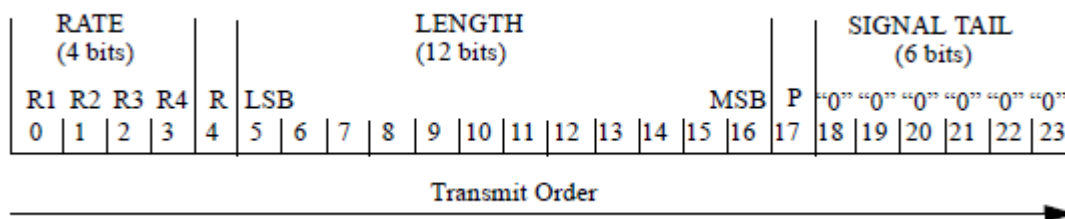
- Bits 0 through 3 specify the data rate (modulation and coding rate) for the non-HT format.

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.
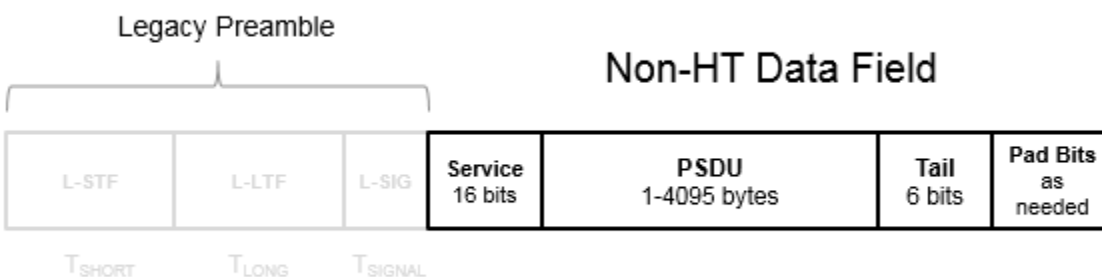
- **Service field** — Contains 16 zeros to initialize the data scrambler.
- **PSDU** — Variable-length field containing the PLCP service data unit (PSDU).
- **Tail** — Tail bits required to terminate a convolutional code. The field uses six zeros for the single encoding stream.
- **Pad Bits** — Variable-length 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$_2$.
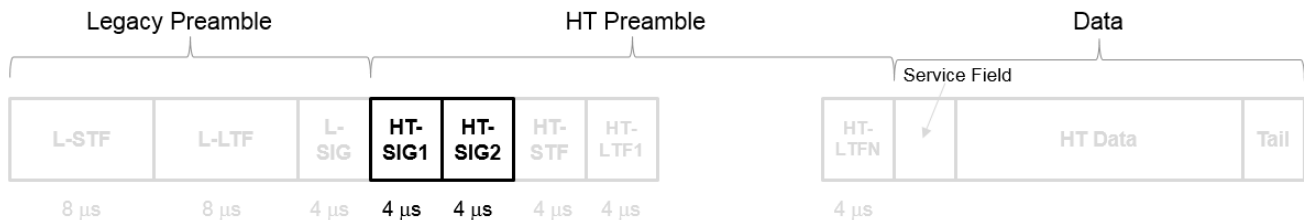
HT-SIG carries information used to decode the HT packet, including the MCS, packet length, FEC coding type, guard interval, number of extension spatial streams, and whether there is payload aggregation. The HT-SIG symbols are also used for auto-detection between HT-mixed format and legacy OFDM packets.
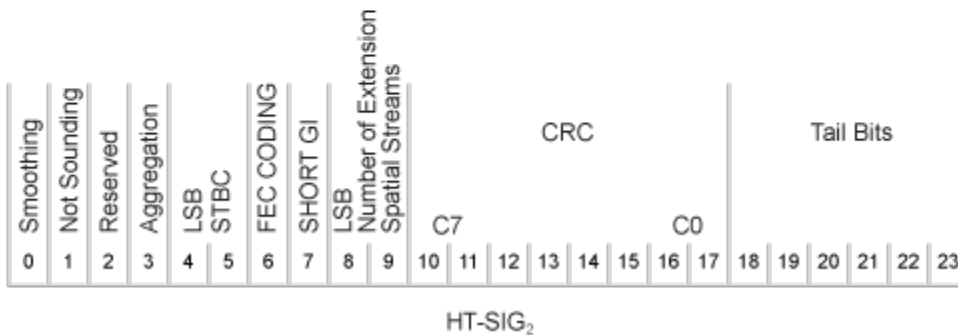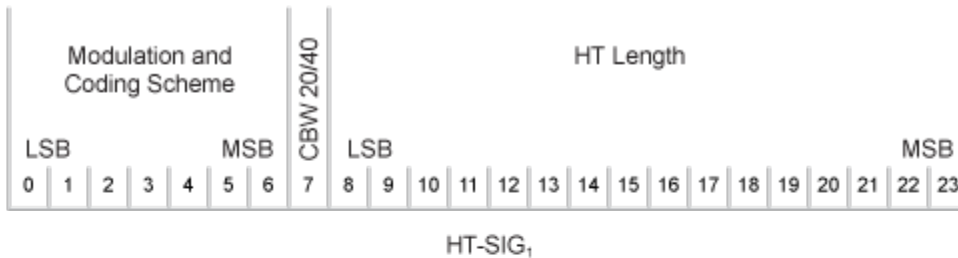
HT-SIG$_1$

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			
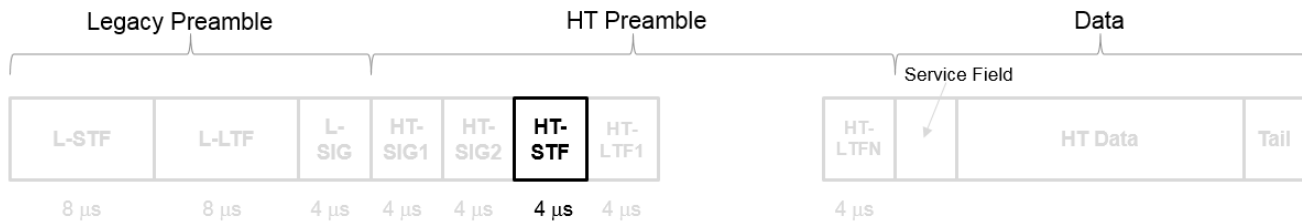
High Throughput Long Training Fields

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**			
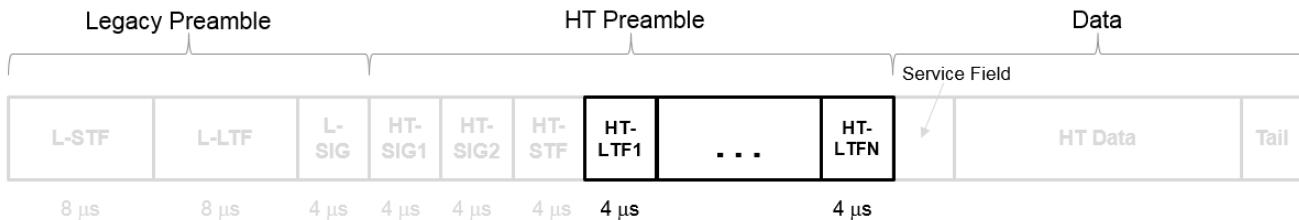
As described in Section 19.3.9.4.6 of IEEE Std 802.11-2016, the receiver can use the HT-LTF to estimate the MIMO channel between the set of QAM mapper outputs (or, if STBC is applied, the STBC encoder outputs) and the receive chains. The HT-LTF portion has one or two parts. The first part consists of 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 high throughput data field (HT-Data) follows the last HT-LTF of an HT-mixed packet.
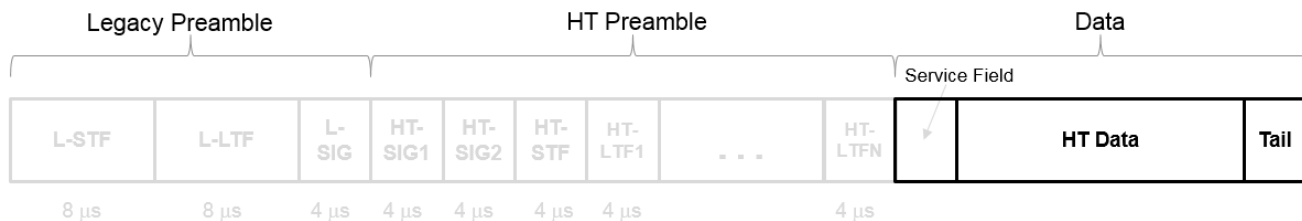
The high throughput data field is used to transmit one or more frames from the MAC layer and consists of four subfields.

- **Service field** — Contains 16 zeros to initialize the data scrambler.
- **PSDU** — Variable-length field containing the PLCP service data unit (PSDU). In 802.11, the PSDU can consist of an aggregate of several MAC service data units.

- **Tail** — Tail bits required to terminate a convolutional code. The field uses six zeros for each encoding stream.

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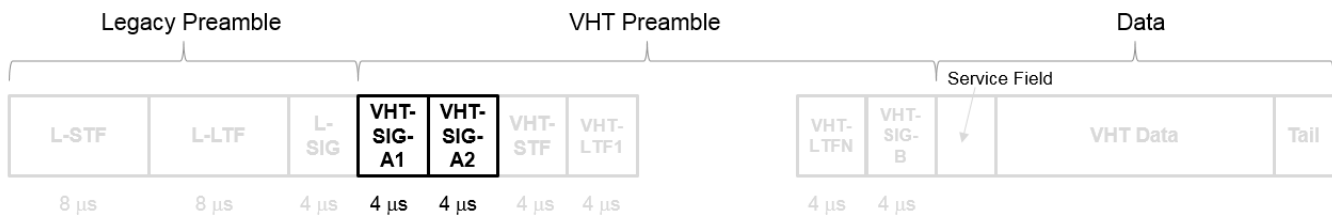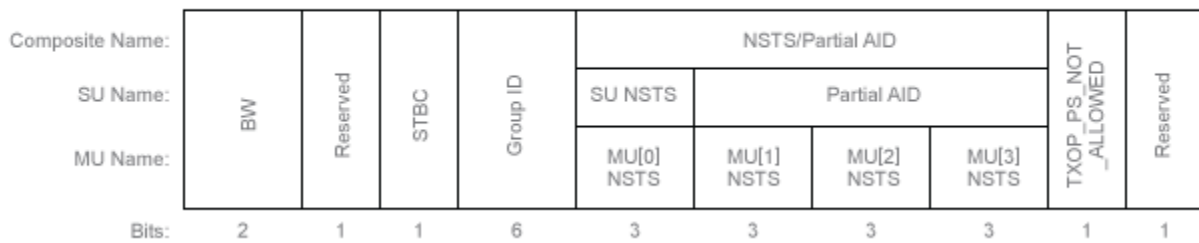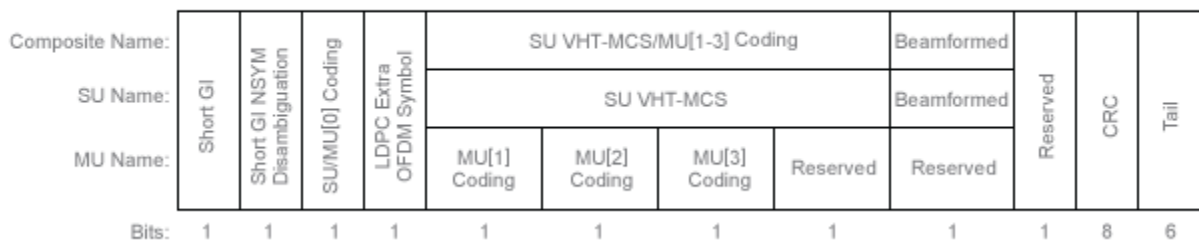
The VHT-SIG-A field includes these components. The bit field structures for VHT-SIG-A1 and VHT-SIG-A2 vary for single user or multiuser transmissions.

- **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.
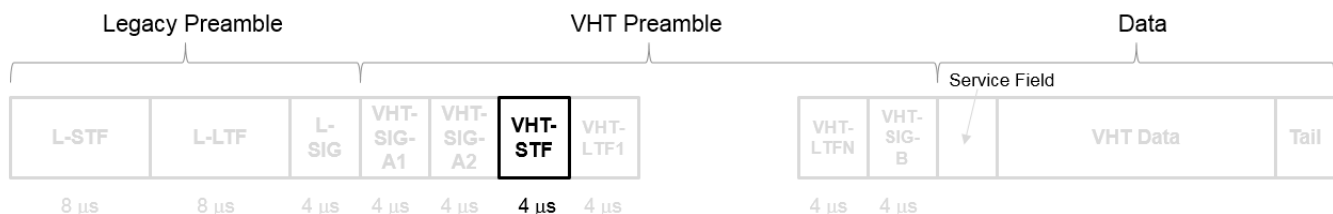
The frequency domain sequence used to construct the VHT-STF for a 20 MHz transmission is identical to the L-STF sequence. Duplicate L-STF sequences are frequency shifted and phase rotated to support VHT transmissions for the 40 MHz, 80 MHz, and 160 MHz channel bandwidths. As such, the L-STF and HT-STF 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.
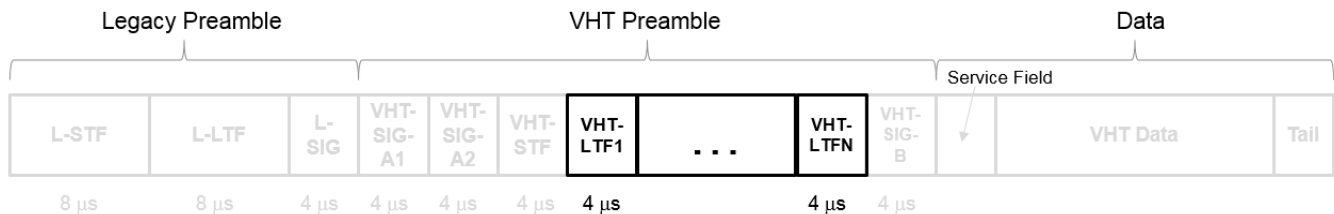
It is used for MIMO channel estimation and pilot subcarrier tracking. The VHT-LTF includes one VHT long training symbol for each spatial stream indicated by the selected MCS. Each symbol is 4 µs long. A maximum of eight symbols are permitted in the VHT-LTF.

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.
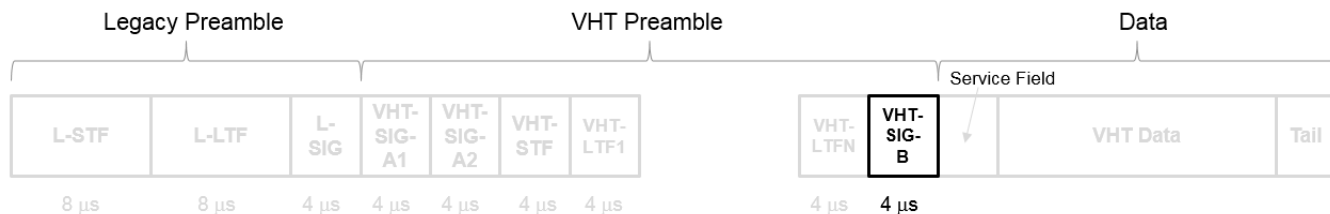
The very high throughput signal B (VHT-SIG-B) field contains the actual rate and A-MPDU length value per user. For a detailed description of the VHT-SIG-B field, see section 21.3.8.3.6 of IEEE Std 802.11-2016. The number of bits in the VHT-SIG-B field varies with the channel bandwidth and the assignment 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 very high throughput data (VHT data) field is used to transmit one or more frames from the MAC layer. It follows the VHT-SIG-B field in the packet structure for the VHT format PPDUs.
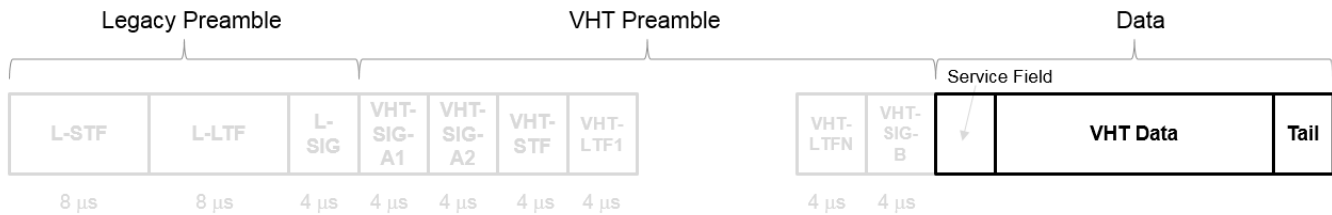
For a detailed description of the VHT Data field, see section 21.3.10 of IEEE Std 802.11-2016. The VHT Data field consists of four subfields.
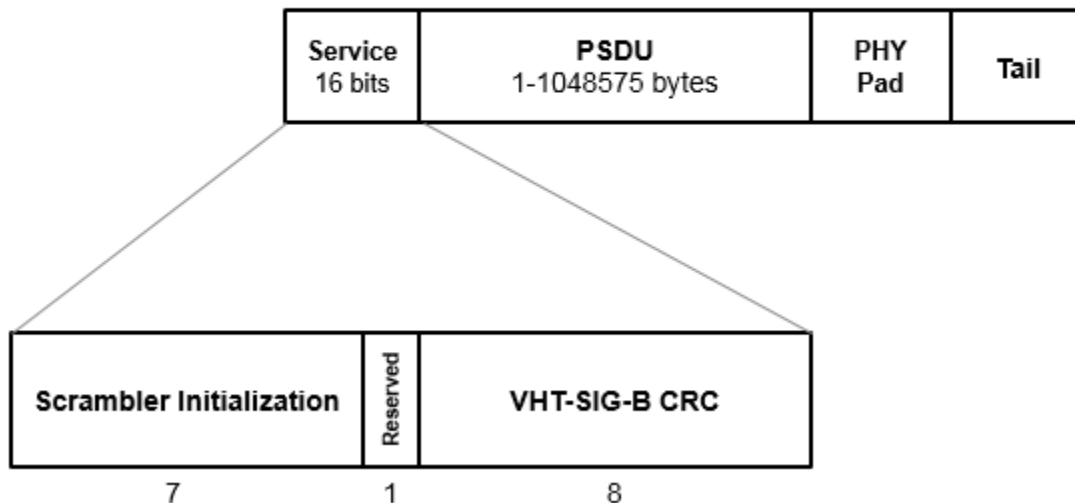
- **Service field** — Contains a seven-bit scrambler initialization state, one bit reserved for future considerations, and eight bits for the VHT-SIG-B CRC field.
- **PSDU** — Variable-length field containing the PLCP service data unit. In 802.11, the PSDU can consist of an aggregate of several MAC service data units.
- **PHY Pad** — Variable number of bits passed to the transmitter to create a complete OFDM symbol.
- **Tail** — Bits used to terminate a convolutional code. Tail bits are not needed when LDPC is used.

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" | "What Is WLAN?" | "Mapping 802.11 Standards to WLAN Toolbox Configuration Objects" on page 1-30 | "Multiuser HE Transmission" on page 1-23

Multiuser HE Transmission

Transmission Mode Options

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

- Orthogonal frequency-division multiple access (OFDMA)
- Full-band multiuser 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 scalar
  - 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
  - An 8-bit binary sequences specified as a string array
  - An 8-bit binary sequences specified as a cell array of character vectors

An RU is a group of 26, 52, 106, 242, 484, 996, or 2×996 subcarriers defining an allocation unit in time and frequency.
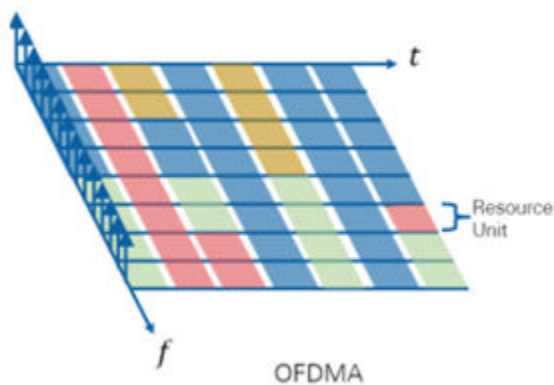
The values specified in the `AllocationIndex` property correspond to the 8-bit indices for each 20 MHz subchannel in the first column of Table 27-24 in [1]. The allocation indices define the number of RUs, RU sizes, and number of users assigned to each RU. When SIGB compression is enabled, the number 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
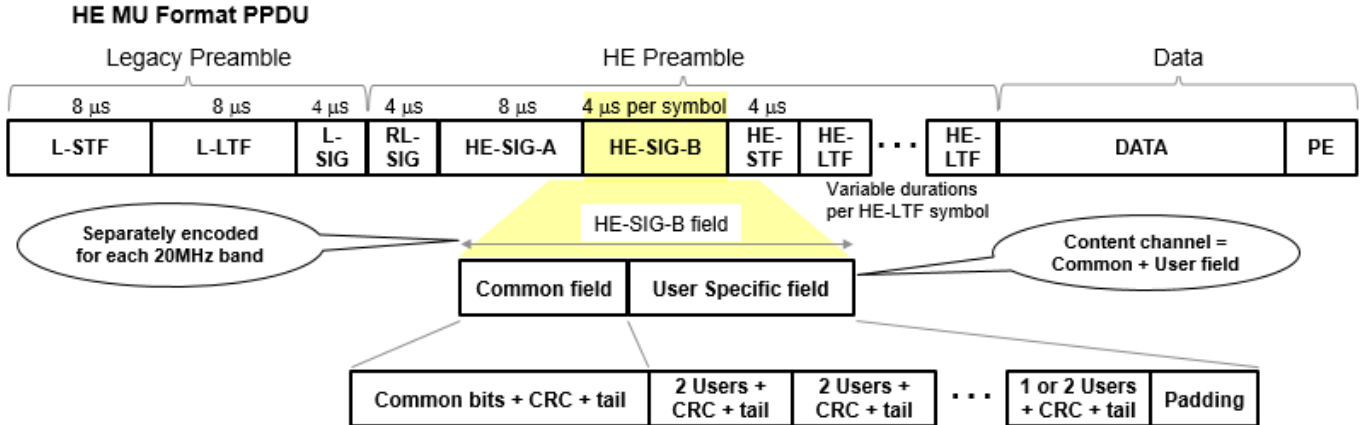
The format of the common field is defined in Table 27-23 of [1]. The RU Allocation subfield in the common field of HE-SIG-B consists of 8 bits that indicate this information for each 20 MHz PPDU bandwidth.

- 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([x 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, 192].	`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 1-3

Mapping 802.11 Standards to WLAN Toolbox Configuration Objects

The table shows the mapping from 802.11 versions to the associated packet format and WLAN Toolbox configuration object.

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	wlanHESUConfig
		wlanHEMUConfig

WLAN Toolbox configuration objects define the properties that enable creation of PPDUs and waveforms for the specified 802.11 transmission format.

See Also

"WLAN Parameterization" on page 1-2 | "WLAN PPDU Structure" on page 1-3 | "Create Configuration Objects" | "Waveform Generation" | "Multiuser HE Transmission" on page 1-23

What is C Code Generation from MATLAB?

You can use WLAN Toolbox together with MATLAB® Coder™ to:

- Create a MEX file to speed up your MATLAB application.
- Generate ANSI®/ISO® compliant C/C++ source code that implements your MATLAB functions and models.
- Generate a standalone executable that runs independently of MATLAB on your computer or another platform.

In general, the code you generate using the toolbox is portable ANSI C code. In order to use code generation, you need a MATLAB Coder license. For more information, see "Get Started with MATLAB Coder" (MATLAB Coder).

Using MATLAB Coder

Creating a MATLAB Coder MEX file can substantially accelerate your MATLAB code. It is also a convenient first step in a workflow that ultimately leads to completely standalone code. When you create a MEX file, it runs in the MATLAB environment. Its inputs and outputs are available for inspection just like any other MATLAB variable. You can then use MATLAB tools for visualization, verification, and analysis.

The simplest way to generate MEX files from your MATLAB code is by using the `codegen` function at the command line. For example, if you have an existing function, `myfunction.m`, you can type the commands at the command line to compile and run the MEX function. `codegen` adds a platform-specific extension to this name. In this case, the `"mex"` suffix is added.

```
codegen myfunction.m
myfunction_mex;
```

Within your code, you can run specific commands either as generated C code or by using the MATLAB engine. In cases where an isolated command does not yet have code generation support, you can use the `coder.extrinsic` command to embed the command in your code. This means that the generated code reenters the MATLAB environment when it needs to run that particular command. This is also useful if you want to embed commands that cannot generate code (such as plotting functions).

To generate standalone executables that run independently of the MATLAB environment, create a MATLAB Coder project inside the MATLAB Coder Integrated Development Environment (IDE). Alternatively, you can call the `codegen` command in the command line environment with appropriate configuration parameters. A standalone executable requires you to write your own `main.c` or `main.cpp` function. See "Generating Standalone C/C++ Executables from MATLAB Code" (MATLAB Coder) for more information.

C/C++ Compiler Setup

Before using `codegen` to compile your code, you must set up your C/C++ compiler. For 32-bit Windows platforms, MathWorks® supplies a default compiler with MATLAB. If your installation does not include a default compiler, you can supply your own compiler. For the current list of supported compilers, see Supported and Compatible Compilers on the MathWorks website. Install a compiler that is suitable for your platform, then read "Setting Up the C or C++ Compiler" (MATLAB Coder).

After installation, at the MATLAB command prompt, run `mex -setup`. You can then use the `codegen` function to compile your code.

Functions and System Objects That Support Code Generation

All WLAN Toolbox functions and System objects support code generation.

See Also

Functions
codegen | mex

More About

• "Code Generation Workflow" (MATLAB Coder)
• Generate C Code from MATLAB Code Video

Functions and System Objects Supported for MATLAB Coder

You can generate efficient C/C++ code for all WLAN Toolbox functions and System objects by using the MATLAB Coder product (requires a license).

An asterisk (*) indicates that the reference page has usage notes and limitations for C/C++ code generation.

comm.ConstellationDiagram*	Display constellation diagram for input signals
comm.ErrorRate*	Compute bit or symbol error rate of input data
comm.EVM*	Measure error vector magnitude
comm.PhaseFrequencyOffset*	Apply phase and frequency offsets to input signal
displayIEs	Display the list of information elements (IEs)
dsp.ArrayPlot*	Display vectors or arrays
dsp.SpectrumAnalyzer*	Display frequency spectrum of time-domain signals
dsp.TimeScope*	Time domain signal display and measurement
getPSDULength	Return HE format PSDU length
getSIGBLength	Return information relevant to HE-SIG-B field length
getTRSConfiguration	Valid HE TB PHY configuration in response to triggering frame containing TRS Control subfield
info	Return characteristic information about TGay multipath fading channel
interpretHESIGABits	Update recovery configuration object with HE-SIG-A bits
packetFormat	Return WLAN packet format
ruInfo	Return HE format resource unit allocation information
showEnvironment	Display channel environment with D-Rays from ray tracing
wlanAMPDUDeaggregate	Deaggregate A-MPDU and extract MPDUs
wlanAPEPLength	Calculate APEP length in octets
wlanBCCDecode	Convolutionally decode input data
wlanBCCDeinterleave	Deinterleave binary convolutionally interleaved input
wlanBCCEncode	Convolutionally encode binary data
wlanBCCInterleave	Interleave binary convolutionally encoded input
wlanClosestReferenceSymbol	Find closest constellation points
wlanCoarseCFOEstimate	Coarse estimate of carrier frequency offset
wlanConstellationDemap	Constellation demapping

`wlanConstellationMap`	Constellation mapping
`wlanDMGConfig`	Create DMG-format configuration object
`wlanDMGDataBitRecover`	Recover data bits from DMG data field
`wlanDMGHeaderBitRecover`	Recover header bits from DMG header field
`wlanDMGOFDMDemodulate`	Demodulate fields of DMG waveform
`wlanDMGOFDMInfo`	Get OFDM information for DMG transmission
`wlanFieldIndices`	Generate PPDU field indices
`wlanFineCFOEstimate`	Fine estimate of carrier frequency offset
`wlanFormatDetect`	Detect packet format
`wlanGolaySequence`	Generate Golay sequence
`wlanHEDataBitRecover`	Recover data bits from HE-Data field
`wlanHEDemodulate`	Demodulate fields of HE waveform
`wlanHEMUConfig`	Create multiuser high-efficiency-format configuration object
`wlanHEOFDMInfo`	Get OFDM information for HE transmission
`wlanHERecoveryConfig`	Create HE recovery configuration object
`wlanHESIGABitRecover`	Recover information bits in HE-SIG-A field
`wlanHESIGBCommonBitRecover`	Recover common field bits in HE-SIG-B field
`wlanHESIGBUserBitRecover`	Recover user field bits in HE-SIG-B field
`wlanHESUConfig`	Create single-user high-efficiency-format configuration object
`wlanHETBConfig`	Create HE TB configuration object
`wlanHTConfig`	Create HT-format configuration object
`wlanHTData`	Generate HT-Data field waveform
`wlanHTDataRecover`	Recover HT data
`wlanHTLTF`	Generate HT-LTF waveform
`wlanHTLTFChannelEstimate`	Channel estimation using HT-LTF
`wlanHTLTFDemodulate`	Demodulate HT-LTF waveform
`wlanHTOFDMInfo`	Return OFDM information for HT transmission
`wlanHTSIG`	Generate HT-SIG waveform
`wlanHTSIGRecover`	Recover HT-SIG information bits
`wlanHTSTF`	Generate HT-STF waveform

`wlanLLTF`	Generate L-LTF waveform
`wlanLLTFChannelEsti mate`	Channel estimation using L-LTF
`wlanLLTFDemodulate`	Demodulate L-LTF waveform
`wlanLSIG`	Generate L-SIG waveform
`wlanLSIGBitRecover`	Recover information bits in L-SIG field
`wlanLSIGRecover`	Recover L-SIG information bits
`wlanLSTF`	Generate L-STF waveform
`wlanMACFrame`	Generate WLAN MAC frame (MPDU or A-MPDU)
`wlanMACFrameConfig`	Create WLAN MAC frame configuration object
`wlanMACManagementCo nfig`	Create WLAN MAC management frame-body configuration object
`wlanMPDUDecode`	Decode MPDU
`wlanMSDULengths`	Calculate MSDU lengths
`wlanNonHTConfig`	Create non-HT-format configuration object
`wlanNonHTData`	Generate non-HT-Data field waveform
`wlanNonHTDataRecove r`	Recover non-HT data
`wlanNonHTOFDMInfo`	Get OFDM information for non-HT transmission
`wlanPacketDetect`	OFDM packet detection using L-STF
`wlanPSDULength`	Calculate PSDU length in octets
`wlanRecoveryConfig`	Create data recovery configuration object
`wlanReferenceSymbol s`	Find reference symbols of constellation diagram
`wlanS1GConfig`	Create S1G-format configuration object
`wlanS1GDemodulate`	Demodulate fields of S1G waveform
`wlanS1GOFDMInfo`	Get OFDM Information for S1G transmission
`wlanScramble`	Scramble and descramble binary input sequence
`wlanSegmentDeparseB its`	Segment-deparse data bits
`wlanSegmentDeparseS ymbols`	Segment-deparse data subcarriers
`wlanSegmentParseBit s`	Segment-parse data bits
`wlanSegmentParseSym bols`	Segment-parse data subcarriers
`wlanStreamDeparse`	Stream-deparse binary input
`wlanStreamParse`	Stream-parse binary input
`wlanSymbolTimingEst imate`	Fine symbol timing estimate using L-LTF

`wlanTGacChannel*`	Filter signal through 802.11ac multipath fading channel
`wlanTGahChannel*`	Filter signal through 802.11ah multipath fading channel
`wlanTGaxChannel*`	Filter signal through an 802.11ax multipath fading channel
`wlanTGayChannel*`	Filter signal through 802.11ay™ multipath fading channel
`wlanTGnChannel*`	Filter signal through 802.11n multipath fading channel
`wlanURAConfig`	Create antenna array configuration object for 802.11ay channel model
`wlanVHTConfig`	Create VHT-format configuration object
`wlanVHTData`	Generate VHT-Data field
`wlanVHTDataRecover`	Recover VHT data
`wlanVHTLTF`	Generate VHT-LTF waveform
`wlanVHTLTFChannelEstimate`	Channel estimation using VHT-LTF
`wlanVHTLTFDemodulate`	Demodulate VHT-LTF waveform
`wlanVHTOFDMInfo`	Get OFDM information for VHT transmission
`wlanVHTSIGA`	Generate VHT-SIG-A waveform
`wlanVHTSIGARecover`	Recover VHT-SIG-A information bits
`wlanVHTSIGB`	Generate VHT-SIG-B waveform
`wlanVHTSIGBRecover`	Recover VHT-SIG-B information bits
`wlanVHTSTF`	Generate VHT-STF waveform
`wlanWaveformGenerator`	Generate WLAN waveform

See Also

More About

- "What is C Code Generation from MATLAB?" on page 1-31

Build HE PPDU

802.11ax Parameterization for Waveform Generation and Simulation

This example shows how to parameterize and generate different IEEE® 802.11ax™ high efficiency (HE) format packets.

Introduction

IEEE P802.11ax/D4.1 [1] specifies four high efficiency (HE) packet formats:

1 Single-user

2 Extended-range single-user

3 Multi-user

4 Trigger-based

This example shows how packets can be generated for these different formats, and demonstrates some of the key features of the draft standard [1].

HE Single-User Format

An HE single-user (SU) packet is a full-band transmission to a single user. The transmit parameters for the HE SU format are configured using a `wlanHESUConfig` object. The `wlanHESUConfig` object can be configured to operate in extended-range mode. To enable or disable this mode, set the `ExtendedRange` property to `true` or `false`. In this example we create a configuration for an HE SU transmission and configure transmission properties.

```
cfgSU = wlanHESUConfig;
cfgSU.ExtendedRange = false;       % Do not use extended-range format
cfgSU.ChannelBandwidth = 'CBW20'; % Channel bandwidth
cfgSU.APEPLength = 1000;           % Payload length in bytes
cfgSU.MCS = 0;                     % Modulation and coding scheme
cfgSU.ChannelCoding = 'LDPC';      % Channel coding
cfgSU.NumSpaceTimeStreams = 1;     % Number of space-time streams
cfgSU.NumTransmitAntennas = 1;     % Number of transmit antennas
```

A single-user packet can be generated with the waveform generator, `wlanWaveformGenerator`. The `getPSDULength()` method returns the required PSDU length given the transmission configuration. This length is used to create a random PSDU for transmission.

```
psdu = randi([0 1],getPSDULength(cfgSU)*8,1,'int8'); % Random PSDU
txSUWaveform = wlanWaveformGenerator(psdu,cfgSU);     % Create packet
```

HE Extended-Range Single-User Format

An extended-range single-user packet has the same fields as the standard single-user format, but the powers of some fields are boosted, and some fields are repeated to improve performance at low SNRs. An extended-range packet can be configured using a `wlanHESUConfig` object with `ChannelBandwidth` set to `'CBW20'` and `ExtendedRange` set to `true`. An extended-range packet has an option to only transmit in the upper 106-tone resource unit (RU) within the 20 MHz channel, or over the entire bandwidth. This can be configured with the `Upper106ToneRU` property:

```
cfgExtSU = cfgSU;
cfgExtSU.ExtendedRange = true;  % Enable extended-range format
```

```
cfgExtSU.Upper106ToneRU = true; % Use only upper 106-tone RU

% Generate a packet
psdu = randi([0 1],getPSDULength(cfgExtSU)*8,1,'int8'); % Random PSDU
txExtSUWaveform = wlanWaveformGenerator(psdu,cfgExtSU);   % Create packet
```

If we look at the spectrum of the data portion we can see only the upper half of the channel is used.

```
fs = wlanSampleRate(cfgExtSU); % Get baseband sample rate
spectrumAnalyzer = dsp.SpectrumAnalyzer;
spectrumAnalyzer.SampleRate = fs;
spectrumAnalyzer.Title = 'HE Extended-Range SU with Active Upper 106-Tone RU';
ind = wlanFieldIndices(cfgExtSU);
spectrumAnalyzer(txExtSUWaveform(ind.HEData(1):ind.HEData(2),:));
```
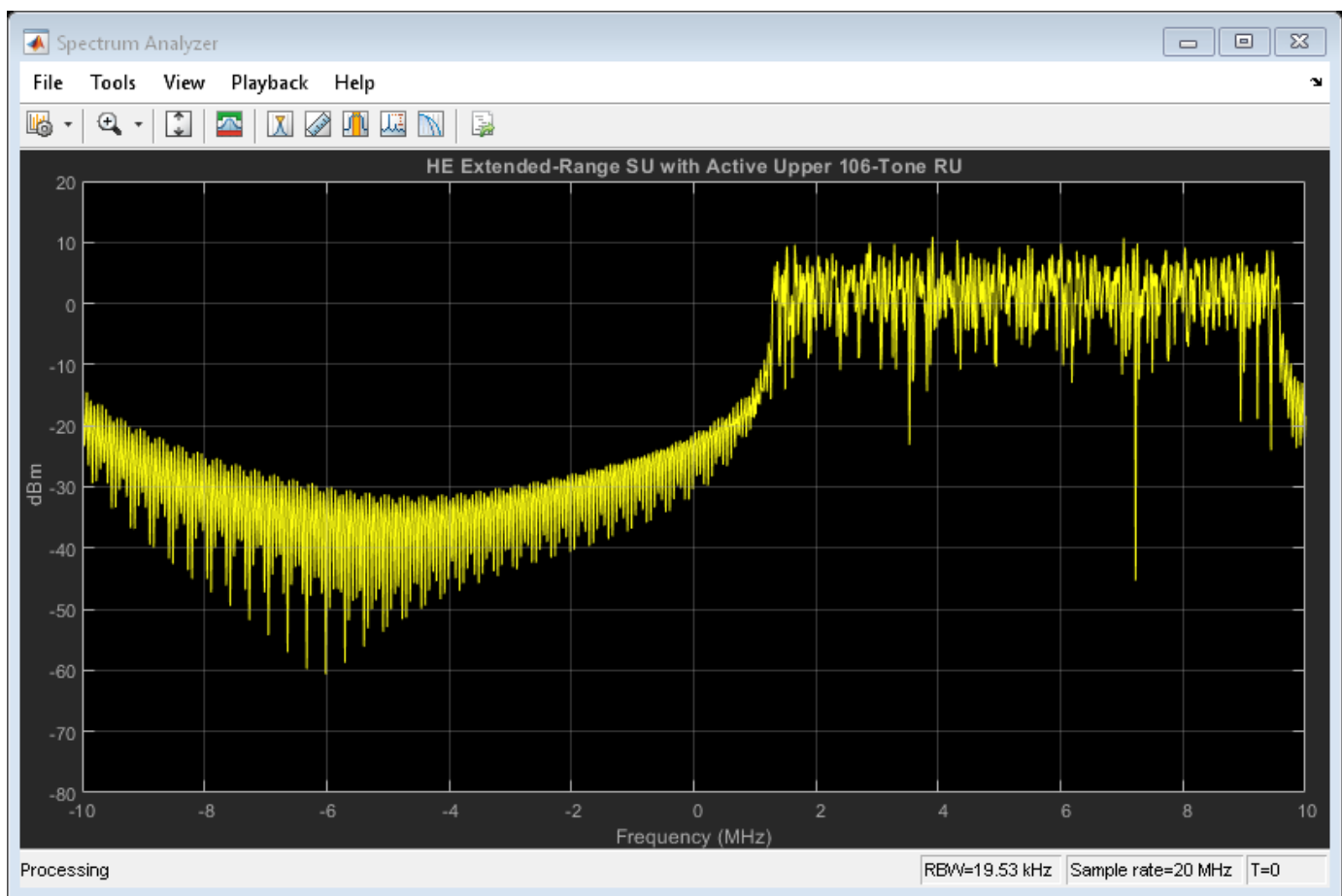

If we compare the power of the L-STF and L-LTF fields we can see the extended-range transmission is boosted by 3 dB.

```
figure;
t = (0:(ind.LLTF(2)-1))/fs*1e6;
plot(t,20*log10(movmean(abs(txSUWaveform(1:ind.LLTF(2))),20)),'-b')
hold on;
plot(t,20*log10(movmean(abs(txExtSUWaveform(1:ind.LLTF(2))),20)),'-r')
grid on;
title('Power of L-STF and L-LTF (1 us Moving Average)');
```

```
xlabel('Time (us)');
ylabel('Power (dBW)');
legend('HE SU','HE Extended-Range SU','Location','SouthWest');
```


Power of L-STF and L-LTF (1 us Moving Average)
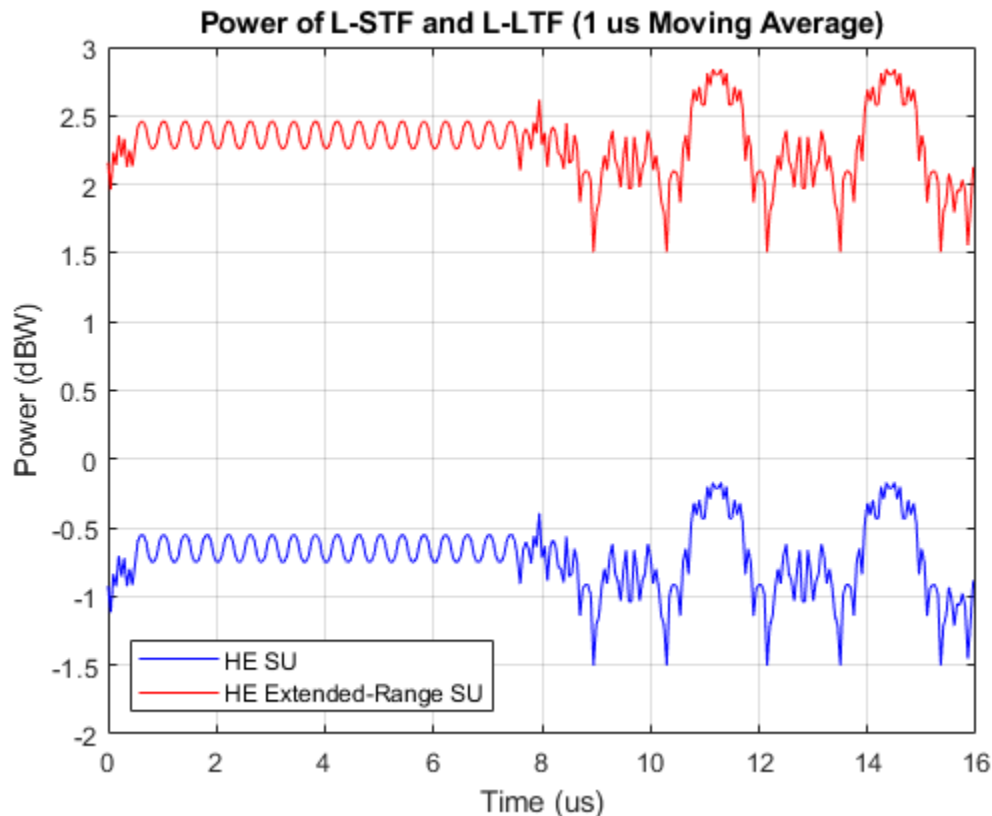
HE Multi-User Format - OFDMA

The HE multi-user (HE MU) format can be configured for an OFDMA transmission, a MU-MIMO transmission, or a combination of the two. This flexibility allows an HE MU packet to transmit to a single user over the whole band, multiple users over different parts of the band (OFDMA), or multiple users over the same part of the band (MU-MIMO).

For an OFDMA transmission, the channel bandwidth is divided into resource units (RUs). An RU is a group of subcarriers assigned to one or more users. An RU is defined by a size (the number of subcarriers) and an index. The RU index specifies the location of the RU within the channel. For example, in an 80 MHz transmission there are four possible 242-tone RUs, one in each 20 MHz subchannel. RU# 242-1 (size 242, index 1) is the RU occupying the lowest absolute frequency within the 80 MHz, and RU# 242-4 (size 242, index 4) is the RU occupying the highest absolute frequency. The draft standard defines possible sizes and locations of RUs in Section 27.3.2.2 of [1].

The assignment of RUs in a transmission is defined by the allocation index. The allocation index is defined in Table 28-24 of [1]. For each 20 MHz subchannel, an 8-bit index describes the number and size of RUs, and the number of users transmitted on each RU. The allocation index also determines which content channel is used to signal a user in HE-SIG-B. The allocation indices within Table 28-24, and the corresponding RU assignments, are provided in the table returned by the function `heRUAllocationTable`. The first 10 allocations within the table are shown below. For each allocation index, the 8-bit allocation index, the number of users, number of RUs, RU indices, RU sizes,

and number of users per RU are displayed. A note is also provided about allocations which are reserved, or serve a special purpose. The allocation table can also be viewed in the Appendix.

```
allocationTable = heRUAllocationTable;
disp('First 10 entries in the allocation table: ')
disp(allocationTable(1:10,:));
```

```
First 10 entries in the allocation table:
    Allocation    BitAllocation    NumUsers    NumRUs    RUIndices       RUSizes        NumUsers
    _____    _____    _____    _____    _____    _____    _____

        0         "00000000"          9          9       {1x9 double}    {1x9 double}    {1x9 dou
        1         "00000001"          8          8       {1x8 double}    {1x8 double}    {1x8 dou
        2         "00000010"          8          8       {1x8 double}    {1x8 double}    {1x8 dou
        3         "00000011"          7          7       {1x7 double}    {1x7 double}    {1x7 dou
        4         "00000100"          8          8       {1x8 double}    {1x8 double}    {1x8 dou
        5         "00000101"          7          7       {1x7 double}    {1x7 double}    {1x7 dou
        6         "00000110"          7          7       {1x7 double}    {1x7 double}    {1x7 dou
        7         "00000111"          6          6       {1x6 double}    {1x6 double}    {1x6 dou
        8         "00001000"          8          8       {1x8 double}    {1x8 double}    {1x8 dou
        9         "00001001"          7          7       {1x7 double}    {1x7 double}    {1x7 dou
```

A `wlanHEMUConfig` object is used to configure the transmission of an HE MU packet. The allocation index for each 20 MHz subchannel must be provided when creating an HE MU configuration object, `wlanHEMUConfig`. An integer between 0 and 223, corresponding to the 8-bit number in Table 28-24 of [1], must be provided for each 20 MHz subchannel.

The allocation index can be provided as a decimal or 8-bit binary sequence. In this example, a 20 MHz HE MU configuration is created with 8-bit allocation index "10000000". This is equivalent to the decimal allocation index 128. This configuration specifies 3 RUs, each with one user.

```
allocationIndex = "10000000"; % 3 RUs, 1 user per RU
cfgMU = wlanHEMUConfig(allocationIndex);
```

The `showAllocation` method visualizes the occupied RUs and subcarriers for the specified configuration. The colored blocks illustrate the occupied subcarriers in the pre-HE and HE portions of the packet. White indicates subcarriers are unoccupied. The pre-HE portion illustrates the occupied subcarriers in the fields preceding HE-STF. The HE portion illustrates the occupied subcarriers in the HE-STF, HE-LTF and HE-Data field and therefore shows the RU allocation. Clicking on an RU will display information about the RU. The RU number corresponds to the i-th RU element of the `cfgMU.RU` property. The size and index are the details of the RU. The RU index is the i-th possible RU of the corresponding RU size within the channel bandwidth, for example Index 2 is the 2nd possible 106-tone RU within the 20 MHz channel bandwidth. The user number corresponds to the i-th User element of the `cfgMU.User` property, and the user field in HE-SIG-B. Note the middle RU (RU #2) is split across the DC subcarriers.

```
showAllocation(cfgMU);
axAlloc = gca; % Get axis handle for subsequent plotting
```
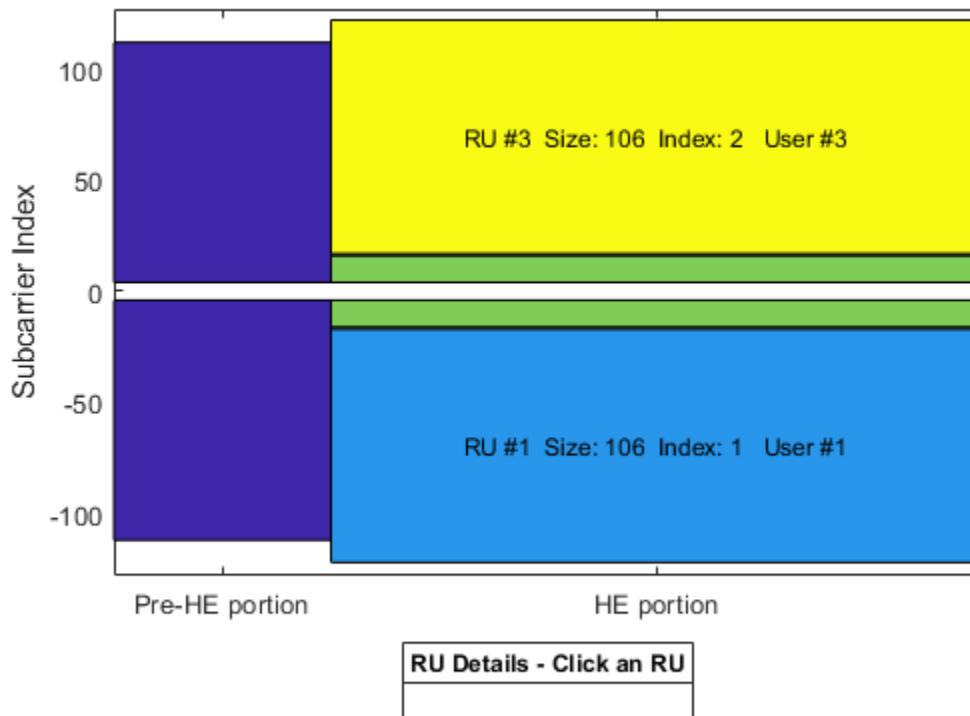
The `ruInfo` method provides details of the RUs in the configuration. In this case we can see three users and three RUs.

```
allocInfo = ruInfo(cfgMU);
disp('Allocation info:')
disp(allocInfo)

Allocation info:
                        NumUsers: 3
                          NumRUs: 3
                       RUIndices: [1 5 2]
                         RUSizes: [106 26 106]
                  NumUsersPerRU: [1 1 1]
       NumSpaceTimeStreamsPerRU: [1 1 1]
          PowerBoostFactorPerRU: [1 1 1]
                      RUNumbers: [1 2 3]
```

The properties of `cfgMU` describe the transmission configuration. The `cfgMU.RU` and `cfgMU.User` properties of `cfgMU` are cell arrays. Each element of the cell arrays contains an object which configures an RU or a User. When the `cfgMU` object is created, the elements of `cfgMU.RU` and `cfgMU.User` are configured to create the desired number of RUs and users. Each element of `cfgMU.RU` is a `wlanHEMURU` object describing the configuration of an RU. Similarly, each element of `cfgMU.User` is a `wlanHEMUUser` object describing the configuration of a User. This object hierarchy is shown below:

```
RU{1}: wlanHEMURU
    PowerBoostFactor: 1

  RU{2}: wlanHEMURU
      PowerBoostFactor: 1

    RU{3}: wlanHEMURU
        PowerBoostFactor: 1
          SpatialMapping: 'Direct'

        Read-only properties:
                      Size: 106
                     Index: 2
               UserNumbers: 3
```

```
wlanHEMUConfig
                 RU: {1×3 cell}
               User: {1×3 cell}
  NumTransmitAntennas: 2
               STBC: 0
                ...
```

```
User{1}: wlanHEMUUser
            APEPLength: 100

  User{2}: wlanHEMUUser
              APEPLength: 100

    User{3}: wlanHEMUUser
                APEPLength: 100
                       MCS: 0
                       ...

        Read-only properties:
                  RUNumber: 3
```

In this example, three RUs are specified by the allocation index 128, therefore `cfgMU.RU` is a cell array with three elements. The index and size of each RU are configured according to the allocation index used to create `cfgMU`. After the object is created, each RU can be configured to create the desired transmission configuration, by setting the properties of the appropriate RU object. For example, the spatial mapping and power boost factor can be configured per RU. The `Size` and `Index` properties of each RU are fixed once the object is created, and therefore are read-only properties. Similarly, the `UserNumbers` property is read-only and indicates which user is transmitted on the RU. For this configuration the first RU is size 106, index 1.

```
disp('First RU configuration:')
disp(cfgMU.RU{1})

First RU configuration:
  wlanHEMURU with properties:

    PowerBoostFactor: 1
      SpatialMapping: 'Direct'

  Read-only properties:
                Size: 106
               Index: 1
         UserNumbers: 1
```

In this example, the allocation index specifies three users in the transmission, therefore `cfgMU.User` contains three elements. The transmission properties of users can be configured by modifying individual user objects, for example the MCS, APEP length and channel coding scheme. The read-only `RUNumber` property indicates which RU is used to transmit this user.

```
disp('First user configuration:')
disp(cfgMU.User{1})

First user configuration:
  wlanHEMUUser with properties:

             APEPLength: 100
                    MCS: 0
     NumSpaceTimeStreams: 1
                    DCM: 0
          ChannelCoding: 'LDPC'
                  STAID: 0
   NominalPacketPadding: 0

  Read-only properties:
               RUNumber: 1
```

The number of users per RU, and mapping of users to RUs is determined by the allocation index. The `UserNumbers` property of an RU object indicates which users (elements of the `cfgMU.User` cell array) are transmitted on that RU. Similarly, the `RUNumber` property of each User object, indicates which RU (element of the `cfgMU.RU` cell array) is used to transmit the user:
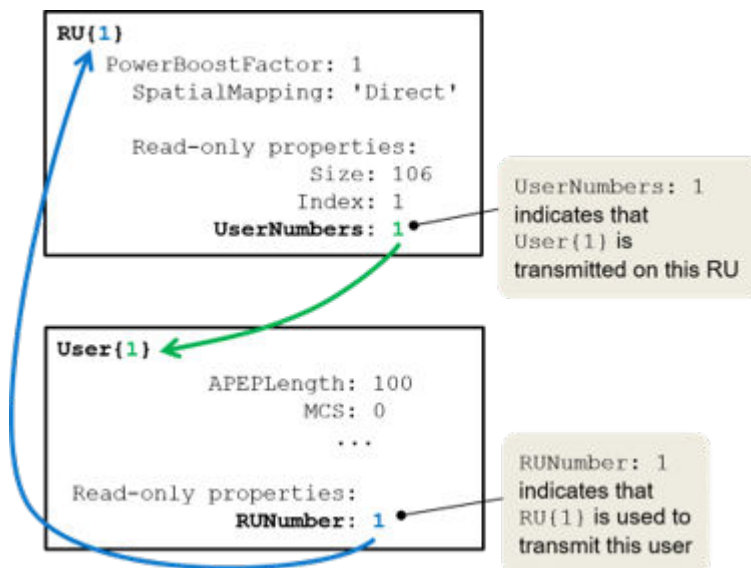
This allows the properties of an RU associated with a User to be accessed easily:

```
ruNum = cfgMU.User{2}.RUNumber; % Get the RU number associated with user 2
disp(cfgMU.RU{ruNum}.SpatialMapping); % Display the spatial mapping

Direct
```

When an RU serves multiple users, in a MU-MIMO configuration, the `UserNumbers` property can index multiple users:
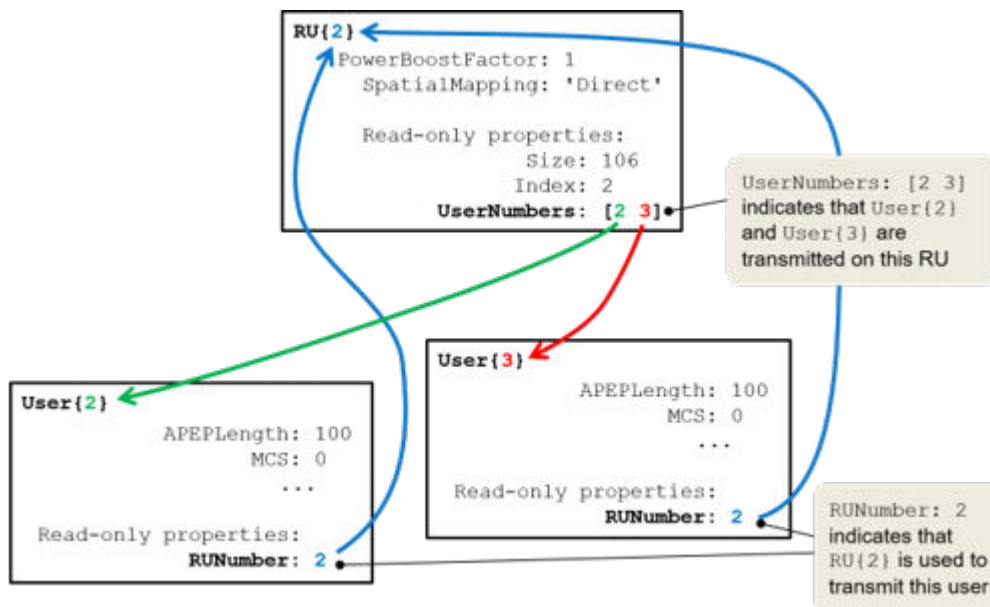
Once the `cfgMU` object is created, transmission parameters can be set as demonstrated below.

```
% Configure RU 1 and user 1
cfgMU.RU{1}.SpatialMapping = 'Direct';
cfgMU.User{1}.APEPLength = 1e3;
cfgMU.User{1}.MCS = 2;
cfgMU.User{1}.NumSpaceTimeStreams = 4;
cfgMU.User{1}.ChannelCoding = 'LDPC';

% Configure RU 2 and user 2
cfgMU.RU{2}.SpatialMapping = 'Fourier';
cfgMU.User{2}.APEPLength = 500;
cfgMU.User{2}.MCS = 3;
cfgMU.User{2}.NumSpaceTimeStreams = 2;
cfgMU.User{2}.ChannelCoding = 'LDPC';

% Configure RU 3 and user 3
cfgMU.RU{3}.SpatialMapping = 'Fourier';
cfgMU.User{3}.APEPLength = 100;
cfgMU.User{3}.MCS = 4;
cfgMU.User{3}.DCM = true;
cfgMU.User{3}.NumSpaceTimeStreams = 1;
cfgMU.User{3}.ChannelCoding = 'BCC';
```

Some transmission parameters are common for all users in the HE MU transmission.

```
% Configure common parameters for all users
cfgMU.NumTransmitAntennas = 4;
cfgMU.SIGBMCS = 2;
```

To generate the HE MU waveform, we first create a random PSDU for each user. A cell array is used to store the PSDU for each user as the PSDU lengths differ. The `getPSDULength()` method returns a vector with the required PSDU per user given the configuration. The waveform generator is then used to create a packet.

```
psduLength = getPSDULength(cfgMU);
psdu = cell(1,allocInfo.NumUsers);
for i = 1:allocInfo.NumUsers
    psdu{i} = randi([0 1],psduLength(i)*8,1,'int8'); % Generate random PSDU
end

% Create MU packet
txMUWaveform = wlanWaveformGenerator(psdu,cfgMU);
```

To configure an OFDMA transmission with a channel bandwidth greater than 20 MHz, an allocation index must be provided for each 20 MHz subchannel. For example, to configure an 80 MHz OFDMA transmission, four allocation indices are required. In this example four 242-tone RUs are configured. The allocation index 192 specifies one 242-tone RU with a single user in a 20 MHz subchannel, therefore the allocation indices [192 192 192 192] are used to create four of these RUs, over 80 MHz:

```
% Display 192 allocation index properties in the table (the 193rd row)
disp('Allocation #192 table entry:')
disp(allocationTable(193,:))

% Create 80 MHz MU configuration, with four 242-tone RUs
cfgMU80MHz = wlanHEMUConfig([192 192 192 192]);
```

```
Allocation #192 table entry:
    Allocation    BitAllocation    NumUsers    NumRUs    RUIndices    RUSizes    NumUsersPerRU
    _____    _____    _____    _____    _____    _____    _____

       192          "11000000"         1          1        {[1]}      {[242]}        {[1]}
```

When multiple 20 MHz subchannels are specified, the ChannelBandwidth property is set to the appropriate value. For this configuration it is set to 'CBW80' as four 20 MHz subchannels are specified. This is also visible in the allocation plot.

```
disp('Channel bandwidth for HE MU allocation:')
disp(cfgMU80MHz.ChannelBandwidth)
showAllocation(cfgMU80MHz,axAlloc)

Channel bandwidth for HE MU allocation:
CBW80
```
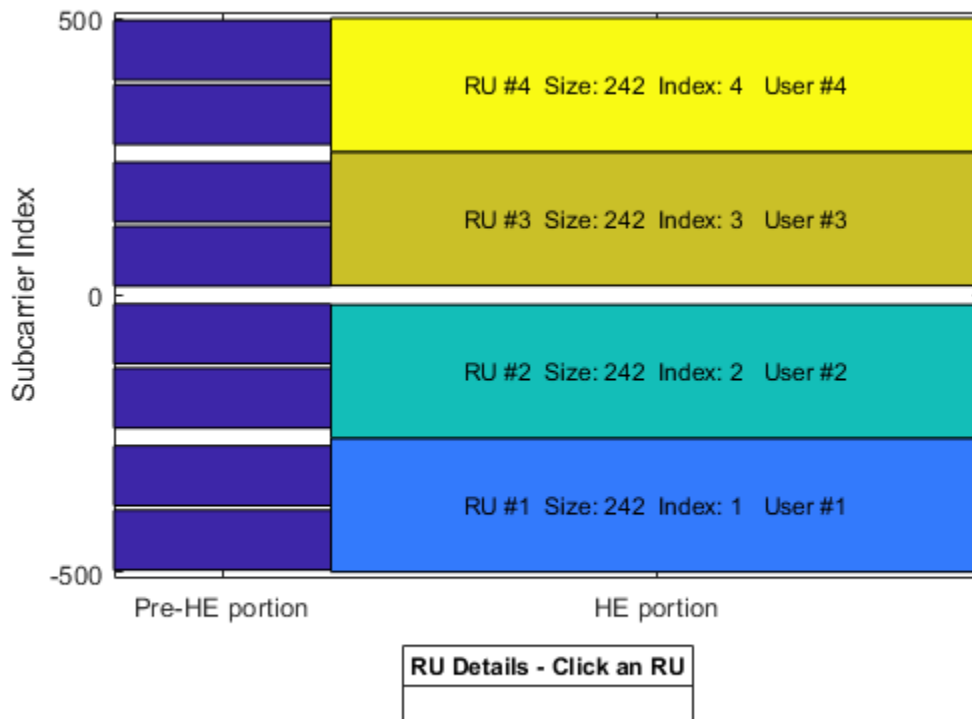
HE Multi-User Format - MU-MIMO

An HE MU packet can also transmit an RU to multiple users using MU-MIMO. For a full band MU-MIMO allocation, the allocation indices between 192 and 199 configure a full-band 20 MHz allocation (242-tone RU). The index within this range determines how many users are configured. The allocation details can be viewed in the allocation table. Note the NumUsers column in the table grows with index but the NumRUs is always 1. The allocation table can also be viewed in the Appendix.

```
disp('Allocation #192-199 table entries:')
disp(allocationTable(193:200,:)) % Indices 192-199 (rows 193 to 200)
```

```
Allocation #192-199 table entries:
    Allocation    BitAllocation    NumUsers    NumRUs    RUIndices    RUSizes     NumUsersPerRU
    _____    _____    _____    _____    _____    _____     _____

       192         "11000000"         1          1         {[1]}      {[242]}        {[1]}
       193         "11000001"         2          1         {[1]}      {[242]}        {[2]}
       194         "11000010"         3          1         {[1]}      {[242]}        {[3]}
       195         "11000011"         4          1         {[1]}      {[242]}        {[4]}
       196         "11000100"         5          1         {[1]}      {[242]}        {[5]}
       197         "11000101"         6          1         {[1]}      {[242]}        {[6]}
       198         "11000110"         7          1         {[1]}      {[242]}        {[7]}
       199         "11000111"         8          1         {[1]}      {[242]}        {[8]}
```

The allocation index 193 transmits a 20 MHz 242-tone RU to two users. In this example, we will create a transmission with a random spatial mapping matrix which maps a single space-time stream for each user, onto two transmit antennas.

```
% Configure 2 users in a 20 MHz channel
cfgMUMIMO = wlanHEMUConfig(193);

% Set the transmission properties of each user
cfgMUMIMO.User{1}.APEPLength = 100; % Bytes
cfgMUMIMO.User{1}.MCS = 2;
cfgMUMIMO.User{1}.ChannelCoding = 'LDPC';
cfgMUMIMO.User{1}.NumSpaceTimeStreams = 1;

cfgMUMIMO.User{2}.APEPLength = 1000; % Bytes
cfgMUMIMO.User{2}.MCS = 6;
cfgMUMIMO.User{2}.ChannelCoding = 'LDPC';
cfgMUMIMO.User{2}.NumSpaceTimeStreams = 1;

% Get the number of occupied subcarriers in the RU
ruIndex = 1; % Get the info for the first (and only) RU
ofdmInfo = wlanHEOFDMInfo('HE-Data',cfgMUMIMO,ruIndex);
numST = ofdmInfo.NumTones; % Number of occupied subcarriers

% Set the number of transmit antennas and generate a random spatial mapping
% matrix
numTx = 2;
allocInfo = ruInfo(cfgMUMIMO);
numSTS = allocInfo.NumSpaceTimeStreamsPerRU(ruIndex);
cfgMUMIMO.NumTransmitAntennas = numTx;
cfgMUMIMO.RU{ruIndex}.SpatialMapping = 'Custom';
cfgMUMIMO.RU{ruIndex}.SpatialMappingMatrix = rand(numST,numSTS,numTx);

% Create packet with a repeated bit sequence as the PSDU
txMUMIMOWaveform = wlanWaveformGenerator([1 0 1 0],cfgMUMIMO);
```

A full band MU-MIMO transmission with a channel bandwidth greater than 20 MHz is created by providing a single RU allocation index within the range 200-223 when creating the wlanHEMUConfig object. For these allocations HE-SIG-B compression is used.

The allocation indices between 200 and 207 configure a full-band MU-MIMO 40 MHz allocation (484-tone RU). The index within this range determines how many users are configured. The allocation details can be viewed in the allocation table. Note the NumUsers column in the table grows with index but the NumRUs is always 1.

```
disp('Allocation #200-207 table entries:')
disp(allocationTable(201:208,:)) % Indices 200-207 (rows 201 to 208)
```

```
Allocation #200-207 table entries:
    Allocation    BitAllocation    NumUsers    NumRUs    RUIndices    RUSizes    NumUsersPerRU
    _____    _____    _____    _____    _____    _____    _____

       200         "11001000"         1          1         {[1]}      {[484]}        {[1]}
       201         "11001001"         2          1         {[1]}      {[484]}        {[2]}
       202         "11001010"         3          1         {[1]}      {[484]}        {[3]}
       203         "11001011"         4          1         {[1]}      {[484]}        {[4]}
       204         "11001100"         5          1         {[1]}      {[484]}        {[5]}
       205         "11001101"         6          1         {[1]}      {[484]}        {[6]}
       206         "11001110"         7          1         {[1]}      {[484]}        {[7]}
```

```
207        "11001111"        8        1        {[1]}        {[484]}        {[8]}
```

Similarly, the allocation indices between 208 and 215 configure a full-band MU-MIMO 80 MHz allocation (996-tone RU), and the allocation indices between 216 and 223 configure a full-band MU-MIMO 160 MHz allocation (2x996-tone RU).

As an example, the allocation index 203 specifies a 484-tone RU with 4 users:

```
cfg484MU = wlanHEMUConfig(203);
showAllocation(cfg484MU,axAlloc)
```
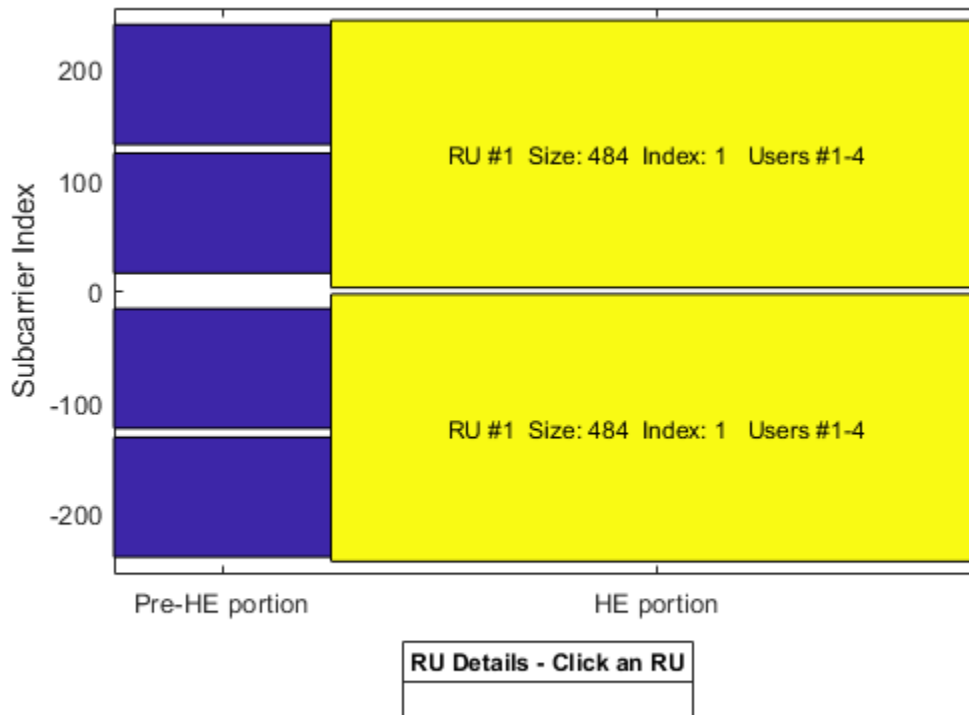

HE Multi-User Format - OFDMA with RU Sizes Greater Than 242 Subcarriers

For an HE MU transmission with a channel bandwidth greater than 20 MHz, two HE-SIG-B content channels are used to signal user configurations. These content channels are duplicated over each 40 MHz subchannel for larger channel bandwidths, as described in Section 27.3.10.8.3 of [1]. When an RU size greater than 242 is specified as part of an OFDMA system, the users assigned to the RU can be signaled on either of the two HE-SIG-B content channels. The allocation index provided when creating an wlanHEMUConfig object controls which content channel each user is signaled on. The allocation table in the Appendix shows the relevant allocation indices.

As an example, consider the following 80 MHz configuration which serves 7 users:

- One 484-tone RU (RU #1) with four users (users #1-4)
- One 242-tone RU (RU #2) with one user (user #5)

- Two 106-tone RUs (RU #3 and #4), each with one user (users #6 and #7)

To configure an 80 MHz OFDMA transmission, four allocation indices are required, one for each 20 MHz subchannel. To configure the above scenario the allocation indices below are used:

```
[X Y 192 96]
```

- X and Y configure the 484-tone RU, with users #1-4. The possible values of X and Y are discussed below.
- 192 configures a 242-tone RU with one user, user #5.
- 96 signals two 106-tone RUs, each with one user, users #6 and #7.

The selection of X and Y configures the appropriate number of users in the 242-tone RU, and determines which HE-SIG-B content channel is used to signal the users. A 484-tone RU spans two 20 MHz subchannels, therefore two allocation indices are required. All seven users from the four RUs will be signaled on the HE-SIG-B content channels, but for now we will only consider the signaling of users on the 484-tone RU. For the 484-tone RU, the four users can be signaled on the two HE-SIG-B content channels in different combinations as shown in Table 1.

Combination	Content Channel 1	Content Channel 2
A	User 1	Users 2-4
B	Users 1-2	Users 3-4
C	Users 1-3	User 4
D	Users 1-4	No users
E	No users	Users 1-4

Table 1

An allocation index within the range 200-207 specifies 1-8 users on a 484-tone RU. To signal no users on a content channel, the allocation index 114 or 115 can be used, for a 448-tone or 996-tone RU. Therefore, the combinations in Table 1 can be defined using two allocation indices as shown in Table 2. The two allocation indices in each row of Table 2 are X and Y.

| Combination	484-Tone RU Allocation Index	Number of Users Per Content Channel	
		Content Channel 1	Content Channel 2
A	[200 202]	1	3
B	[201 201]	2	2
C	[202 200]	3	1
D	[203 114]	4	0
E	[114 203]	0	4

Table 2

Therefore, to configure 'Combination E' the following 80 MHz allocation indices are used:

```
[114 203 192 96]
```

- 114 and 203 configure the 484-tone RU, with users #1-4.
- 192 configures a 242-tone RU with one user, user #5.

- **96** signals two 106-tone RUs, each with one user, users #6 and #7.

```
cfg484OFDMA = wlanHEMUConfig([114 203 192 96]);
showAllocation(cfg484OFDMA,axAlloc);
```
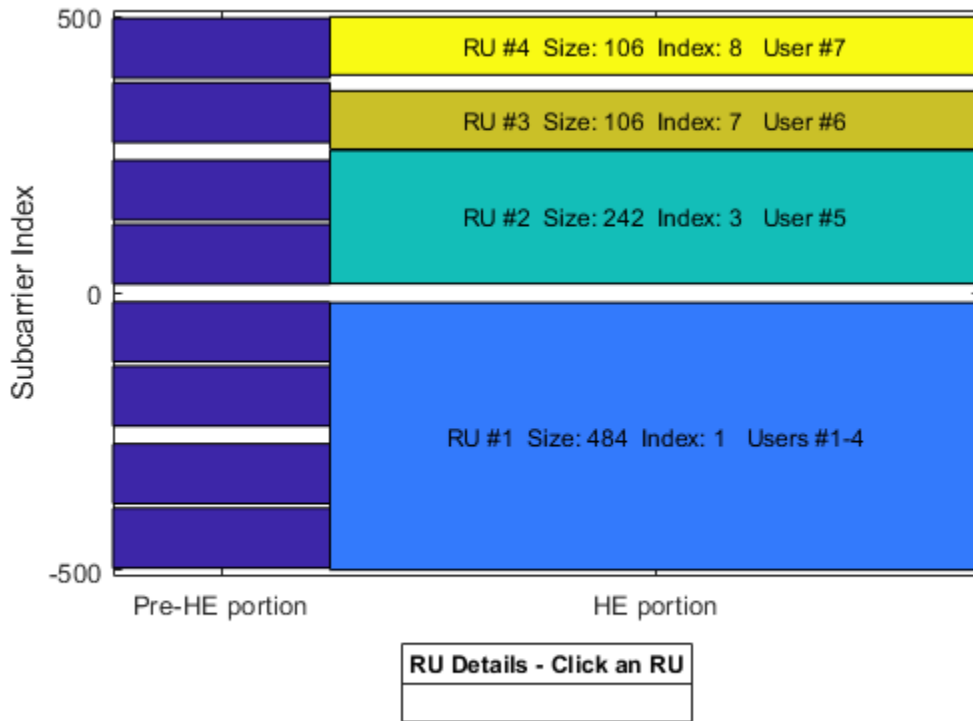

The HE-SIG-B allocation signaling can be viewed with the function hePlotHESIGBAllocationMapping. This shows the user fields signaled on each HE-SIG-B content channel, and which RU and user in the `wlanHEMUConfig` object, each user field signals. In this case we can see the users on RU #1, 3 and 4 are all signaled on content channel 2, and the user of RU #2 is signaled on content channel 1. The second content channel signals six users, while the first content channel only signals one user. Therefore, the first content channel will be padded up to the length of the second for transmission. In the diagram, the RU allocation information is provided in the form index-size, e.g. RU8-106 is the 8th 106-tone RU.

```
figure;
hePlotHESIGBAllocationMapping(cfg484OFDMA);
axSIGB = gca; % Get axis handle for subsequent plotting
```
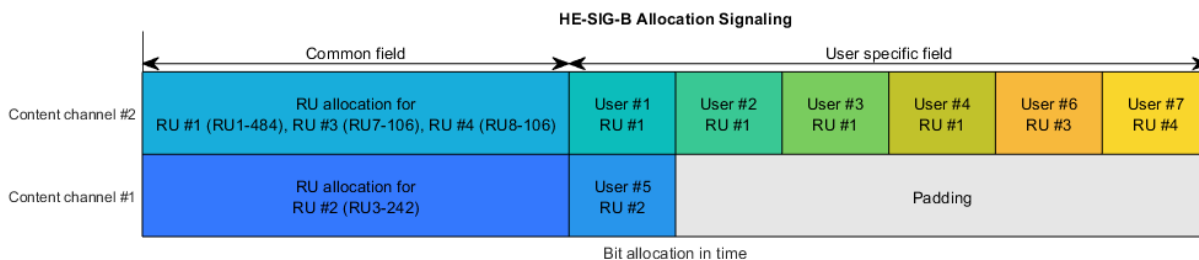
To balance the user field signaling in HE-SIG-B, we can use 'Combination B' in Table 2 when creating the allocation index for the 484-tone RU. This results in two users being signaled on each content channel of HE-SIG-B, creating a better balance of user fields, and potentially fewer HE-SIG-B symbols in the transmission.

```
cfg484OFDMABalanced = wlanHEMUConfig([201 201 96 192]);
hePlotHESIGBAllocationMapping(cfg484OFDMABalanced,axSIGB);
```

HE-SIG-B Allocation Signaling

	Common field		User specific field			
Content channel #2	RU allocation for RU #1 (RU1-484), RU #4 (RU4-242)		User #3 RU #1	User #4 RU #1	User #7 RU #4	Padding
Content channel #1	RU allocation for RU #1 (RU1-484), RU #2 (RU5-106), RU #3 (RU6-106)		User #1 RU #1	User #2 RU #1	User #5 RU #2	User #6 RU #3

Bit allocation in time
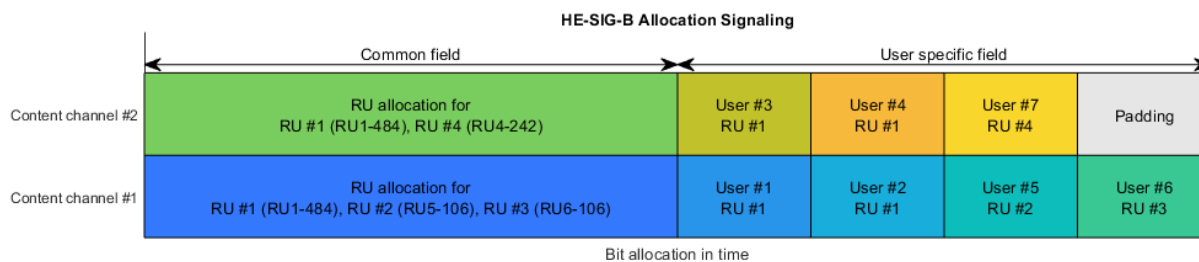
HE Multi-User Format - Central 26-Tone RU

In an 80 MHz transmission, when a full band RU is not used, the central 26-tone RU can be optionally active. The central 26-tone RU is enabled using a name-value pair when creating the `wlanHEMUConfig` object.

```
% Create a configuration with no central 26-tone RU
cfgNoCentral = wlanHEMUConfig([192 192 192 192],'LowerCenter26ToneRU',false);
showAllocation(cfgNoCentral,axAlloc);

% Create a configuration with a central 26-tone RU
cfgCentral = wlanHEMUConfig([192 192 192 192],'LowerCenter26ToneRU',true);
showAllocation(cfgCentral,axAlloc);
```
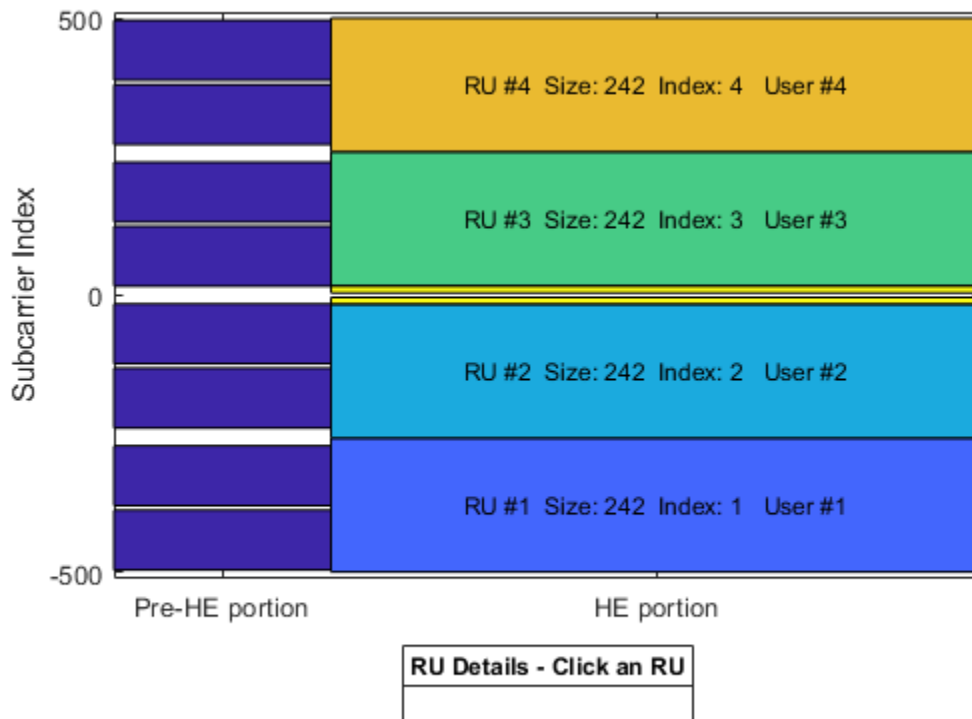
Similarly, for a 160 MHz transmission, the central 26-tone RU in each 80 MHz segment can be optionally used. Each central 26-tone RU can be enabled using name-value pairs when creating the `wlanHEMUConfig` object. In this example only the upper central 26-tone RU is created. Four 242-tone RUs, each with one user are specified with the allocation index [200 114 114 200 200 114 114 200].

```
cfgCentral160MHz = wlanHEMUConfig([200 114 114 200 200 114 114 200],'UpperCenter26ToneRU',true);
disp(cfgCentral160MHz)
```

```
  wlanHEMUConfig with properties:

                     RU: {1x5 cell}
                   User: {1x5 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: 'CBW160'
```

```
        AllocationIndex: [200 114 114 200 200 114 114 200]
    LowerCenter26ToneRU: 0
    UpperCenter26ToneRU: 1
```

HE Multi-User Format - Preamble Puncturing

In an 80 MHz or 160 MHz transmission, 20 MHz subchannels can be punctured to allow a legacy system to operate in the punctured channel. This method is also described as channel bonding. To null a 20 MHz subchannel the 20 MHz subchannel allocation index 113 can be used. The punctured 20 MHz subchannel can be viewed with the showAllocation method.

```
% Null second lowest 20 MHz subchannel in a 160 MHz configuration
cfgNull = wlanHEMUConfig([192 113 114 200 208 115 115 115]);

% Plot the allocation
showAllocation(cfgNull,axAlloc);
```
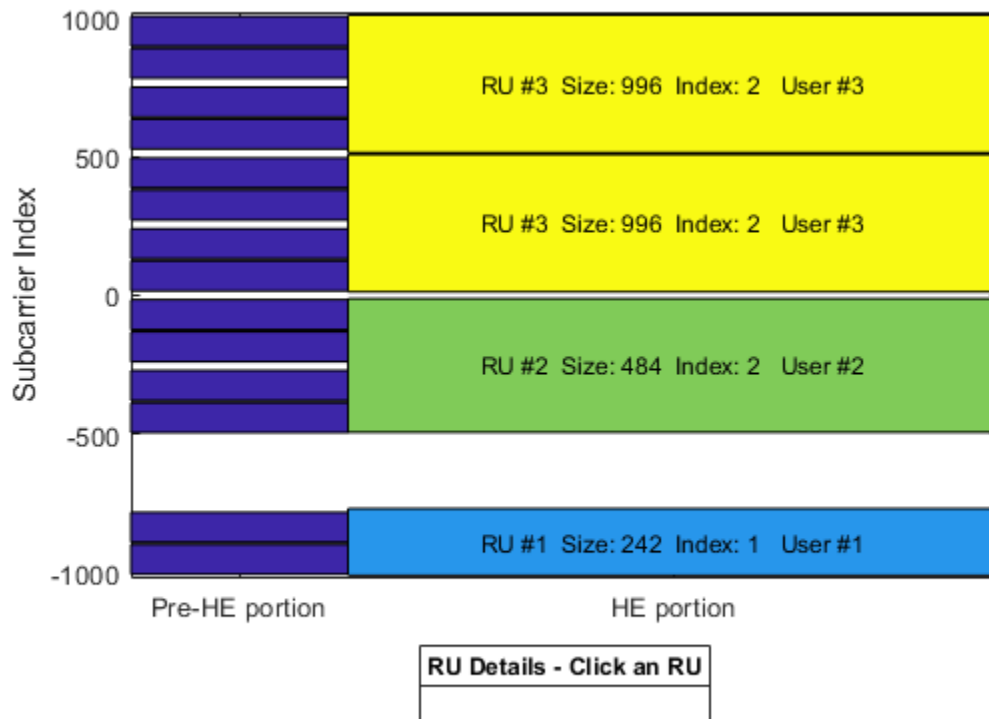

The punctured 20 MHz can also be viewed with the generated waveform and the spectrum analyzer.

```
% Set the transmission properties of each user in all RUs
cfgNull.User{1}.APEPLength = 100;
cfgNull.User{1}.MCS = 2;
cfgNull.User{1}.ChannelCoding = 'LDPC';
cfgNull.User{1}.NumSpaceTimeStreams = 1;

cfgNull.User{2}.APEPLength = 1000;
cfgNull.User{2}.MCS = 6;
```

```
cfgNull.User{2}.ChannelCoding = 'LDPC';
cfgNull.User{2}.NumSpaceTimeStreams = 1;

cfgNull.User{3}.APEPLength = 100;
cfgNull.User{3}.MCS = 1;
cfgNull.User{3}.ChannelCoding = 'LDPC';
cfgNull.User{3}.NumSpaceTimeStreams = 1;

% Create packet
txNullWaveform = wlanWaveformGenerator([1 0 1 0],cfgNull);

spectrumAnalyzer = dsp.SpectrumAnalyzer;
spectrumAnalyzer.SampleRate = wlanSampleRate(cfgNull);
spectrumAnalyzer.Title = '160 MHz HE MU Transmission with Punctured 20 MHz Channel';
spectrumAnalyzer(txNullWaveform);
```
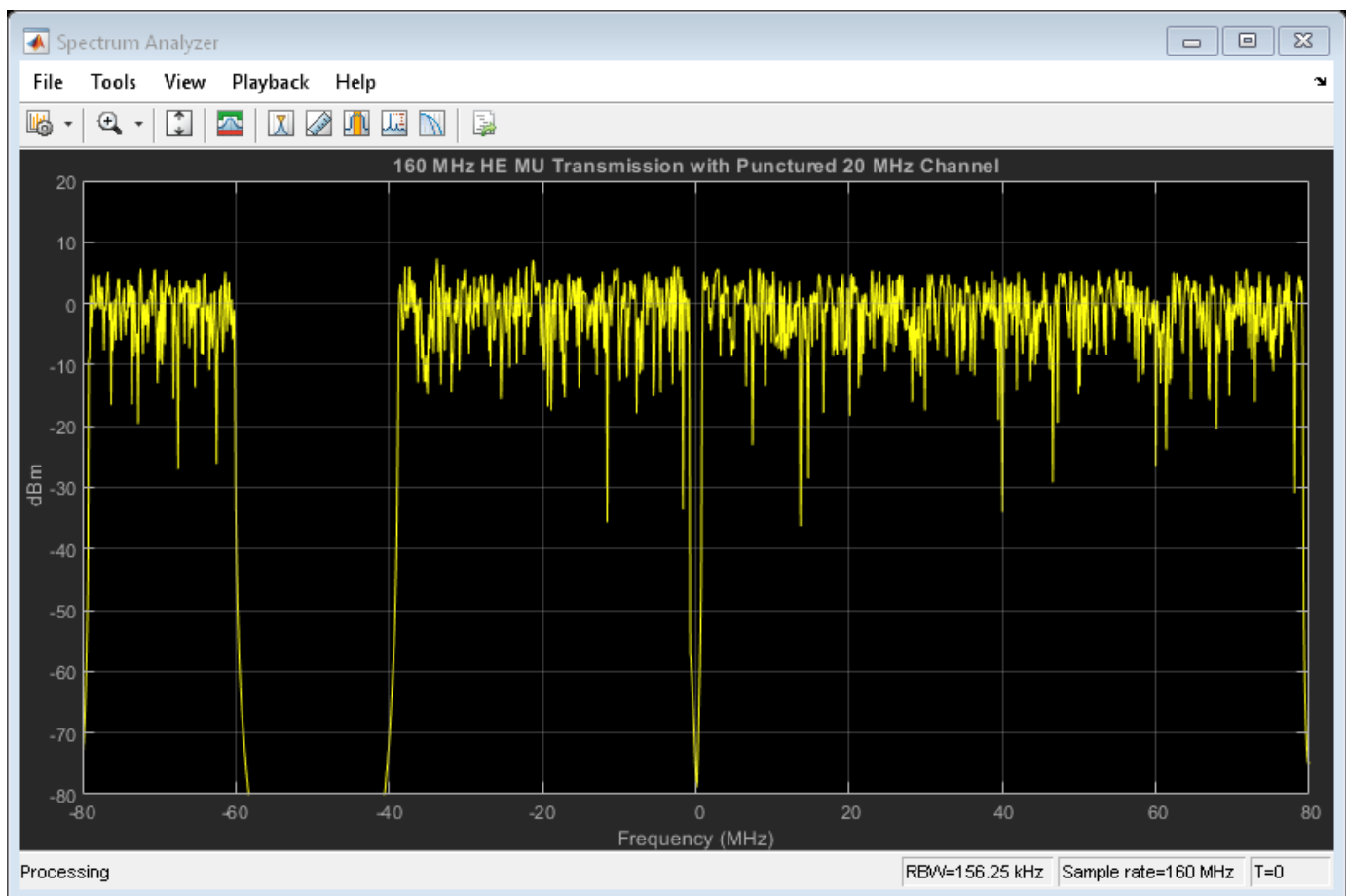

Trigger-Based MU Format

The HE trigger-based (TB) format allows for OFDMA or MU-MIMO transmission in the uplink. Each station (STA) transmits a TB packet simultaneously, when triggered by the access point (AP). A TB transmission is controlled entirely by the AP. All the parameters required for the transmission are provided in a trigger frame to all STAs participating in the TB transmission. In this example a TB transmission in response to a trigger frame for three users in an OFDMA/MU-MIMO system is configured; three STAs will transmit simultaneously to an AP.

The 20 MHz allocation 97 is used which corresponds to two RUs, one of which serves two users in MU-MIMO.

```
disp('Allocation #97 table entry:')
disp(allocationTable(98,:)) % Index 97 (row 98)
```

```
Allocation #97 table entry:
    Allocation    BitAllocation    NumUsers    NumRUs    RUIndices      RUSizes      NumUser
    _____    _____    _____    _____    _____   _____   _____

        97          "01100001"        3          2       {1x2 double}   {1x2 double}   {1x2 dou
```

The allocation information is obtained by creating a MU configuration with `wlanHEMUConfig`.

```
% Generate an OFDMA allocation
cfgMU = wlanHEMUConfig(97);
allocationInfo = ruInfo(cfgMU);
```

In a TB transmission several parameters are the same for all users in the transmission. Some of these are specified below:

```
% These parameters are the same for all users in the OFDMA system
trgMethod = 'TriggerFrame'; % Method used to trigger an HE TB PPDU
channelBandwidth = cfgMU.ChannelBandwidth; % Bandwidth of OFDMA system
lsigLength = 142;          % L-SIG length
preFECPaddingFactor = 2;   % Pre-FEC padding factor
ldpcExtraSymbol = false;   % LDPC extra symbol
numHELTFSymbols = 2;       % Number of HE-LTF symbols
```

A TB transmission for a single user within the system is configured with a `wlanHETBConfig` object. In this example, a cell array of three objects is created to describe the transmission of the three users.

```
% Create a trigger configuration for each user
numUsers = allocationInfo.NumUsers;
cfgTriggerUser = repmat({wlanHETBConfig},1,numUsers);
```

The non-default system-wide properties are set for each user.

```
for userIdx = 1:numUsers
    cfgTriggerUser{userIdx}.TriggerMethod = trgMethod;
    cfgTriggerUser{userIdx}.ChannelBandwidth = channelBandwidth;
    cfgTriggerUser{userIdx}.LSIGLength = lsigLength;
    cfgTriggerUser{userIdx}.PreFECPaddingFactor = preFECPaddingFactor;
    cfgTriggerUser{userIdx}.LDPCExtraSymbol = ldpcExtraSymbol;
    cfgTriggerUser{userIdx}.NumHELTFSymbols = numHELTFSymbols;
end
```

Next the per-user properties are set. When multiple users are transmitting in the same RU, in a MU-MIMO configuration, each user must transmit on different space-time stream indices. The properties `StartingSpaceTimeStream` and `NumSpaceTimeStreamSteams` must be set for each user to make sure different space-time streams are used. In this example user 1 and 2 are in a MU-MIMO configuration, therefore `StartingSpaceTimeStream` for user two is set to 2, as user one is configured to transmit 1 space-time stream with `StartingSpaceTimeStream = 1`.

```
% These parameters are for the first user - RU#1 MU-MIMO user 1
cfgTriggerUser{1}.RUSize = allocationInfo.RUSizes(1);
```

```
cfgTriggerUser{1}.RUIndex = allocationInfo.RUIndices(1);
cfgTriggerUser{1}.MCS = 4;                          % Modulation and coding scheme
cfgTriggerUser{1}.NumSpaceTimeStreams = 1;     % Number of space-time streams
cfgTriggerUser{1}.NumTransmitAntennas = 1;     % Number of transmit antennas
cfgTriggerUser{1}.StartingSpaceTimeStream = 1; % The starting index of the space-time streams
cfgTriggerUser{1}.ChannelCoding = 'LDPC';      % Channel coding

% These parameters are for the second user - RU#1 MU-MIMO user 2
cfgTriggerUser{2}.RUSize = allocationInfo.RUSizes(1);
cfgTriggerUser{2}.RUIndex = allocationInfo.RUIndices(1);
cfgTriggerUser{2}.MCS = 3;                          % Modulation and coding scheme
cfgTriggerUser{2}.NumSpaceTimeStreams = 1;     % Number of space-time streams
cfgTriggerUser{2}.StartingSpaceTimeStream = 2; % The starting index of the space-time streams
cfgTriggerUser{2}.NumTransmitAntennas = 1;     % Number of transmit antennas
cfgTriggerUser{2}.ChannelCoding = 'LDPC';      % Channel coding

% These parameters are for the third user - RU#2
cfgTriggerUser{3}.RUSize = allocationInfo.RUSizes(2);
cfgTriggerUser{3}.RUIndex = allocationInfo.RUIndices(2);
cfgTriggerUser{3}.MCS = 4;                          % Modulation and coding scheme
cfgTriggerUser{3}.NumSpaceTimeStreams = 2;     % Number of space-time streams
cfgTriggerUser{3}.StartingSpaceTimeStream = 1; % The starting index of the space-time streams
cfgTriggerUser{3}.NumTransmitAntennas = 2;     % Number of transmit antennas
cfgTriggerUser{3}.ChannelCoding = 'BCC';       % Channel coding
```

A packet containing random data is now transmitted by each user with `wlanWaveformGenerator`. The waveform transmitted by each user is stored for analysis.

```
trigInd = wlanFieldIndices(cfgTriggerUser{1}); % Get the indices of each field
txTrigStore = zeros(trigInd.HEData(2),numUsers);
for userIdx = 1:numUsers
    % Generate waveform for a user
    cfgTrigger = cfgTriggerUser{userIdx};
    txPSDU = randi([0 1],getPSDULength(cfgTrigger)*8,1);
    txTrig = wlanWaveformGenerator(txPSDU,cfgTrigger);

    % Store the transmitted STA waveform for analysis
    txTrigStore(:,userIdx) = sum(txTrig,2);
end
```

The spectrum of the transmitted waveform from each STA shows the different portions of the spectrum used, and the overlap in the MU-MIMO RU.

```
spectrumAnalyzer = dsp.SpectrumAnalyzer;
spectrumAnalyzer.SampleRate = wlanSampleRate(cfgTriggerUser{1});
spectrumAnalyzer.ChannelNames = {'RU#1 User 1','RU#1 User 2','RU#2'};
spectrumAnalyzer.ShowLegend = true;
spectrumAnalyzer.Title = 'Transmitted HE TB Waveform per User';
spectrumAnalyzer(txTrigStore);
```
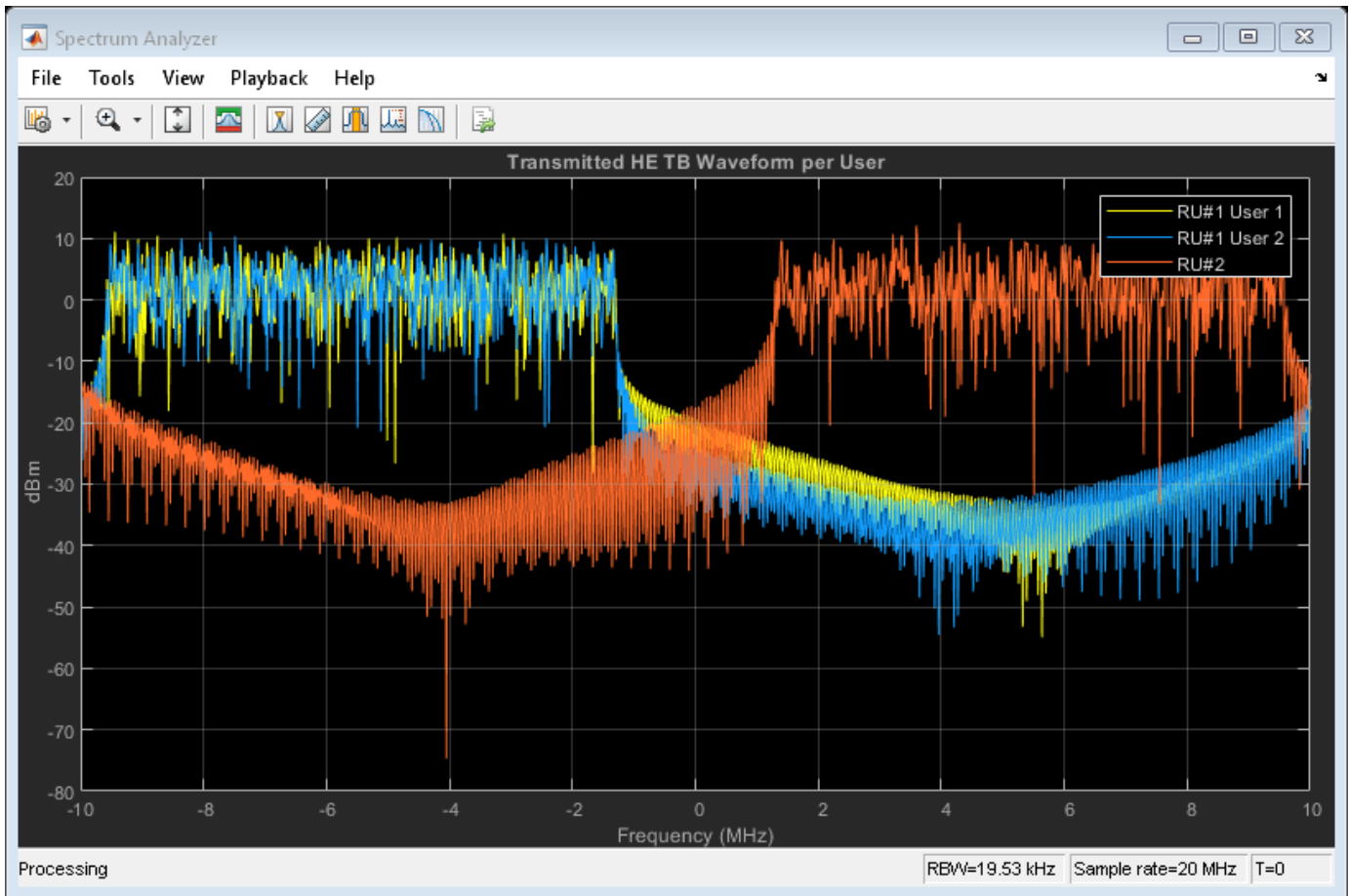
Appendix

The RU allocation table for allocations <= 20 MHz is shown below, with annotated descriptions.

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 + N)	52	52	-	106 (N users)					
24-31 (23 + N)	106 (N users)	-	52	52					
32-39 (31 + N)	26	26	26	26	26	106 (N users)			
40-47 (39 + N)	26	26	52	26	106 (N users)				
48-55 (47 + N)	52	26	26	26	106 (N users)				
56-63 (55 + N)	52	52	26	106 (N users)					
64-71 (63 + N)	106 (N users)	26	26	26	26	26			
72-79 (71 + N)	106 (N users)	26	26	26	52				
80-87 (79 + N)	106 (N users)	26	52	26	26				
88-95 (87 + N)	106 (N users)	26	52	52					
96-99 (95 + M)	106	-	106 (M users)						
100-103 (99 + M)	106 (2 users)	-	106 (M users)						
104-107 (103 + M)	106 (3 users)	-	106 (M users)						
108-111 (107 + M)	106 (4 users)	-	106 (M users)						
112	52	52	-	52	52				
113	Empty 242-tone RU - No user assigned								
116-127	Reserved								
128-135 (127 + N)	106	26	106 (N users)						
136-143 (135 + N)	106 (2 users)	26	106 (N users)						
144-151 (143 + N)	106 (3 users)	26	106 (N users)						
152-159 (151 + N)	106 (4 users)	26	106 (N users)						
160-167 (159 + N)	106 (5 users)	26	106 (N users)						
168-175 (167 + N)	106 (6 users)	26	106 (N users)						
176-183 (175 + N)	106 (7 users)	26	106 (N users)						
184-191 (183 + N)	106 (8 users)	26	106 (N users)						
192-199 (191 + N)	242 (N users)								

Annotations:
- 26 tone RU assigned to 1 user as part of a 20 MHz subchannel assignment of 9 26-tone RUs
- No users assigned to this RU; no data field transmitted on these subcarriers
- The number of users (N) assigned to this 106-tone RU depends on the allocation index and must be 1-8.
- The number of users (M) assigned to this 106-tone RU depends on the allocation index and must be 1-4.
- The number of users assigned to the upper 106-tone RU depends on the allocation index, but 2 users are always assigned to the lower 106-tone RU
- If selected, this 20 MHz subchannel is unused; the subchannel is punctured
- RU assigned to 1 user
- RU assigned to 1-4/8 users, depending on the allocation index
- RU assigned to specified number of users, irrespective of the allocation index

The RU allocation and HE-SIG-B user signaling for allocations > 20 MHz is shown in the table below, with annotated descriptions.

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 + N)	Full band 40 MHz (N users), or 484-tone RU with N users signaled in the corresponding HE-SIG-B content channel
208-215 (207 + N)	Full band 80 MHz (N users), or 996-tone RU with N users signaled in the corresponding HE-SIG-B content channel
216-223 (215 + N)	Full band 160 MHz (N users)
224-255	Reserved

Annotations:
- Must be used with other allocation indices. Signifies a 484-tone RU with zero users signaled 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 N users. N must be 1-8.
- >242-tone RU allocation

This example uses the following helper objects and functions:

- hePlotHESIGBAllocationMapping.m
- heRUAllocationTable.m

Selected Bibliography

1 IEEE P802.11ax™/D4.1 Draft 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 6: Enhancements for High Efficiency WLAN.

See Also

"Build HT PPDU" on page 1-65 | "Build Non-HT PPDU" on page 1-67 | "Build VHT PPDU" on page 1-63 | "Build S1G PPDU" on page 1-62 | "Build DMG PPDU" on page 1-60

More About

*   "Multiuser HE Transmission" on page 1-23

Build DMG PPDU

Build DMG PPDUs by using the waveform generator function.

Waveform Generator

Create an DMG configuration object.

```
dmg = wlanDMGConfig

dmg =
  wlanDMGConfig with properties:

                          MCS: '0'
               TrainingLength: 0
                   PSDULength: 1000
        ScramblerInitialization: 2
                   Turnaround: 0
```
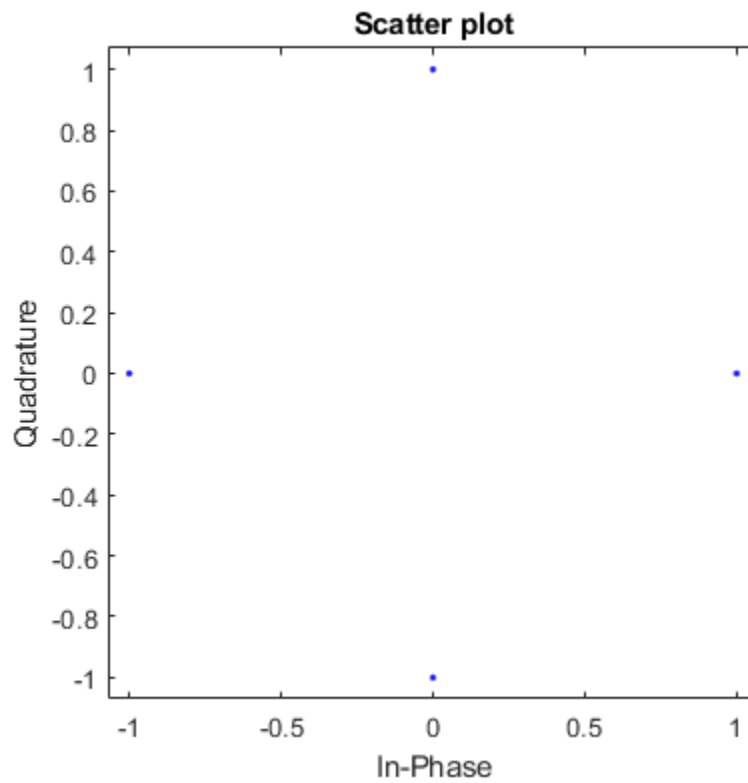
Generate the DMG PPDU. The length of the input data sequence in bits must be 8 times the length of the PSDU, which is expressed in bytes. Turn off windowing.

```
psdu = randi([0 1],dmg.PSDULength*8,1);
tx = wlanWaveformGenerator(psdu,dmg,'WindowTransitionTime',0);
```

The waveform has MCS=0, which is single carrier and DBPSK modulated. Plot the constellation of the waveform.

```
scatterplot(tx)
```

Scatter plot

See Also
"Build HT PPDU" on page 1-65 | "Build Non-HT PPDU" on page 1-67 | "Build VHT PPDU" on page 1-63 | "Build S1G PPDU" on page 1-62

Build S1G PPDU

Build S1G PPDUs by using the waveform generator function.

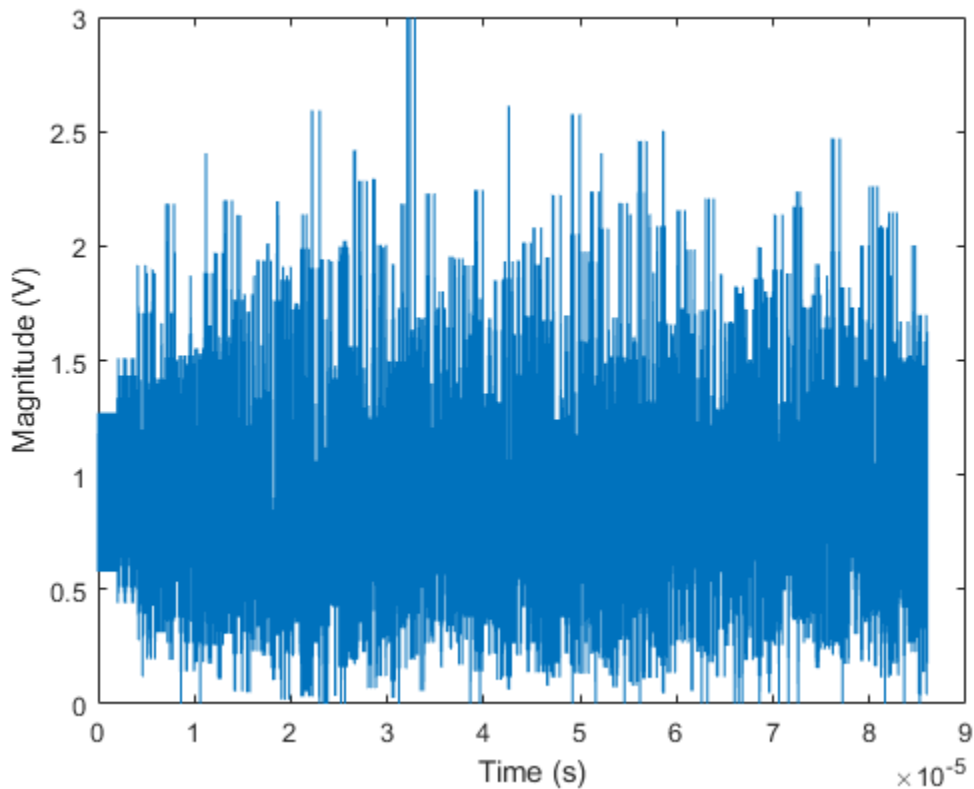**Waveform Generator**

Create an S1G configuration object.

```
s1g = wlanS1GConfig;
```

Generate the S1G PPDU. The length of the input data sequence in bits must be 8 times the length of the PSDU, which is expressed in bytes. Turn off windowing.

```
x = randi([0 1],s1g.PSDULength*8,1);
y = wlanWaveformGenerator(x,s1g,'WindowTransitionTime',0);
```

Plot the magnitude of the waveform.

```
t = ((1:length(y))'-1)/80e6;
plot(t,abs(y))
xlabel('Time (s)')
ylabel('Magnitude (V)')
```


See Also

"Build DMG PPDU" on page 1-60 | "Build HT PPDU" on page 1-65 | "Build Non-HT PPDU" on page 1-67 | "Build VHT PPDU" on page 1-63

Build VHT PPDU

Build VHT PPDUs by using the waveform generator function or by building each field individually.

Waveform Generator

Create a VHT configuration object.

```
vht = wlanVHTConfig;
```

Generate the VHT PPDU. The length of the input data sequence in bits must be 8 times the length of the PSDU, which is expressed in bytes. Turn off windowing.

```
x = randi([0 1],vht.PSDULength*8,1);
y = wlanWaveformGenerator(x,vht,'WindowTransitionTime',0);
```

Plot the magnitude of the waveform.

```
t = ((1:length(y))'-1)/80e6;
plot(t,abs(y))
xlabel('Time (s)')
ylabel('Magnitude (V)')
```
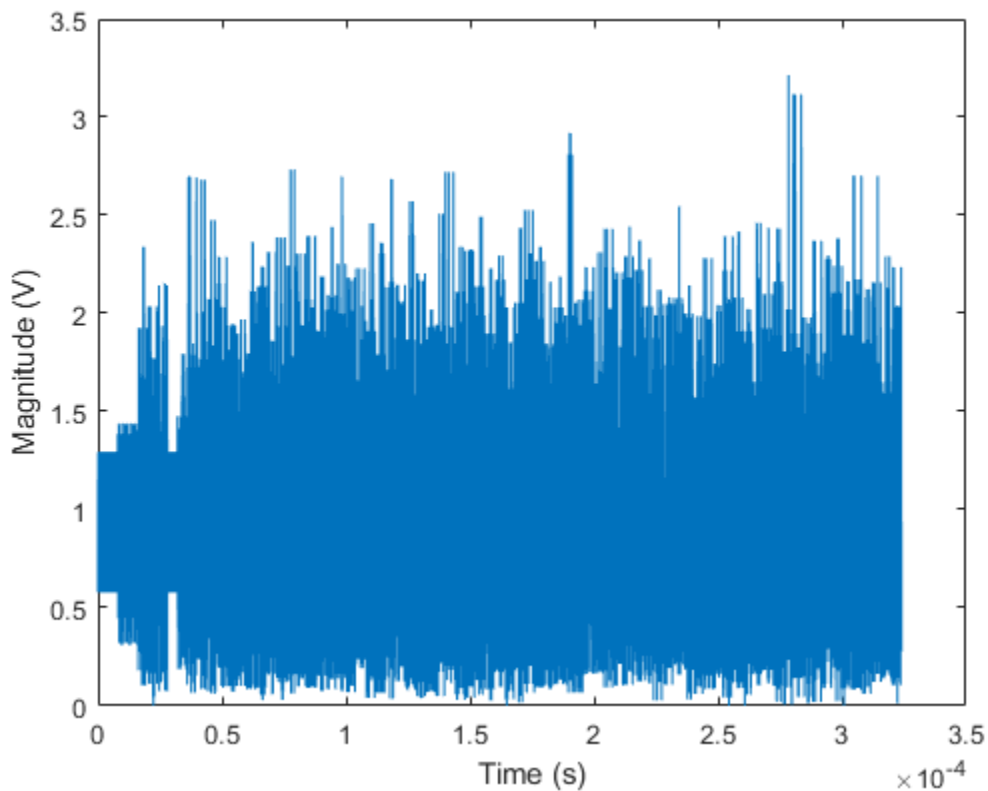

Individual PPDU Fields

Create L-STF, L-LTF, L-SIG, VHT-SIG-A, VHT-STF, VHT-LTF, and VHT-SIG-B preamble fields.

```
lstf = wlanLSTF(vht);
lltf = wlanLLTF(vht);
lsig = wlanLSIG(vht);
vhtSigA = wlanVHTSIGA(vht);
vhtstf = wlanVHTSTF(vht);
vhtltf = wlanVHTLTF(vht);
vhtSigB = wlanVHTSIGB(vht);
```

Generate the VHT-Data field using input data field x, which was used as an input to the waveform generator.

```
vhtData = wlanVHTData(x,vht);
```

Concatenate the individual fields to create a single PPDU.

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

Verify that the PPDUs created by the two methods are identical.

```
isequal(y,z)
```

```
ans = logical
   1
```

See Also
"Build DMG PPDU" on page 1-60 | "Build HT PPDU" on page 1-65 | "Build Non-HT PPDU" on page 1-67 | "Build S1G PPDU" on page 1-62

Build HT PPDU

Build HT PPDUs by using the waveform generator function or by building each field individually.

Waveform Generator

Create an HT configuration object.

```
ht = wlanHTConfig;
```

Generate the HT PPDU. The length of the input data sequence in bits must be 8 times the length of the PSDU, which is expressed in bytes. Turn windowing off.

```
x = randi([0 1],ht.PSDULength*8,1);
y = wlanWaveformGenerator(x,ht,'WindowTransitionTime',0);
```

Plot the magnitude of the waveform.

```
t = ((1:length(y))'-1)/20e6;
plot(t,abs(y))
xlabel('Time (s)')
ylabel('Magnitude (V)')
```
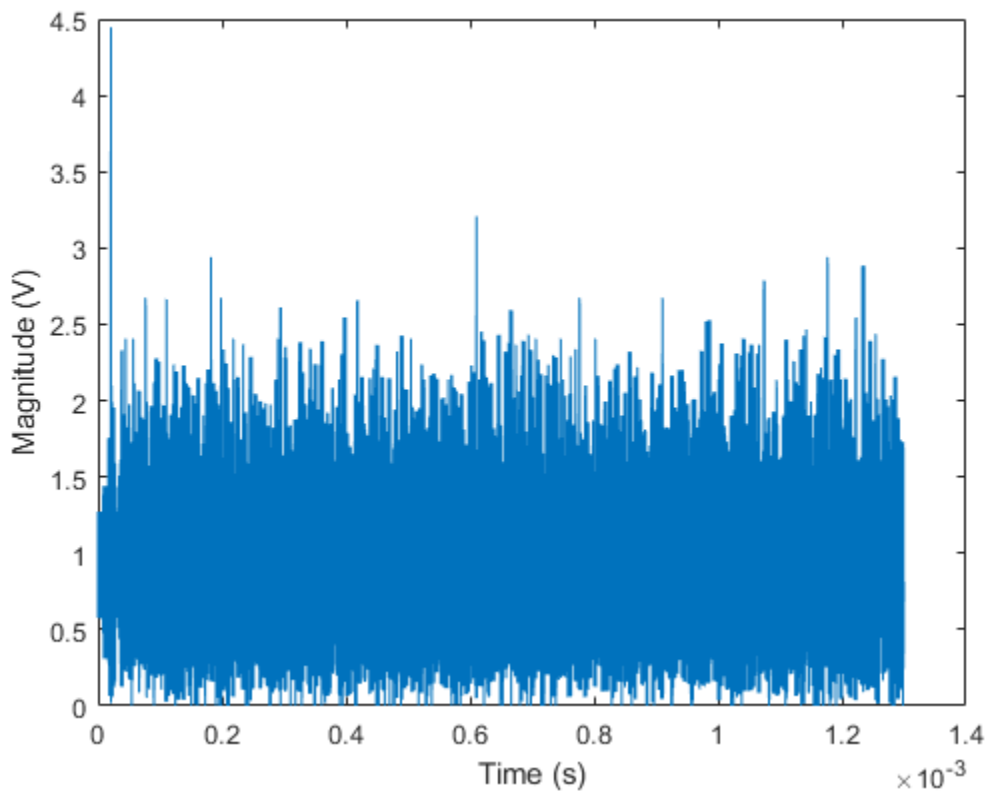

Individual PPDU Fields

Create L-STF, L-LTF, L-SIG, HT-SIG, HT-STF, and HT-LTF preamble fields.

```
lstf = wlanLSTF(ht);
lltf = wlanLLTF(ht);
lsig = wlanLSIG(ht);
htsig = wlanHTSIG(ht);
htstf = wlanHTSTF(ht);
htltf = wlanHTLTF(ht);
```

Generate the HT-Data field using input data field x, which is the same input signal that was used with the waveform generator.

```
htData = wlanHTData(x,ht);
```

Concatenate the individual fields to create a single PPDU.

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

Verify that the PPDUs created by the two methods are identical.

```
isequal(y,z)
```

```
ans = logical
   1
```

See Also
"Build DMG PPDU" on page 1-60 | "Build Non-HT PPDU" on page 1-67 | "Build S1G PPDU" on page 1-62 | "Build VHT PPDU" on page 1-63

Build Non-HT PPDU

Build non-HT PPDUs by using the waveform generator function or by building each field individually.

Waveform Generator

Create a non-HT configuration object.

```
nht = wlanNonHTConfig;
```

Generate the non-HT PPDU. The length of the input data sequence in bits must be 8 times the length of the PSDU, which is expressed in bytes. Turn off windowing.

```
x = randi([0 1],nht.PSDULength*8,1);
y = wlanWaveformGenerator(x,nht,'WindowTransitionTime',0);
```

Plot the magnitude of the waveform.

```
t = ((1:length(y))'-1)/20e6;
plot(t,abs(y))
xlabel('Time (s)')
ylabel('Magnitude (V)')
```
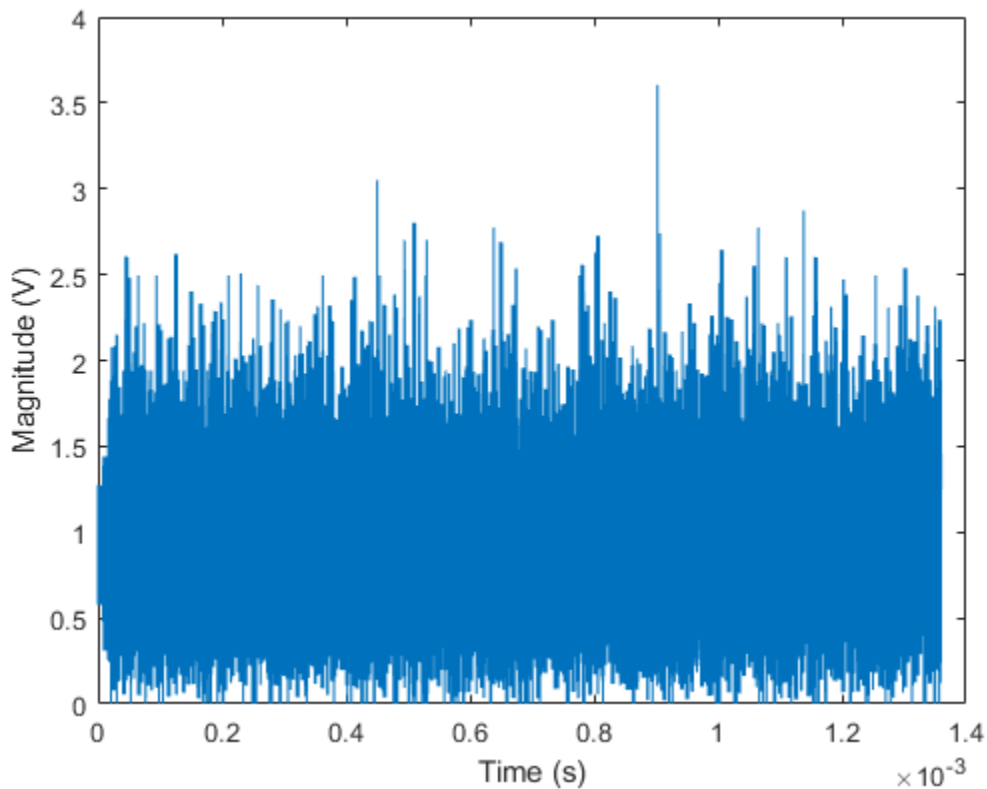

Individual PPDU Fields

Create L-STF, L-LTF, and L-SIG preamble fields.

```
lstf = wlanLSTF(nht);
lltf = wlanLLTF(nht);
lsig = wlanLSIG(nht);
```

Generate the Non-HT-data field using input data field x, which was used as the input to the waveform generator.

```
nhtData = wlanNonHTData(x,nht);
```

Concatenate the individual fields to create a single PPDU.

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

Verify that the PPDUs created by the two methods are identical.

```
isequal(y,z)
```

```
ans = logical
   1
```

See Also

"Build DMG PPDU" on page 1-60 | "Build HT PPDU" on page 1-65 | "Build S1G PPDU" on page 1-62 | "Build VHT PPDU" on page 1-63

Transmit and Recover L-SIG, VHT-SIG-A, VHT-SIG-B in Fading Channel

Transmit a VHT waveform through a noisy MIMO channel. Extract the L-SIG, VHT-SIG-A, and VHT-SIG-B fields and verify that they were correctly recovered.

Set the parameters used throughout the example.

```
cbw = 'CBW40';                      % Channel bandwidth
fs = 40e6;                          % Sample rate (Hz)
ntx = 2;                            % Number of transmit antennas
nsts = 2;                           % Number of space-time streams
nrx = 3;                            % Number of receive antennas
```

Create a VHT configuration object that supports a 2x2 MIMO transmission and has an APEP length of 2000.

```
vht = wlanVHTConfig('ChannelBandwidth',cbw,'APEPLength',2000, ...
    'NumTransmitAntennas',ntx,'NumSpaceTimeStreams',nsts, ...
    'SpatialMapping','Direct','STBC',false);
```

Generate a VHT waveform containing a random PSDU.

```
txPSDU = randi([0 1],vht.PSDULength*8,1);
txPPDU = wlanWaveformGenerator(txPSDU,vht);
```

Create a 2x2 TGac channel and an AWGN channel with an SNR=10 dB.

```
tgacChan = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',cbw, ...
    'NumTransmitAntennas',ntx,'NumReceiveAntennas',nrx, ...
    'LargeScaleFadingEffect','Pathloss and shadowing', ...
    'DelayProfile','Model-C');

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

Pass the VHT waveforms through a 2x2 TGac channel and add the AWGN channel noise.

```
rxPPDU = chNoise(tgacChan(txPPDU));
```

Add additional white noise corresponding to a receiver with a 9 dB noise figure. The noise variance is equal to $k*T*B*F$, where k is Boltzmann's constant, T is the ambient temperature, B is the channel bandwidth (sample rate), and F is the receiver noise figure.

```
nVar = 10^((-228.6+10*log10(290) + 10*log10(fs) + 9 )/10);
rxNoise = comm.AWGNChannel('NoiseMethod','Variance','Variance',nVar);

rxPPDU = rxNoise(rxPPDU);
```

Find the start and stop indices for all component fields of the PPDU.
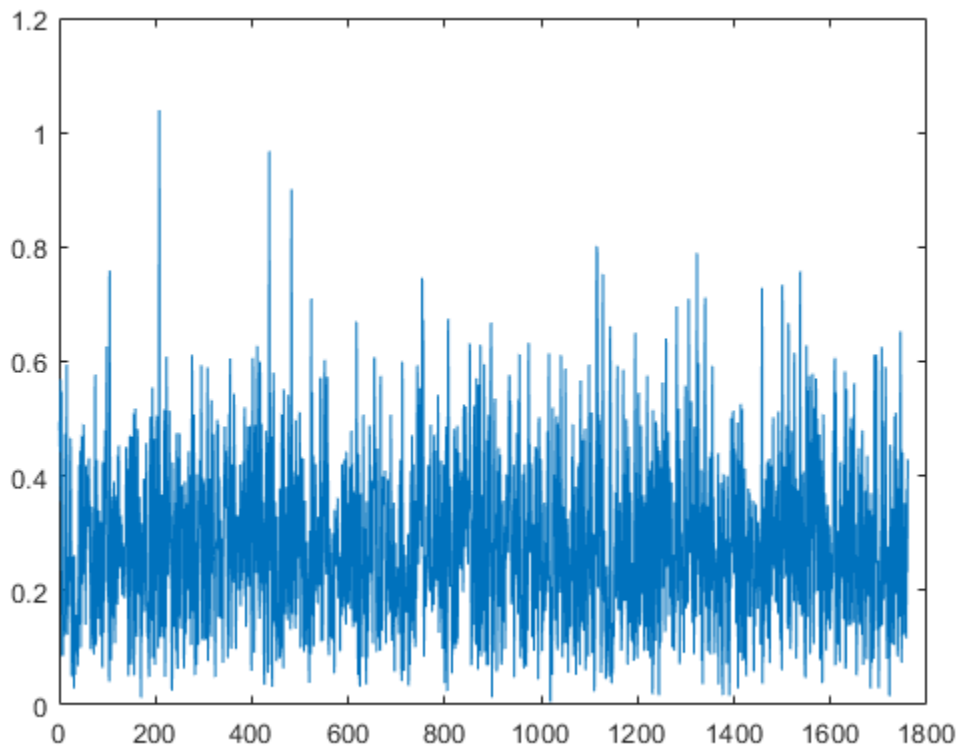
```
ind = wlanFieldIndices(vht)

ind = struct with fields:
      LSTF: [1 320]
      LLTF: [321 640]
      LSIG: [641 800]
```

```
    VHTSIGA: [801 1120]
     VHTSTF: [1121 1280]
     VHTLTF: [1281 1600]
    VHTSIGB: [1601 1760]
    VHTData: [1761 25600]
```

The preamble is contained in the first 1760 symbols. Plot the preamble.

```
plot(abs(rxPPDU(1:1760)))
```


Extract the L-LTF from the received PPDU using the start and stop indices determined by the `wlanFieldIndices` function. Demodulate the L-LTF and estimate the channel coefficients.

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

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

```
rxLSIG = rxPPDU(ind.LSIG(1):ind.LSIG(2),:);
infoLSIG = wlanLSIGRecover(rxLSIG,chEstLLTF,nVar,cbw);
```

Inspect the L-SIG rate information and confirm that the sequence [1 1 0 1] is received. This sequence corresponds to a 6 MHz data rate, which is used for all VHT transmissions.

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

```
rate = 1x4 int8 row vector

   0   1   1   1
```

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

```
rxVHTSIGA = rxPPDU(ind.VHTSIGA(1):ind.VHTSIGA(2),:);
[infoVHTSIGA,failCRC] = wlanVHTSIGARecover(rxVHTSIGA, ...
    chEstLLTF,nVar,cbw);
failCRC
```

```
failCRC = logical
   1
```

Extract and demodulate the VHT-LTF. Use the demodulated signal to estimate the channel coefficients needed to recover the VHT-SIG-B field.

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

Extract and recover the VHT-SIG-B.

```
rxVHTSIGB = rxPPDU(ind.VHTSIGB(1):ind.VHTSIGB(2),:);
infoVHTSIGB = wlanVHTSIGBRecover(rxVHTSIGB,chEstVHTLTF,nVar,cbw);
```

Verify that the APEP length, contained in the first 19 bits of the VHT-SIG-B, corresponds to the specified length of 2000 bits.

```
pktLbits = infoVHTSIGB(1:19)';
pktLen = bi2de(double(pktLbits))*4
```

```
pktLen = 1676920
```

End-to-End VHT Simulation with Frequency Correction

This example shows how to generate, transmit, recover and view a VHT MIMO waveform.

Steps in the example:

- Transmit a VHT waveform through a MIMO channel with AWGN
- Perform a two-stage process to estimate and correct for a frequency offset
- Estimate the channel response
- Recover the VHT data field
- Compare the transmitted and received PSDUs to determine if bit errors occurred

Set the parameters used throughout the example.

```
cbw = 'CBW160';                    % Channel bandwidth
fs = 160e6;                        % Sample rate (Hz)
ntx = 2;                           % Number of transmit antennas
nsts = 2;                          % Number of space-time streams
nrx = 2;                           % Number of receive antennas
```

Create a VHT configuration object that supports a 2x2 MIMO transmission and has an APEP length of 2000.

```
vht = wlanVHTConfig('ChannelBandwidth',cbw,'APEPLength',2000, ...
    'NumTransmitAntennas',ntx,'NumSpaceTimeStreams',nsts, ...
    'SpatialMapping','Direct','STBC',false);
```

Generate a VHT waveform containing a random PSDU.

```
txPSDU = randi([0 1],vht.PSDULength*8,1);
txPPDU = wlanWaveformGenerator(txPSDU,vht);
```

Create a 2x2 TGac channel and an AWGN channel.

```
tgacChan = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',cbw, ...
    'NumTransmitAntennas',ntx,'NumReceiveAntennas',nrx, ...
    'LargeScaleFadingEffect','Pathloss and shadowing', ...
    'DelayProfile','Model-C');
awgnChan = comm.AWGNChannel('NoiseMethod','Variance', ...
    'VarianceSource','Input port');
```

Create a phase/frequency offset object.

```
pfOffset = comm.PhaseFrequencyOffset('SampleRate',fs,'FrequencyOffsetSource','Input port');
```

Calculate the noise variance for a receiver with a 9 dB noise figure. Pass the transmitted waveform through the noisy TGac channel.

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

Introduce a frequency offset of 500 Hz.

```
rxPPDUcfo = pfOffset(rxPPDU,500);
```

Find the start and stop indices for all component fields of the PPDU.

```
ind = wlanFieldIndices(vht);
```

Extract the L-STF. Estimate and correct for the carrier frequency offset.

```
rxLSTF = rxPPDUcfo(ind.LSTF(1):ind.LSTF(2),:);
```

```
foffset1 = wlanCoarseCFOEstimate(rxLSTF,cbw);
rxPPDUcorr = pfOffset(rxPPDUcfo,-foffset1);
```

Extract the L-LTF from the corrected signal. Estimate and correct for the residual frequency offset.

```
rxLLTF = rxPPDUcorr(ind.LLTF(1):ind.LLTF(2),:);
```

```
foffset2 = wlanFineCFOEstimate(rxLLTF,cbw);
rxPPDU2 = pfOffset(rxPPDUcorr,-foffset2);
```

Extract and demodulate the VHT-LTF. Estimate the channel coefficients.

```
rxVHTLTF = rxPPDU2(ind.VHTLTF(1):ind.VHTLTF(2),:);
dLTF = wlanVHTLTFDemodulate(rxVHTLTF,vht);
chEst = wlanVHTLTFChannelEstimate(dLTF,vht);
```

Extract the VHT data field from the received and frequency-corrected PPDU. Recover the data field.

```
rxVHTData = rxPPDU2(ind.VHTData(1):ind.VHTData(2),:);
rxPSDU = wlanVHTDataRecover(rxVHTData,chEst,nVar,vht);
```

Calculate the number of bit errors in the received packet.

```
numErr = biterr(txPSDU,rxPSDU)
```
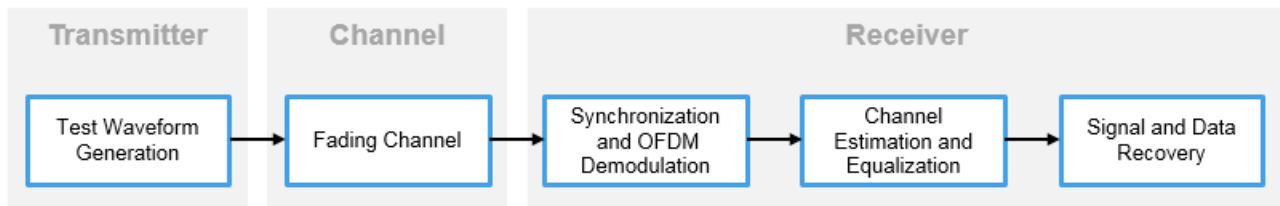
```
numErr = 0
```

Transmit-Receive Chain

In this section...
"Transmit Processing Chain" on page 1-74
"Receiver Processing Chain" on page 1-77

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

- Transmitter functions enable simulation of the various IEEE 802.11 [2] 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 multiuser (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.

---

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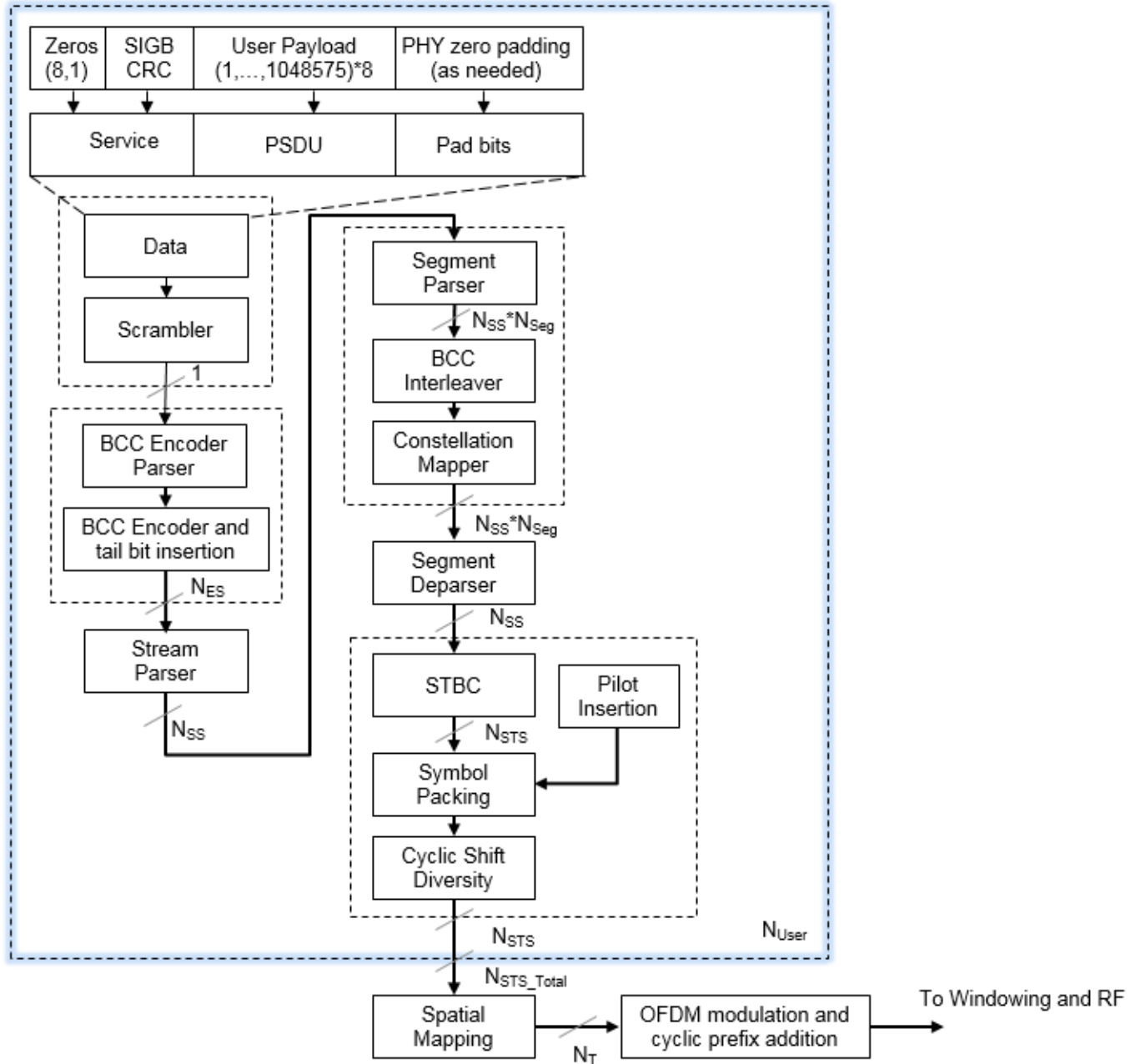
HT Data Transmit Processing Chain

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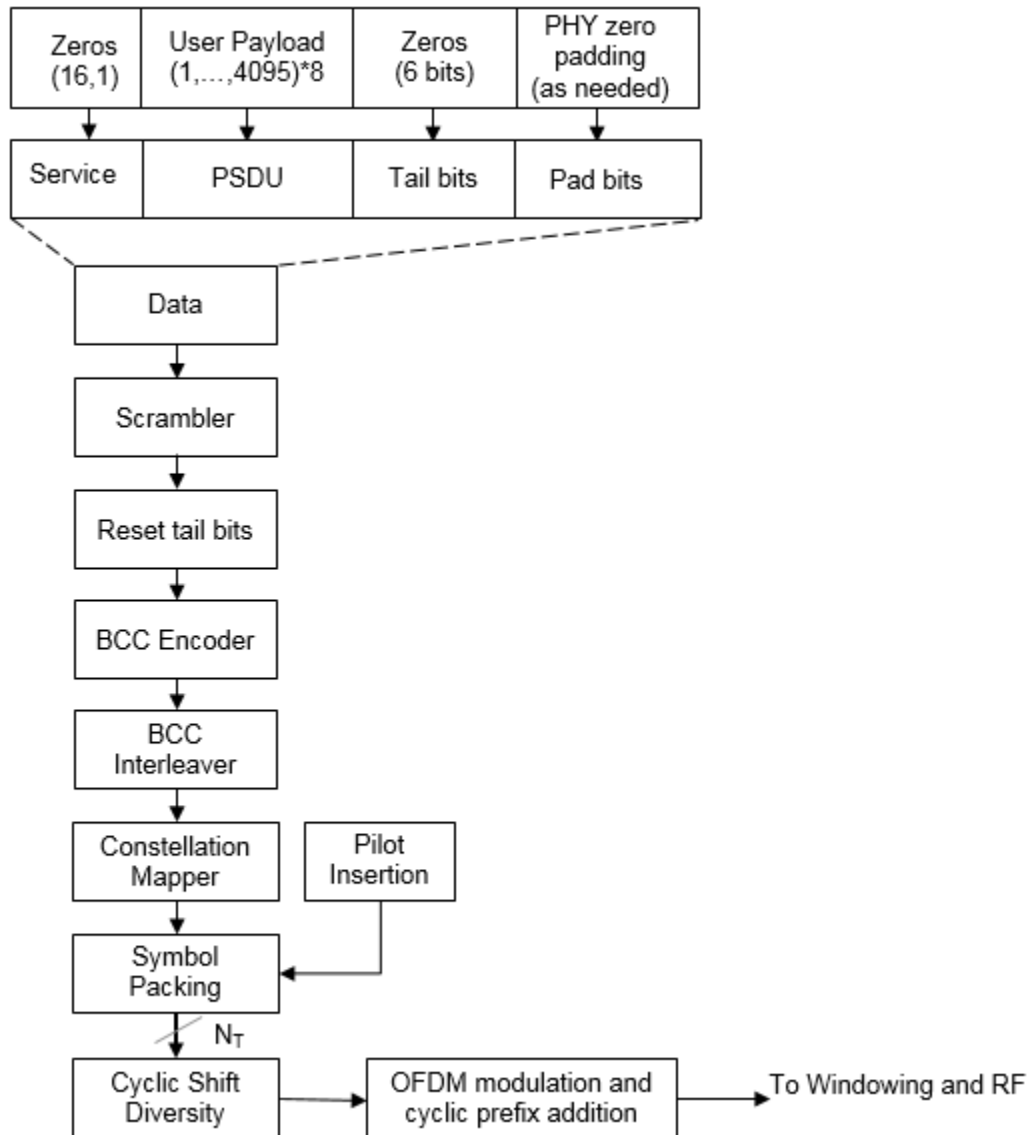
IEEE 802.11-2012 [3], Section 18 defines requirements for physical layer processing associated with each PPDU field for the OFDM modulation scheme. IEEE 802.11-2012 [3], Section 17, and Section 19 define requirements for physical layer processing associated with each PPDU field for the DSSS modulation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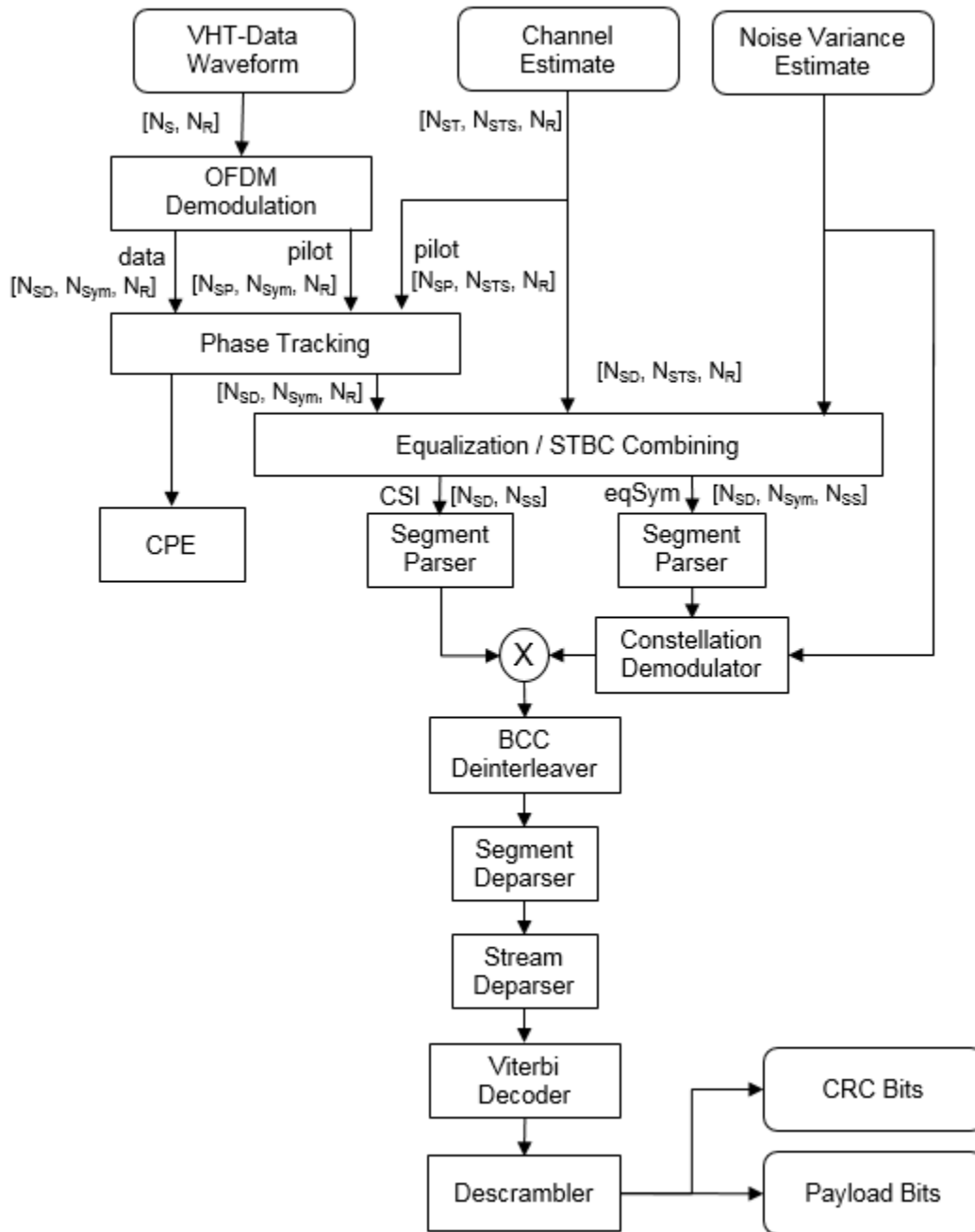
HT Data Receive Processing Chain

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.

```
┌─────────────────┐      ┌─────────────────┐      ┌─────────────────┐
│    HT-Data      │      │    Channel      │      │ Noise Variance  │
│    Waveform     │      │    Estimate     │      │    Estimate     │
└─────────────────┘      └─────────────────┘      └─────────────────┘
```
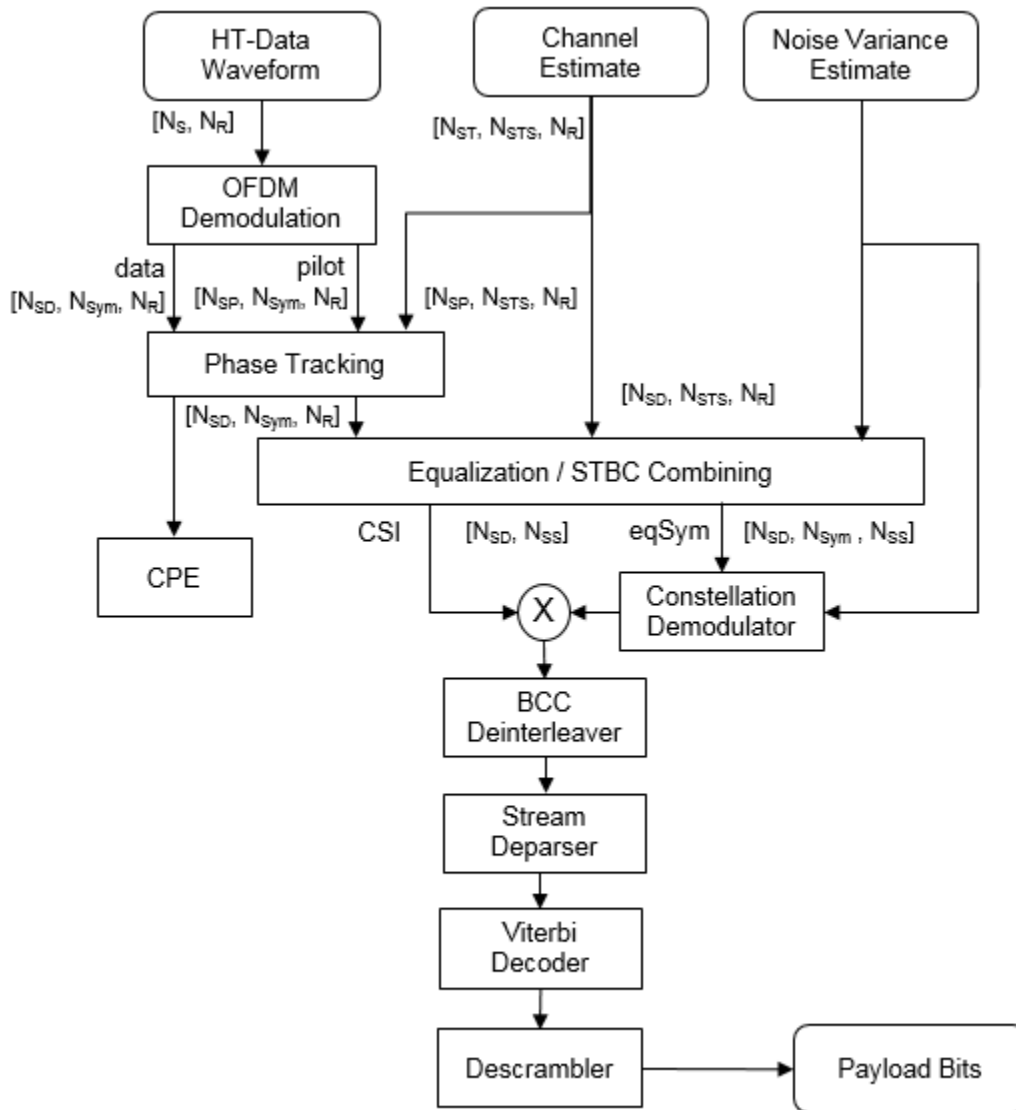
$[N_S, N_R]$ $[N_{ST}, N_{STS}, N_R]$

```
┌─────────────────┐
│     OFDM        │
│  Demodulation   │
└─────────────────┘
```

data   pilot

$[N_{SD}, N_{Sym}, N_R]$ $[N_{SP}, N_{Sym}, N_R]$ $[N_{SP}, N_{STS}, N_R]$

```
┌─────────────────┐
│  Phase Tracking │
└─────────────────┘
```

$[N_{SD}, N_{Sym}, N_R]$ $[N_{SD}, N_{STS}, N_R]$

```
┌──────────────────────────────────────────┐
│        Equalization / STBC Combining      │
└──────────────────────────────────────────┘
```

CSI $[N_{SD}, N_{SS}]$ eqSym $[N_{SD}, N_{Sym}, N_{SS}]$

```
┌────────┐        ⊗(X)      ┌─────────────────┐
│  CPE   │                   │  Constellation  │
└────────┘                   │   Demodulator   │
                             └─────────────────┘
┌─────────────────┐
│      BCC        │
│  Deinterleaver  │
└─────────────────┘
┌─────────────────┐
│     Stream      │
│    Deparser     │
└─────────────────┘
┌─────────────────┐
│     Viterbi     │
│     Decoder     │
└─────────────────┘
┌─────────────────┐        ┌─────────────────┐
│   Descrambler   │ ─────> │  Payload Bits   │
└─────────────────┘        └─────────────────┘
```

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.
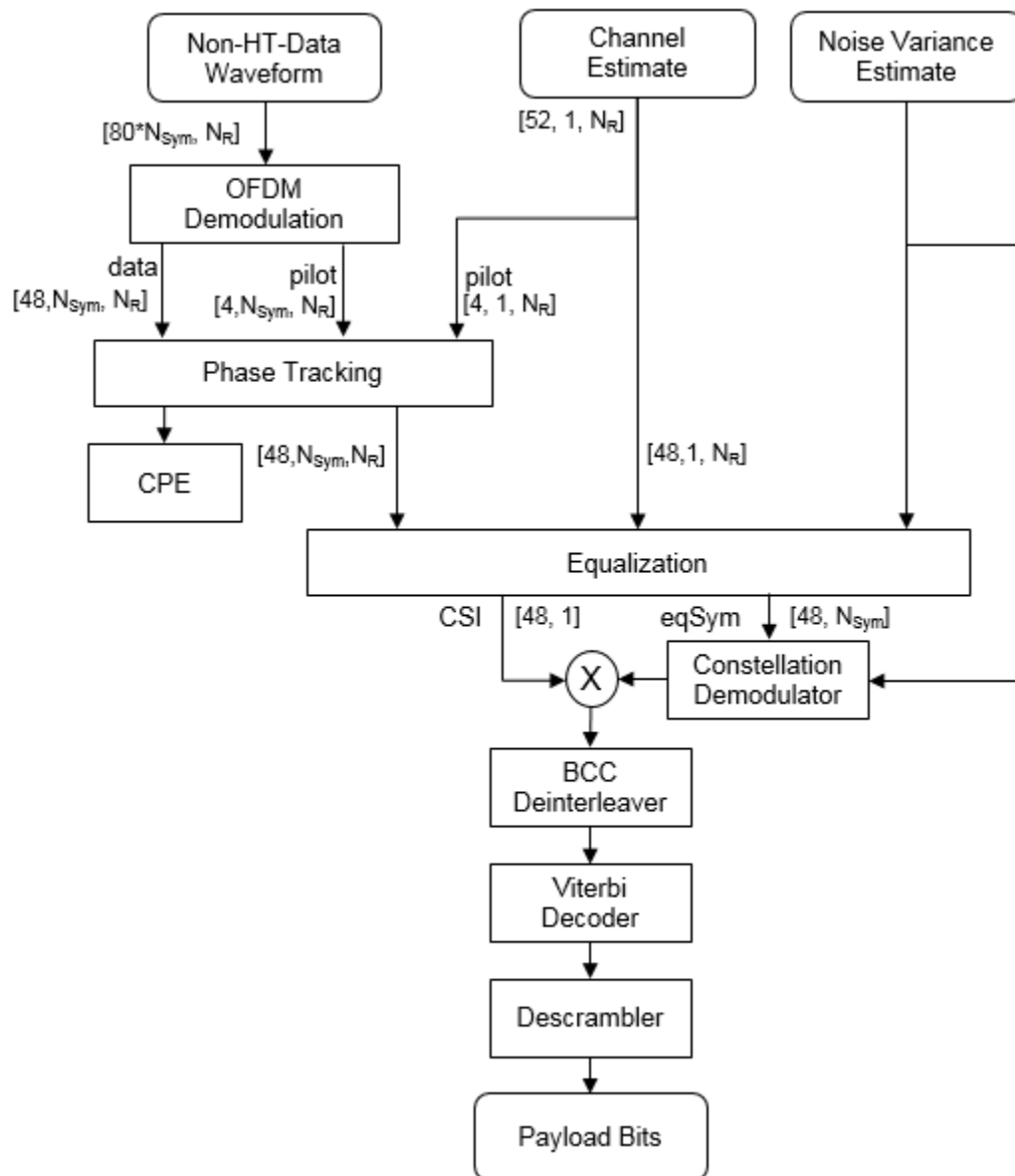
References

[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” on page 1-69 | “End-to-End VHT Simulation with Frequency Correction” on page 1-72

Delay Profile and Fluorescent Lighting Effects

This example demonstrates the impact of changing the TGac delay profile, and it shows how fluorescent lighting affects the time response of the channel.

Delay Profile Effects

Create VHT configuration object. Set the sample rate to 80 MHz.

```
vht = wlanVHTConfig;
fs = 80e6;
```

Generate random binary data and create a transmit waveform using the configuration objects.

```
d = randi([0 1],8*vht.PSDULength,1);
txSig = wlanWaveformGenerator(d,vht);
```

Create a TGac channel object. Set the delay profile to `'Model-A'`, which corresponds to flat fading. Disable the large-scale fading effects.

```
tgacChan = wlanTGacChannel('SampleRate',fs, ...
    'ChannelBandwidth',vht.ChannelBandwidth, ...
    'DelayProfile','Model-A', ...
    'LargeScaleFadingEffect','None');
```

Pass the transmitted waveform through the TGac channel.

```
rxSigA = tgacChan(txSig);
```

Set the delay profile to `'Model-C'`. Model-C corresponds to a multipath channel having 14 distinct paths, with a 30 ns RMS delay spread. The maximum delay spread is 200 ns, which corresponds to a coherence bandwidth of 2.5 MHz.
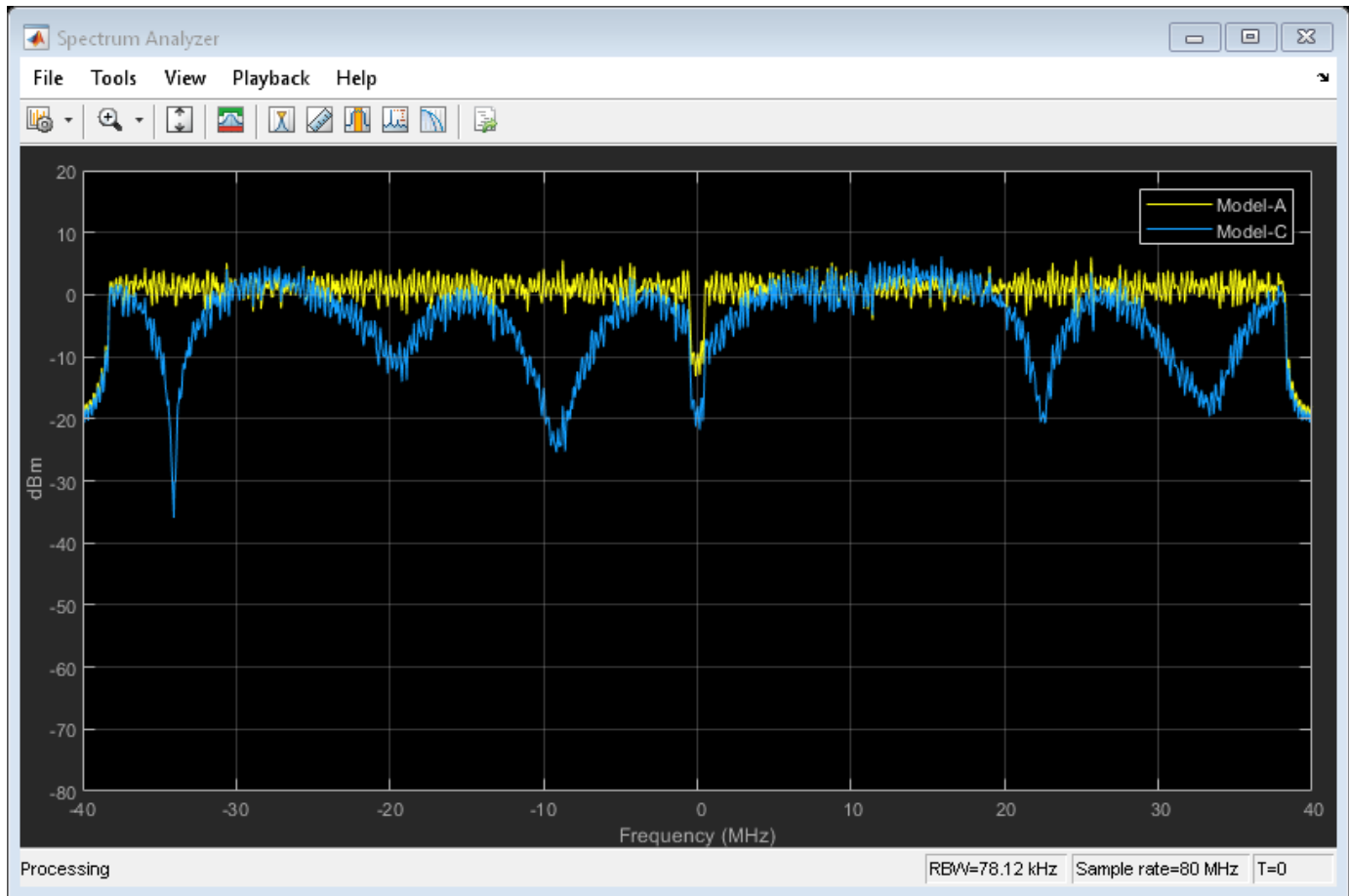
```
release(tgacChan)
tgacChan.DelayProfile = 'Model-C';
```

Pass the waveform through the model-C channel.

```
rxSigC = tgacChan(txSig);
```

Create a spectrum analyzer and use it to visualize the spectrum of the received signals.

```
saScope = dsp.SpectrumAnalyzer('SampleRate',fs, ...
    'ShowLegend',true,'ChannelNames',{'Model-A','Model-C'}, ...
    'SpectralAverages',10);
saScope([rxSigA rxSigC])
```

As expected, the frequency response of the model-A signal is flat across the 80 MHz bandwidth. Conversely, the model-C frequency response varies because its coherence bandwidth is much smaller than the channel bandwidth.

Fluorescent Effects

Release the TGac channel, and set its delay profile to `'Model-D'`. Disable the fluorescent lighting effect.

```
release(tgacChan)
tgacChan.DelayProfile = 'Model-D';
tgacChan.FluorescentEffect = false;
```

To better illustrate the Doppler effects of fluorescent lighting, change the bandwidth and sample rate of the channel. Generate a test waveform of all ones.

```
tgacChan.ChannelBandwidth = 'CBW20';
fs = 20e6;
tgacChan.SampleRate = fs;

txSig = ones(5e5,1);
```

To ensure repeatability, set the global random number generator to a fixed value.

```
rng(37)
```

Pass the waveform through the TGac channel.

```
rxSig0 = tgacChan(txSig);
```

Enable the fluorescent lighting effect. Reset the random number generator, and pass the waveform through the channel.

```
release(tgacChan)
tgacChan.FluorescentEffect = true;
rng(37)
rxSig1 = tgacChan(txSig);
```

Determine the time axis and channel filter delay.

```
t = ((1:size(rxSig0,1))'-1)/fs;
fDelay = tgacChan.info.ChannelFilterDelay;
```

Plot the magnitude of the received signals while accounting for the channel filter delay.

```
plot(t(fDelay+1:end),[abs(rxSig0(fDelay+1:end)) abs(rxSig1(fDelay+1:end))])
xlabel('Time (s)')
ylabel('Magnitude (V)')
legend('Fluorescent Off','Fluorescent On','location','best')
```
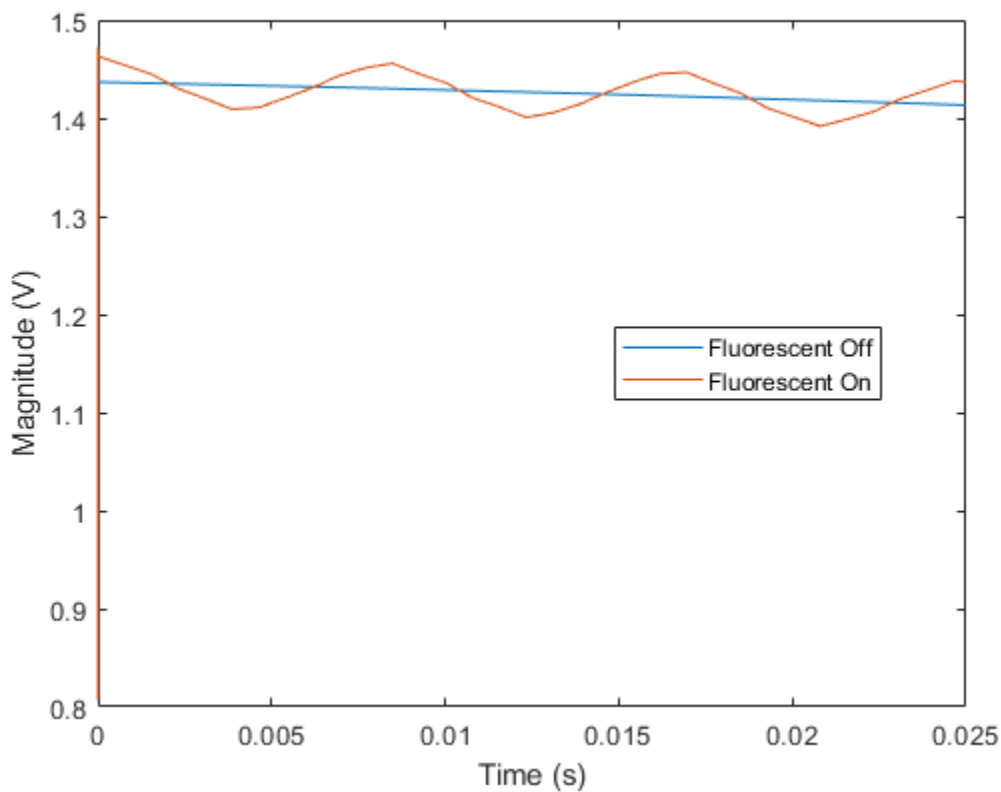
Fluorescent lighting introduces a Doppler component at twice the power line frequency (120 Hz in the U.S.).

Confirm that the peaks are separated by approximately 0.0083 s (inverse of 120 Hz) by measuring distance between the second and third peaks.

```
[~,loc] = findpeaks(abs(rxSig1(1e5:4e5)));
peakTimes = loc/fs;
peakSeparation = diff(peakTimes)


peakSeparation =

    0.0085
```

Generate Multiuser VHT Waveform

This example shows how to generate a multiuser VHT waveform from individual components. It also shows how to generate the same waveform by using the `wlanWaveformGenerator` function.

Create a VHT configuration object, specifying three users and three transmit antennas.

```
vht = wlanVHTConfig('NumUsers',3,'NumTransmitAntennas',3);
```

Set the number of space-time streams to the vector [1 1 1], which indicates that each user is assigned one space-time stream. Set the user positions to [0 1 2]. Set the group ID to 5. Group ID values from 1 to 62 apply for multiuser operation.

```
vht.NumSpaceTimeStreams = [1 1 1];
vht.UserPositions = [0 1 2];
vht.GroupID = 5;
```

Set a different MCS value for each user.

```
vht.MCS = [0 2 4];
```

Set the APEP length to 2000, 1400, and 1800 bytes. Each element corresponds to the number of bytes assigned to each user.

```
vht.APEPLength = [2000 1400 1800]

vht =
  wlanVHTConfig with properties:

        ChannelBandwidth: 'CBW80'
                NumUsers: 3
           UserPositions: [0 1 2]
     NumTransmitAntennas: 3
    NumSpaceTimeStreams: [1 1 1]
          SpatialMapping: 'Direct'
                     MCS: [0 2 4]
          ChannelCoding: 'BCC'
              APEPLength: [2000 1400 1800]
           GuardInterval: 'Long'
                 GroupID: 5

    Read-only properties:
              PSDULength: [2000 6008 12019]
```

Display the PSDU lengths for the three users. The PSDU length is a function of both the APEP length and the MCS value.

```
vht.PSDULength

ans = 1×3

        2000        6008       12019
```

Display the field indices for the VHT waveform.

```
ind = wlanFieldIndices(vht)
```

```
ind = struct with fields:
        LSTF: [1 640]
        LLTF: [641 1280]
        LSIG: [1281 1600]
     VHTSIGA: [1601 2240]
      VHTSTF: [2241 2560]
      VHTLTF: [2561 3840]
     VHTSIGB: [3841 4160]
     VHTData: [4161 48000]
```

Create the individual fields that comprise the VHT waveform.

```
lstf = wlanLSTF(vht);
lltf = wlanLLTF(vht);
lsig = wlanLSIG(vht);
[vhtsigA,sigAbits] = wlanVHTSIGA(vht);
vhtstf = wlanVHTSTF(vht);
vhtltf = wlanVHTLTF(vht);
[vhtsigB,sigBbits] = wlanVHTSIGB(vht);
```

Extract the first two VHT-SIG-A information bits and convert them to their decimal equivalent.

```
bw = bi2de(double(sigAbits(1:2)'))
```

```
bw = 2
```

The value, 2, corresponds to an 80 MHz bandwidth (see `wlanVHTSIGA`).

Extract VHT-SIG-A information bits 5 through 10, and convert them to their decimal equivalent.

```
groupid = bi2de(double(sigAbits(5:10)'))
```

```
groupid = 5
```

The extracted group ID, 5, matches the corresponding property in the VHT configuration object.

Extract the packet length from the VHT-SIG-B information bits. For multiuser operation with an 80 MHz bandwidth, the first 19 bits contain the APEP length information. Convert the field lengths to their decimal equivalents. Multiply them by 4 because the length of the VHT-SIG-B field is expressed in units of 4 bytes.

```
pktLen = bi2de(double(sigBbits(1:19,:)'))*4
```

```
pktLen = 3×1

        2000
        1400
        1800
```

Confirm that the extracted APEP length matches the value set in the configuration object.

```
isequal(pktLen',vht.APEPLength)
```

```
ans = logical
   1
```

Extract the MCS values from the VHT-SIG-B information bits. The MCS component is specified by bits 20 to 23.

```
mcs = bi2de(double(sigBbits(20:23,:)'))
```

mcs = *3×1*

```
     0
     2
     4
```

The values correspond to those set in the VHT configuration object.

Create three data sequences, one for each user.

```
d1 = randi([0 1],vht.PSDULength(1)*8,1);
d2 = randi([0 1],vht.PSDULength(2)*8,1);
d3 = randi([0 1],vht.PSDULength(3)*8,1);
```

Generate a VHT data field using these data sequences.

```
vhtdata = wlanVHTData({d1 d2 d3},vht);
```

Generate a multiuser VHT waveform with windowing is disabled. Extract the data field from the waveform.
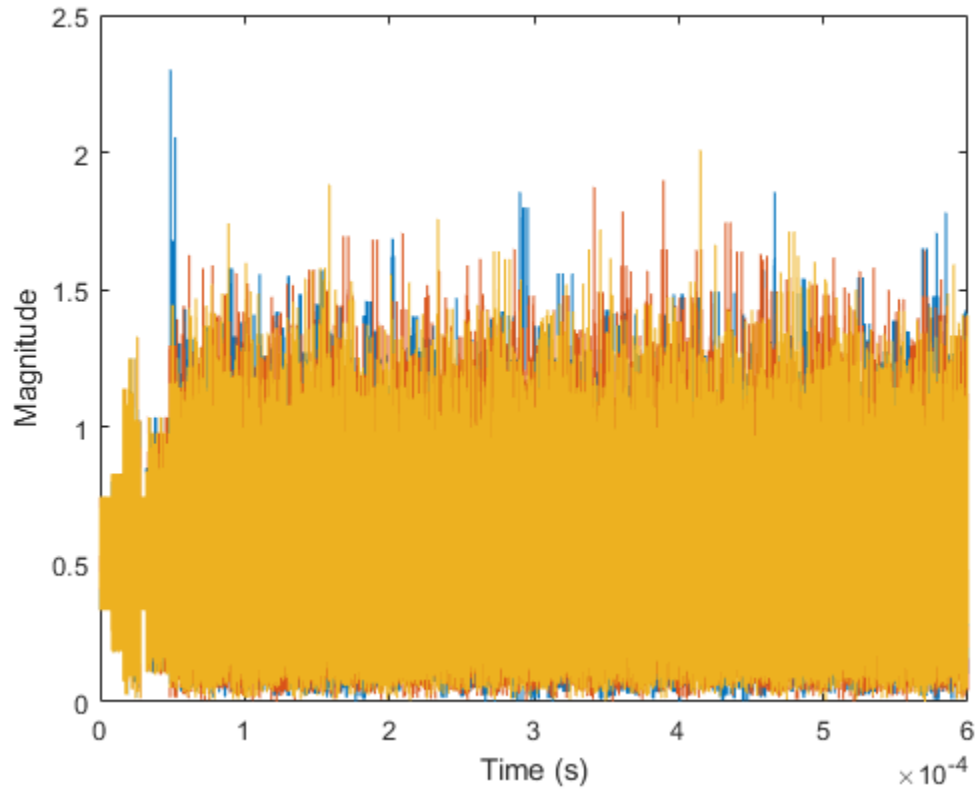
```
wv = wlanWaveformGenerator({d1 d2 d3},vht,'WindowTransitionTime',0);
wvdata = wv(ind.VHTData(1):ind.VHTData(2),:);
```

Confirm that the two generation approaches produce identical results.

```
isequal(vhtdata,wvdata)
```

ans = *logical*
```
   1
```

Visualize the waveform by plotting its magnitude.

```
t = ((1:length(wv))'-1)/80e6;
plot(t,abs(wv))
xlabel('Time (s)')
ylabel('Magnitude')
```

Basic VHT Data Recovery Steps

This example shows how to perform basic VHT data recovery. It also shows how to recover VHT data when the received signal has a carrier frequency offset. Similar procedures can be used to recover data with the HT and non-HT formats.

Basic Data Recovery

The WLAN Toolbox™ provides functions to generate and recover IEEE® 802.11ac™ standards compliant waveforms. Data recovery is accomplished by these steps:

1 Generate a VHT waveform

2 Pass the waveform through a channel

3 Extract the VHT-LTF and demodulate

4 Estimate the channel by using the demodulated VHT-LTF

5 Extract the data field

6 Recover the data by using the channel and noise variance estimates

The block diagram shows these steps, along with their corresponding commands.
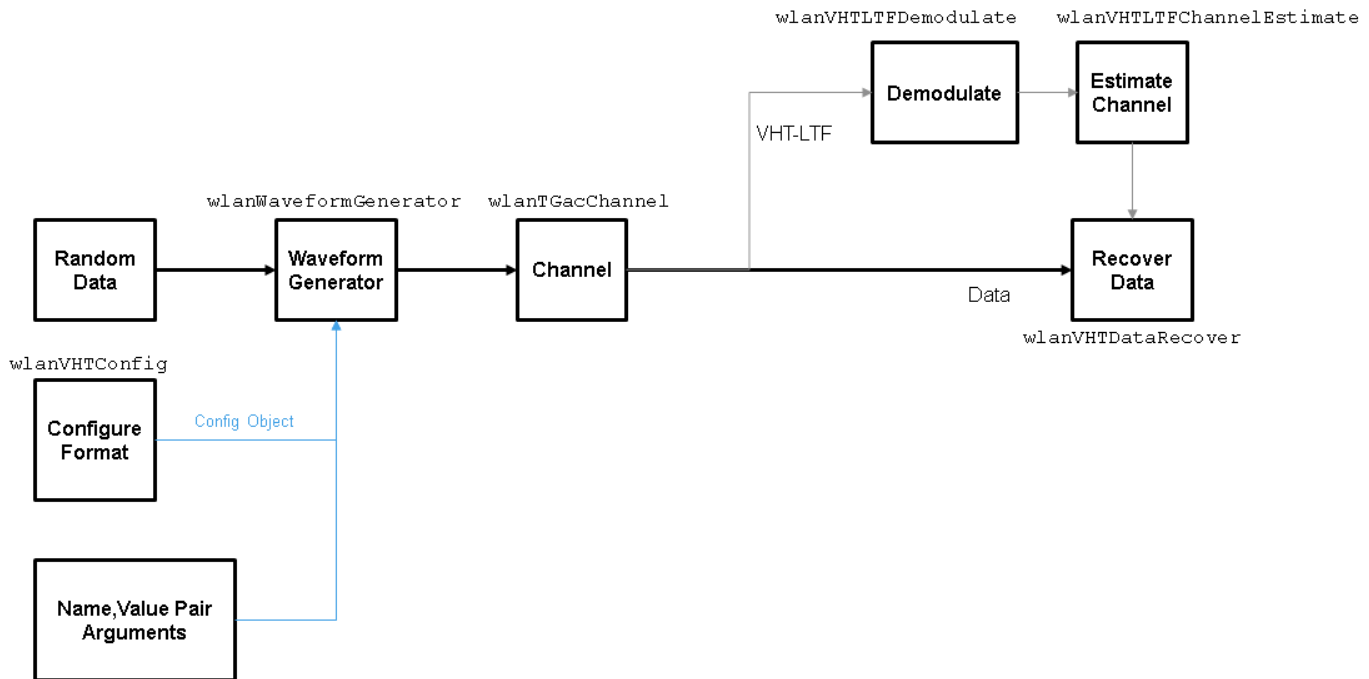
Create VHT format configuration object.

```
vht = wlanVHTConfig;
```

Create a VHT transmit waveform by using the VHT configuration object. Set the data sequence to [1;0;1;1]. The data sequence is repeated to generate the specified number of packets.

```
txSig = wlanWaveformGenerator([1;0;1;1],vht);
```

Pass the received signal through an AWGN channel.

```
rxSig = awgn(txSig,10);
```

Determine the field indices of the waveform.

```
ind = wlanFieldIndices(vht);
```

Extract the VHT-LTF from the received signal.

```
rxVHTLTF = rxSig(ind.VHTLTF(1):ind.VHTLTF(2),:);
```

Demodulate the VHT-LTF. Estimate the channel response by using the demodulated signal.

```
demodVHTLTF = wlanVHTLTFDemodulate(rxVHTLTF,vht);
chEst = wlanVHTLTFChannelEstimate(demodVHTLTF,vht);
```

Extract the VHT data field.

```
rxData = rxSig(ind.VHTData(1):ind.VHTData(2),:);
```

Recover the information bits by using the channel and noise variance estimates. Confirm that the first 8 bits match two repetitions of the input data sequence of [1;0;1;1].

```
rxBits = wlanVHTDataRecover(rxData,chEst,0.1,vht);

rxBits(1:8)

ans = 8x1 int8 column vector

   1
   0
   1
   1
   1
   0
   1
   1
```

Data Recovery with Frequency Correction

Data recovery when a carrier frequency offset is present is accomplished by these steps:

1 Generate a VHT waveform

2 Pass the waveform through a channel

3 Extract the L-STF and perform a coarse frequency offset estimate

4 Correct for the offset by using the coarse estimate

5 Extract the L-LTF and perform a fine frequency offset estimate

6 Correct for the offset by using the fine estimate

7 Extract the VHT-LTF and demodulate

8 Estimate the channel by using the demodulated VHT-LTF

9 Extract the data field

10 Recover the data by using the channel and noise variance estimates

The block diagram shows these steps, along with their corresponding commands.
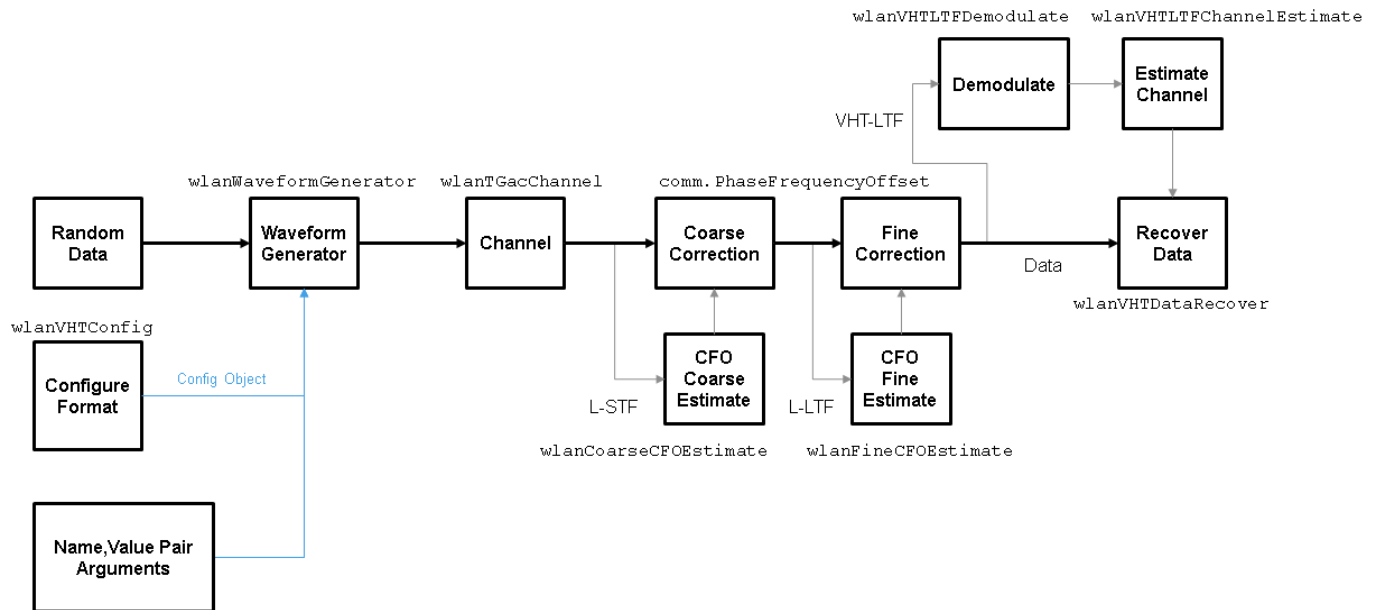
Set the channel bandwidth and sample rate.

```
cbw = 'CBW160';
fs = 160e6;
```

Create a VHT configuration object that supports a 2x2 MIMO transmission.

```
vht = wlanVHTConfig('ChannelBandwidth',cbw, ...
    'NumTransmitAntennas',2,'NumSpaceTimeStreams',2);
```

Generate a VHT waveform containing a random PSDU.

```
txPSDU = randi([0 1],vht.PSDULength*8,1);
txSig = wlanWaveformGenerator(txPSDU,vht);
```

Create a 2x2 TGac channel.

```
tgacChan = wlanTGacChannel('SampleRate',fs,'ChannelBandwidth',cbw, ...
    'NumTransmitAntennas',2,'NumReceiveAntennas',2);
```

Create a phase and frequency offset object.

```
pfOffset = comm.PhaseFrequencyOffset('SampleRate',fs,'FrequencyOffsetSource','Input port');
```

Pass the transmitted waveform through the noisy TGac channel.

```
rxSigNoNoise = tgacChan(txSig);
rxSig = awgn(rxSigNoNoise,15);
```

Introduce a frequency offset of 500 Hz to the received signal.

```
rxSigFreqOffset = pfOffset(rxSig,500);
```

Find the start and stop indices for all component fields of the PPDU.

```
ind = wlanFieldIndices(vht);
```

Extract the L-STF. Estimate and correct for the carrier frequency offset.

```
rxLSTF = rxSigFreqOffset(ind.LSTF(1):ind.LSTF(2),:);
```

```
foffset1 = wlanCoarseCFOEstimate(rxLSTF,cbw);
rxSig1 = pfOffset(rxSigFreqOffset,-foffset1);
```

Extract the L-LTF from the corrected signal. Estimate and correct for the residual frequency offset.

```
rxLLTF = rxSig1(ind.LLTF(1):ind.LLTF(2),:);
```

```
foffset2 = wlanFineCFOEstimate(rxLLTF,cbw);
rxSig2 = pfOffset(rxSig1,-foffset2);
```

Extract and demodulate the VHT-LTF. Estimate the channel coefficients.

```
rxVHTLTF = rxSig2(ind.VHTLTF(1):ind.VHTLTF(2),:);
demodVHTLTF = wlanVHTLTFDemodulate(rxVHTLTF,vht);
chEst = wlanVHTLTFChannelEstimate(demodVHTLTF,vht);
```

Extract the VHT data field from the received and frequency-corrected PPDU. Recover the data field.

```
rxData = rxSig2(ind.VHTData(1):ind.VHTData(2),:);
rxPSDU = wlanVHTDataRecover(rxData,chEst,0.03,vht);
```

Calculate the number of bit errors in the received packet.

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

```
numErr = 2
```

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 a function 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 multiuser 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 multiuser 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 multiuser 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 1-3