Agilent Technologies

# Advanced Design System 2011.01

Feburary 2011

WLAN Design Library

**© Agilent Technologies, Inc. 2000-2011**
5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA
No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

**Acknowledgments**

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXlm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 http://www.xs4all.nl/~kholwerd/bool.html . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at http://www.mozilla.org/MPL/ . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", http://www.cs.umn.edu/~metis , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, http://www.intel.com/software/products/mkl

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program.All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: http://www.7-zip.org/

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: http://www.cise.ufl.edu/research/sparse/amd

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: http://www.cise.ufl.edu/research/sparse/umfpack UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at http://www.cise.ufl.edu/research/sparse . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at http://www.cise.ufl.edu/research/sparse . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. http://www.mathworks.com . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at http://www.netlib.org ). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the
the
terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User
documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: http://www.qtsoftware.com/downloads Patches Applied to Qt can be found in the installation at: $HPEESOF_DIR/prod/licenses/thirdparty/qt/patches. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

**Errata** The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

**Warranty** The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

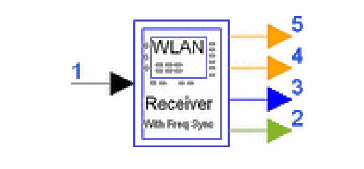
**Technology Licenses** The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at http://systemc.org/ . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

**Restricted Rights Legend** U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

# 11a Receivers

- *WLAN 80211a RF RxFSync* (wlan)
- *WLAN 80211a RF RxNoFSync* (wlan)
- *WLAN 80211a RF Rx Soft* (wlan)
- *WLAN 80211aRxFSync* (wlan)
- *WLAN 80211aRxFSync1* (wlan)
- *WLAN 80211aRxNoFSync* (wlan)
- *WLAN 80211aRxNoFSync1* (wlan)
- *WLAN 80211aRx Soft* (wlan)
- *WLAN BurstSync* (wlan)
- *WLAN ChEstimator* (wlan)
- *WLAN FineFreqSync* (wlan)
- *WLAN FreqSync* (wlan)
- *WLAN OFDMEqualizer* (wlan)
- *WLAN PhaseEst* (wlan)
- *WLAN PhaseTrack* (wlan)
- *WLAN RmvNullCarrier* (wlan)

# WLAN_80211a_RF_RxFSync



**Description Receiver of IEEE 802.11a with full frequency synchronization**
**Library WLAN, Receiver**
**Class TSDFWLAN_80211a_RF_RxFSync**
**Derived From WLAN_ReceiverBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp |  | real | [-273.15, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 |  | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 |  | real | (-∞, ∞) |
| RefFreq | internal reference frequency | 5200MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 |  | real | (-∞, ∞) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| Length | octet number of PSDU | 256 |  | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 |  | enum |  |
| Order | FFT points=2^Order | 6 |  | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 |  | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 |  | enum |  |
| GuardInterval | guard interval defined by user | 16 |  | int | [0, $2^{Order}$] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 |  | int | [0, ∞) |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

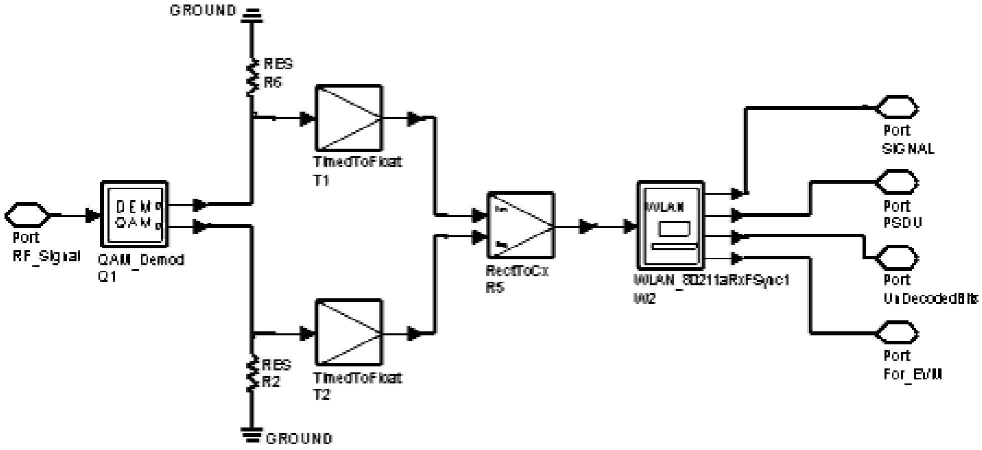
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

## Notes/Equations

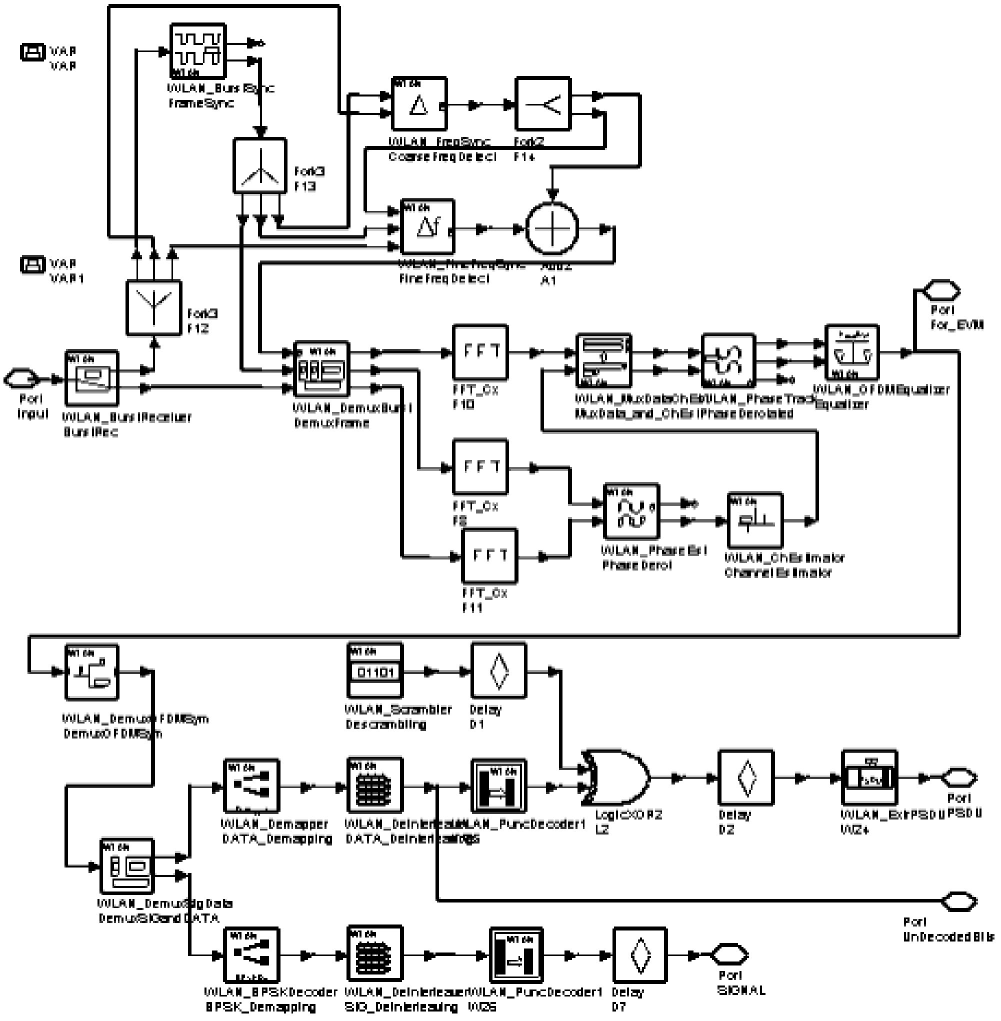
1. This WLAN receiver provides full frequency synchronization according to the IEEE 802.11a Standard; it can be configured in a top-level design using model parameters.
   This subnetwork integrates an RF demodulator and baseband receiver. The schematic is shown in the following figure.



**WLAN_80211a_RF_RxFSync Schematic**

2. Receiver functions are implemented as specified in the IEEE 802.11a Standard.
   - Start of frame is detected.
   - The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
   - Coarse and fine frequency offsets are estimated.
   - The packet is derotated according to estimated frequency offset (coarse and fine frequency synchronization).
   - Complex channel response coefficients are estimated for each subcarrier (channel estimation).
   - Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase, and each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization).
   - The equalized signal is demultiplexed into SIGNAL and PSDU parts.
   - SIGNAL and PSDU are demapped, deinterleaved and decoded, respectively.
   - Demodulated SIGNAL and PSDU bits are output.
   - The equalized receiver signal is output for EVM measurement.
   - The deinterleaved PSDU signal is output, which is the signal before decoding.

- The WLAN_80211aRxFSync1 receiver schematic is shown in the following figure.

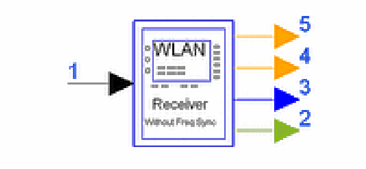


**WLAN_80211aRxFSync1 Schematic**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211a_RF_RxNoFSync



**Description** Receiver of IEEE 802.11a without frequency synchronization
**Library** WLAN, Receiver
**Class** TSDFWLAN_80211a_RF_RxNoFSync
**Derived From** WLAN_ReceiverBase

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp | | real | [-273.15, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| RefFreq | internal reference frequency | 5200MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 | | real | (-∞, ∞) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$] |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

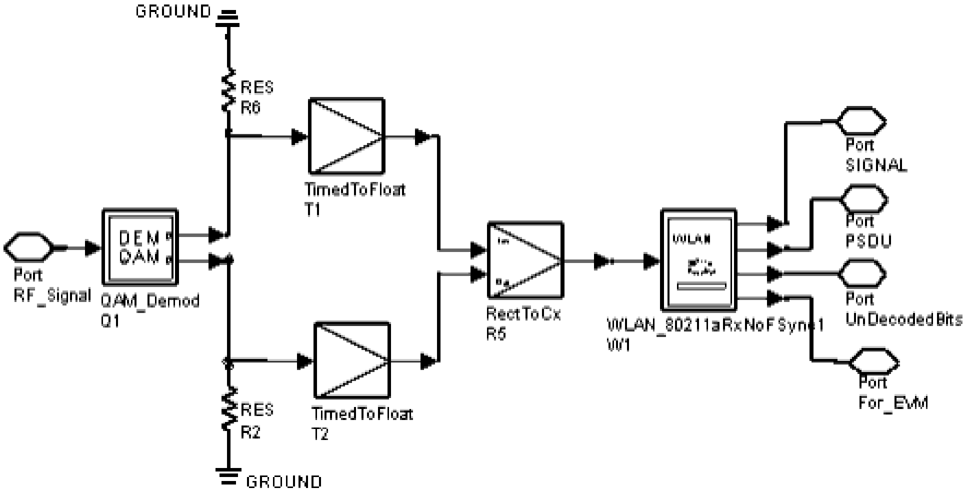
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

### Notes/Equations

1. This WLAN receiver, without frequency synchronization, is according to the IEEE 802.11a Standard; it can be configured in a top-level design using model parameters.
   This subnetwork integrates an RF demodulator and baseband receiver. The schematic is shown in the following figure.

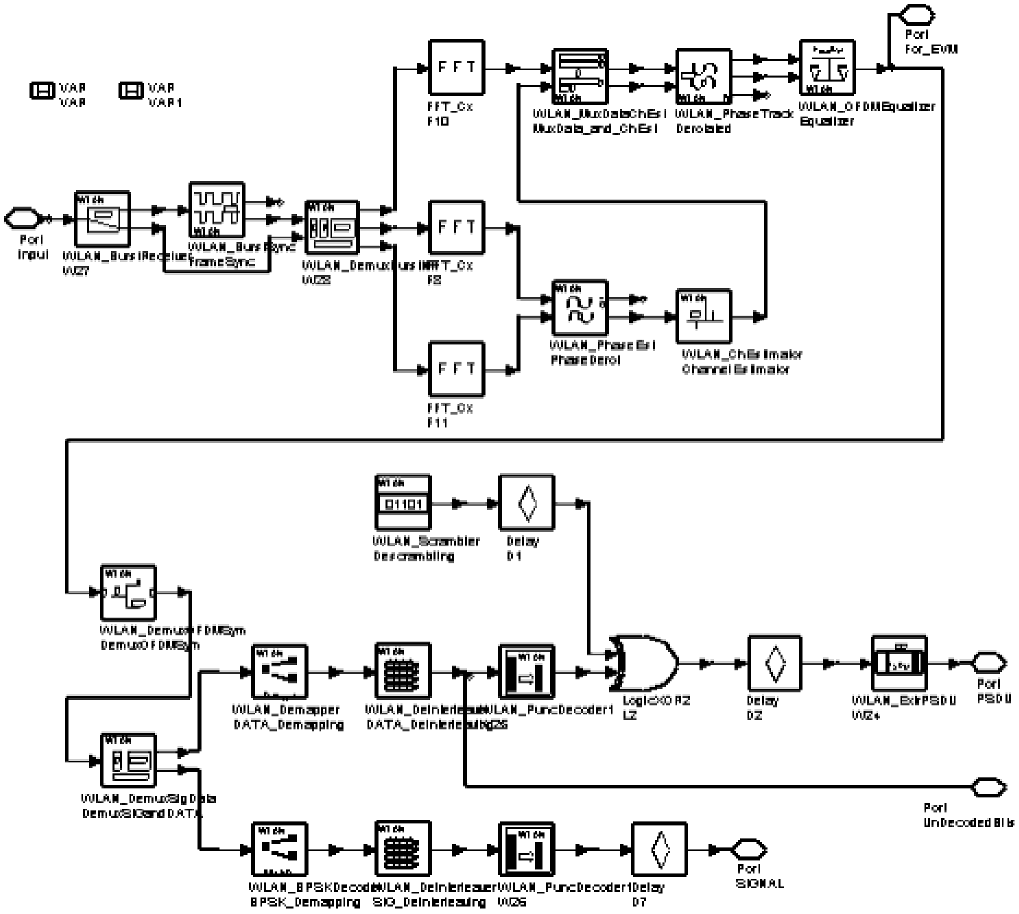


**WLAN_80211a_RF_RxNoFSync Schematic**

Receiver functions are implemented as specified in the IEEE 802.11a Standard.

- Start of frame is detected.
- The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
- Complex channel response coefficients are estimated for each subcarrier (channel estimation).
- Each data OFDM symbol will be transformed into subcarrier received values; pilot subcarrier phases will be estimated; subcarrier values will be derotated according to estimated phase; and, each subcarrier value will be divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization).
- The equalized signal is demultiplexed into SIGNAL and PSDU parts.
- SIGNAL and PSDU are demapped, deinterleaved and decoded, respectively.
- Demodulated SIGNAL and PSDU bits are output.
- The equalized receiver signal is output for EVM measurement.
- The deinterleaved PSDU signal is output, which is the signal before decoding.

- The WLAN_80211aRxNoFSync1 schematic is shown in the following figure.

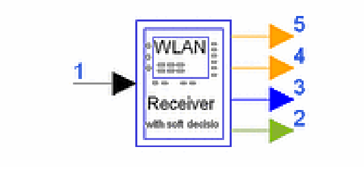


**WLAN_80211aRxNoFSync1 Schematic**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211a_RF_Rx_Soft



**Description Receiver of IEEE 802.11a with full frequency synchronization**
**Library WLAN, Receiver**
**Class TSDFWLAN_80211a_RF_Rx_Soft**
**Derived From WLAN_ReceiverBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp | | real | [-273.15, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| RefFreq | internal reference frequency | 5200MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 | | real | (-∞, ∞) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| DecoderType | demapping type: Hard, Soft, CSI | CSI | | enum | |
| TrunLen | path memory truncation length | 60 | | int | [20, 200] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

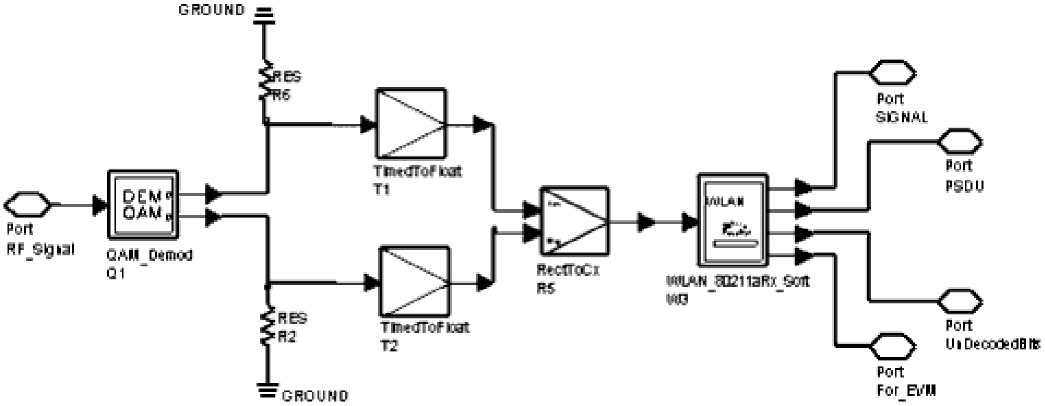
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

### Notes/Equations

1. This WLAN receiver provides full-frequency synchronization according to the IEEE 802.11a Standard. It can be configured in a top-level design using model parameters.
   This subnetwork integrates an RF demodulator and baseband receiver. The schematic is shown in the following figure.



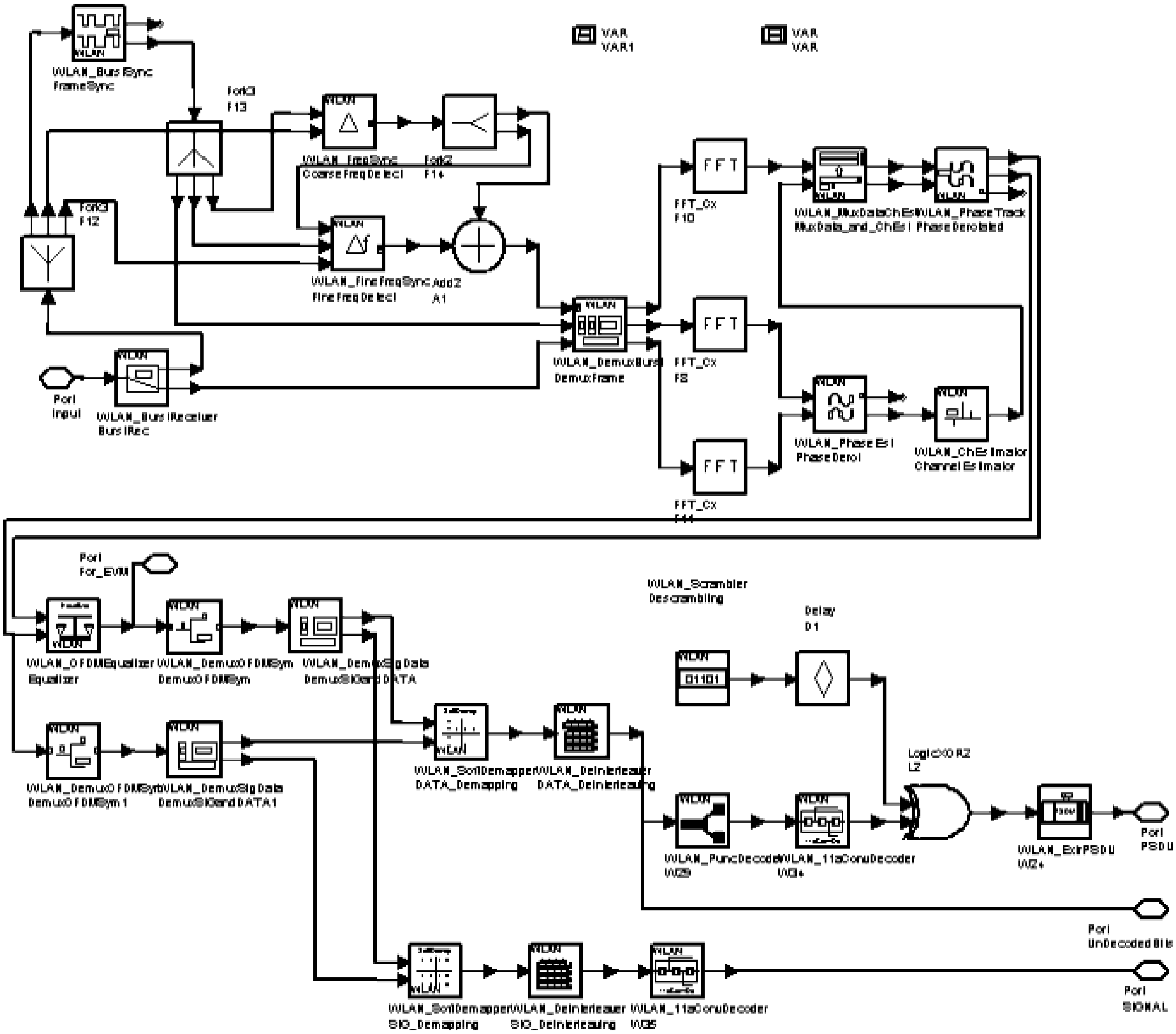
**WLAN_80211a_RF_Rx_Soft Schematic**

2. Receiver functions are implemented as specified in the IEEE 802.11a Standard.
   - Start of frame is detected.
   - The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
   - Coarse and fine frequency offsets are estimated.
   - The packet is derotated according to estimated frequency offset (coarse and fine frequency synchronization).
   - Complex channel response coefficients are estimated for each subcarrier (channel estimation).
   - Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase, and each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization).
   - The equalized signal is demultiplexed into SIGNAL and PSDU parts.
   - SIGNAL and PSDU are demapped, deinterleaved and decoded. A soft viterbi decoding scheme is used in which the received complex symbols are demapped into soft bit information that is weighted by the channel response coefficient then fed to a conventional soft binary viterbi decoder. Viterbi algorithm finds the path that maximizes:

$$\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}\sum_{m=0}^{M-1}\left|H_k\right|^2 c^{nrz}_{l,k,m}\, x^{soft}_{l,k,m}$$ .

- Demodulated SIGNAL and PSDU bits are output.
- The equalized receiver signal is output for EVM measurement.
- The deinterleaved PSDU signal is output, which is the signal before decoding.

- The WLAN_80211aRx_Soft receiver schematic is shown in the following figure.



**WLAN_80211aRx_Soft Schematic**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. M.R.G. Butler, S. Armour, P.N. Fletcher, A.R. Nix, D.R. Bull, "Viterbi Decoding Strategies for 5 GHz Wireless LAN Systems," VTC 2001 Fall. IEEE VTS 54th.

# WLAN_80211aRxFSync

**Description Receiver of IEEE 802.11a with full frequency synchronization**
**Library WLAN, Receiver**
**Class SDFWLAN_80211aRxFSync**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be demodulated | complex |

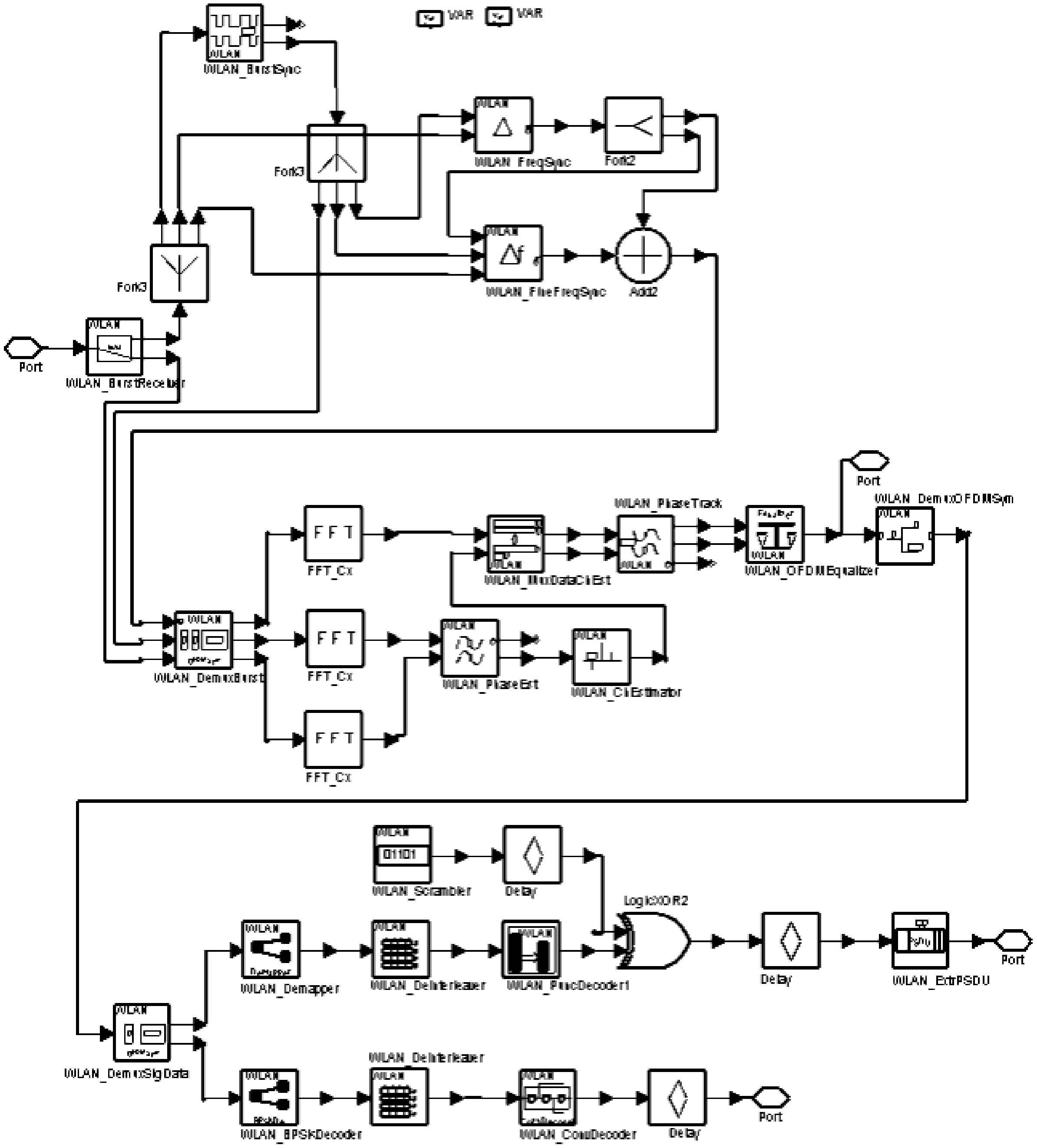
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | SIGNAL | demodulated SIGNAL signal | int |
| 3 | DATA | demodulated DATA signal | int |
| 4 | output | demodulated signal | complex |

## Notes/Equations

1. This subnetwork implements an IEEE 802.11a receiver with full frequency synchronization. Demodulated SIGNAL, DATA, and data are output. The schematic for this subnetwork is shown in the following figure.
2. Receiver functions are implemented according to the IEEE 802.11a Standard.

- Start of frame is detected. WLAN_BurstSync calculates the correlation between the received signal and the 10 short preambles, and selects the index with the maximum correlation value as the start of frame. The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
- Coarse and fine frequency offsets are estimated. WLAN_FreqSync calculates the coarse frequency offset and makes coarse frequency synchronization using the 8th and 9th short preambles. WLAN_FineFreqSync calculates the fine frequency offset and makes fine frequency synchronization using the two long preambles.
- The packet is derotated according to the estimated coarse and fine frequency offsets (coarse and fine frequency synchronization). The phase effect caused by the frequency offset is compensated by WLAN_DemuxBurst. WLAN_DemuxBurst outputs two long preambles and the OFDM symbols for DATA demodulation. The two long preamble outputs are used for channel estimation.
- Complex channel response coefficients are estimated for each subcarrier (channel estimation). The phases of the two long preambles are aligned by WLAN_PhaseEst before the channel estimator. WLAN_ChEstimator performs channel estimation for 52 subcarriers by combining the two long preambles.



**WLAN_80211aRxFSync Schematic**

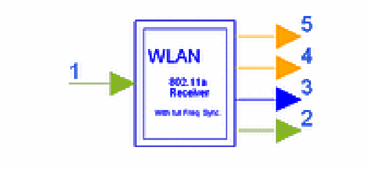
- Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase. WLAN_PhaseTrack implements these functions. WLAN_MuxDataChEst only duplicates the estimated complex channel response coefficients the number of OFDM symbols for DATA and SIGNAL times.
- Each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization). This simple one-tap frequency domain channel response compensation is implemented by WLAN_OFDMEqualizer.
- After equalization, WLAN_DemuxOFDMSym demultiplexes 52 subcarriers into 48 data and 4 pilot subcarriers. The demodulated burst is then demultiplexed into SIGNAL and PSDU parts in WLAN_DemuxSigData.

- SIGNAL and DATA are demapped, deinterleaved and decoded.
- Demodulated SIGNAL and PSDU bits are output.
- The equalized receiver signal is output for EVM measurement.
- The deinterleaved PSDU signal is output, which is the signal before decoding.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211aRxFSync1



**Description Receiver of IEEE 802.11a with full frequency synchronization**
**Library WLAN, Receiver**
**Class SDFWLAN_80211aRxFSync1**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{Order}]$ |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be demodulated | complex |

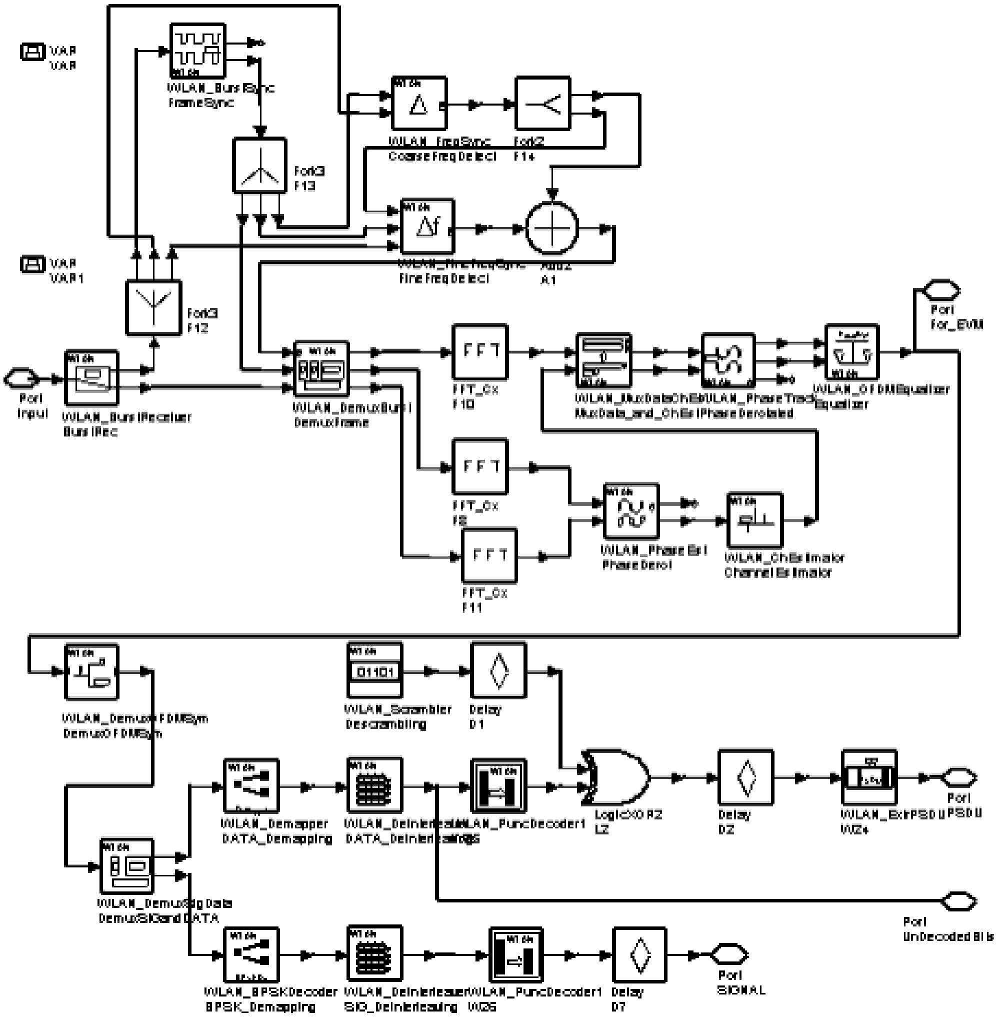
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

## Notes/Equations

1. The model for this subnetwork is based on an IEEE 802.11a receiver with full frequency synchronization. The schematic is shown in the following figure.

2. Receiver functions are implemented according to the IEEE 802.11a Standard.
    - Start of frame is detected. WLAN_BurstSync calculates the correlation between the received signal and the 10 short preambles, and selects the index with the maximum correlation value as the start of frame. The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
    - Coarse and fine frequency offsets are estimated. WLAN_FreqSync calculates the coarse frequency offset and makes coarse frequency synchronization using the 8th and 9th short preambles. WLAN_FineFreqSync calculates the fine frequency offset and makes fine frequency synchronization using the two long preambles.
    - The packet is derotated according to the estimated coarse and fine frequency offsets (coarse and fine frequency synchronization). The phase effect caused by the frequency offset is compensated by WLAN_DemuxBurst. WLAN_DemuxBurst outputs two long preambles and the OFDM symbols for DATA demodulation. The two long preamble outputs are used for channel estimation.
    - Complex channel response coefficients are estimated for each subcarrier (channel estimation). The phases of the two long preambles are aligned by WLAN_PhaseEst before the channel estimator. WLAN_ChEstimator performs channel estimation for 52 subcarriers by combining the two long preambles.



**WLAN_80211aRxFSync1 Schematic**

- Each data OFDM symbol is transformed into subcarrier received values, pilot

subcarrier phases are estimated, subcarrier values are derotated according to estimated phase. WLAN_PhaseTrack implements these functions. WLAN_MuxDataChEst only duplicates the estimated complex channel response coefficients the number of OFDM symbols for DATA and SIGNAL times.
- Each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization). This simple one-tap frequency domain channel response compensation is implemented by WLAN_OFDMEqualizer.
- After equalization, WLAN_DemuxOFDMSym demultiplexes 52 subcarriers into 48 data and 4 pilot subcarriers. The demodulated burst is then demultiplexed into SIGNAL and PSDU parts in WLAN_DemuxSigData.
- SIGNAL and DATA are demapped, deinterleaved and decoded.
- Demodulated SIGNAL and PSDU bits are output.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211aRxNoFSync

**Description Receiver of IEEE 802.11a without frequency synchronization**
**Library WLAN, Receiver**
**Class SDFWLAN_80211aRxNoFSync**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | int | [0, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

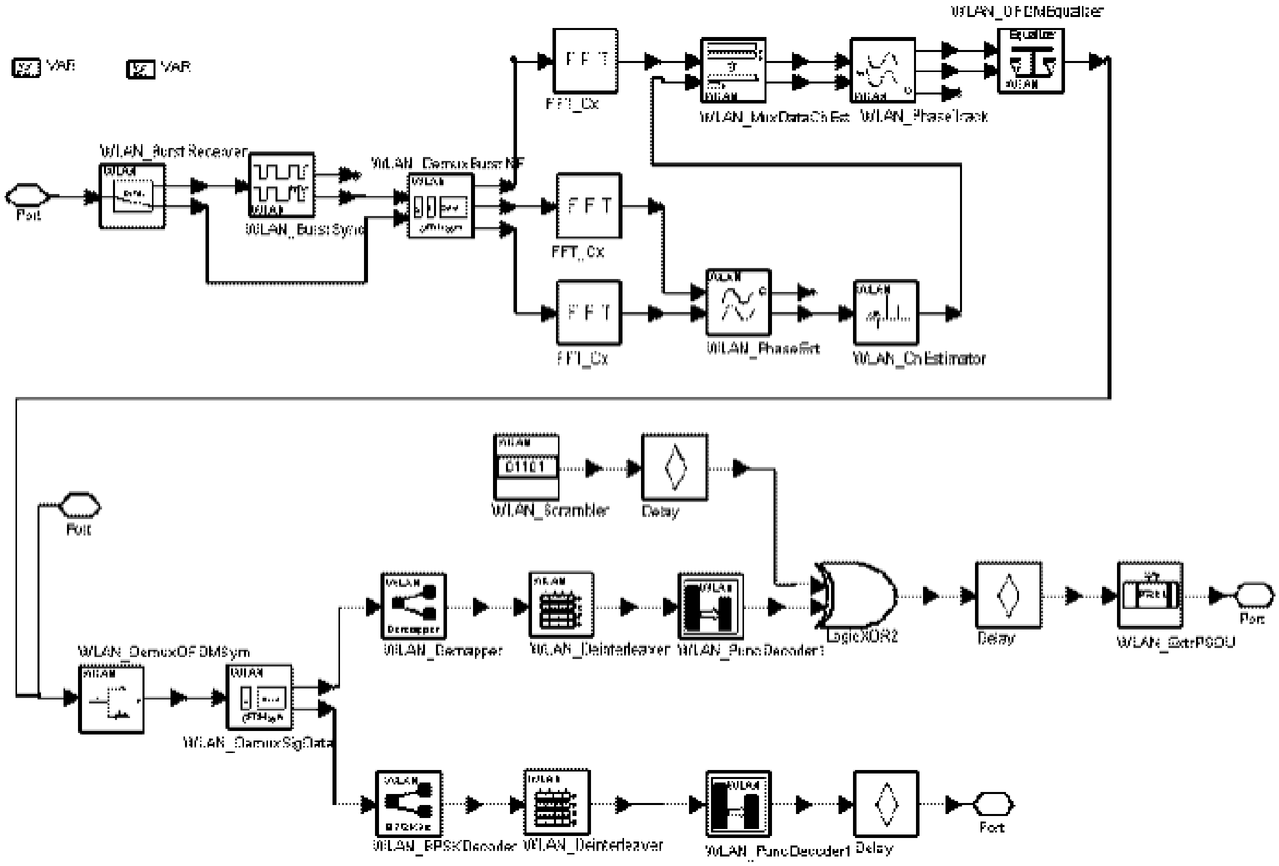
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | SIGNAL | demodulated SIGNAL signal | int |
| 3 | DATA | demodulated DATA signal | int |
| 4 | output | demodulated signal | complex |

## Notes/Equations

1. This subnetwork model implements a 802.11a receiver without frequency synchronization. Demodulated SIGNAL, DATA, and data are output. The schematic for this subnetwork is shown in the following figure.



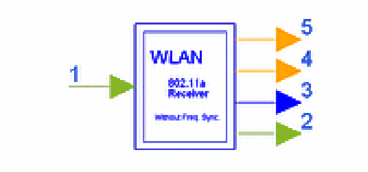
**WLAN_80211aRxNoFSync Schematic**

2. Receiver functions are implemented according to the IEEE 802.11a Standard.
    - Start of frame is detected.
    - Transition from short to channel estimation sequences are detected, and time (with one sample resolution) will be established (burst synchronization).
    - Complex channel response coefficients are estimated for each subcarrier (channel estimation).
    - Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase, and each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization).
    - The equalized signal is demultiplexed into SIGNAL and DATA parts.
    - SIGNAL and DATA are demapped, deinterleaved and decoded, respectively.
    - Demodulated SIGNAL and DATA bits are output.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211aRxNoFSync1



**Description Receiver of IEEE 802.11a without frequency synchronization**
**Library WLAN, Receiver**
**Class SDFWLAN_80211aRxNoFSync1**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | int | [0, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

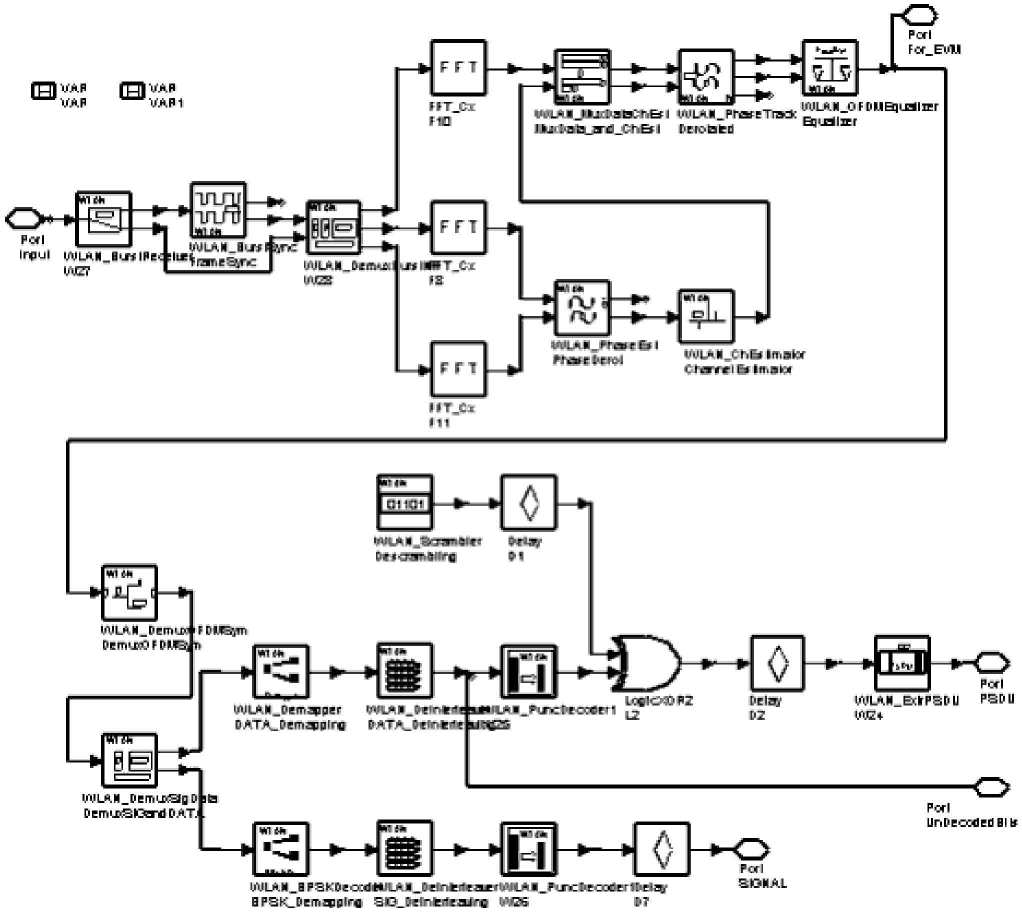
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

## Notes/Equations

1. The model for this subnetwork is based on an IEEE 802.11a receiver without frequency synchronization. The schematic is shown in the following figure.



**WLAN_80211aRxNoFSync1 Subnetwork**

Receiver functions are implemented as specified in the IEEE 802.11a Standard.

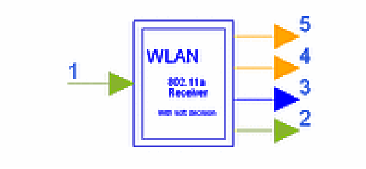
- Start of frame is detected.
- The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
- The complex channel response coefficients are estimated for each subcarrier (channel estimation).
- Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase, and each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization).
- The equalized signal is demultiplexed into SIGNAL and PSDU parts.
- SIGNAL and PSDU are demapped, deinterleaved and decoded, respectively.
- Demodulated SIGNAL and PSDU bits are output.
- The equalized receiver signal is output for EVM measurement.
- The deinterleaved PSDU signal is output, which is the signal before decoding.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211aRx_Soft



**Description Receiver of IEEE 802.11a with full frequency synchronization**
**Library WLAN, Receiver**
**Class SDFWLAN_80211aRx_Soft**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{Order}]$ |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| DecoderType | demapping type: Hard, Soft, CSI | CSI | | enum | |
| TrunLen | path memory truncation length | 60 | | int | [20, 200] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

## Pin Inputs

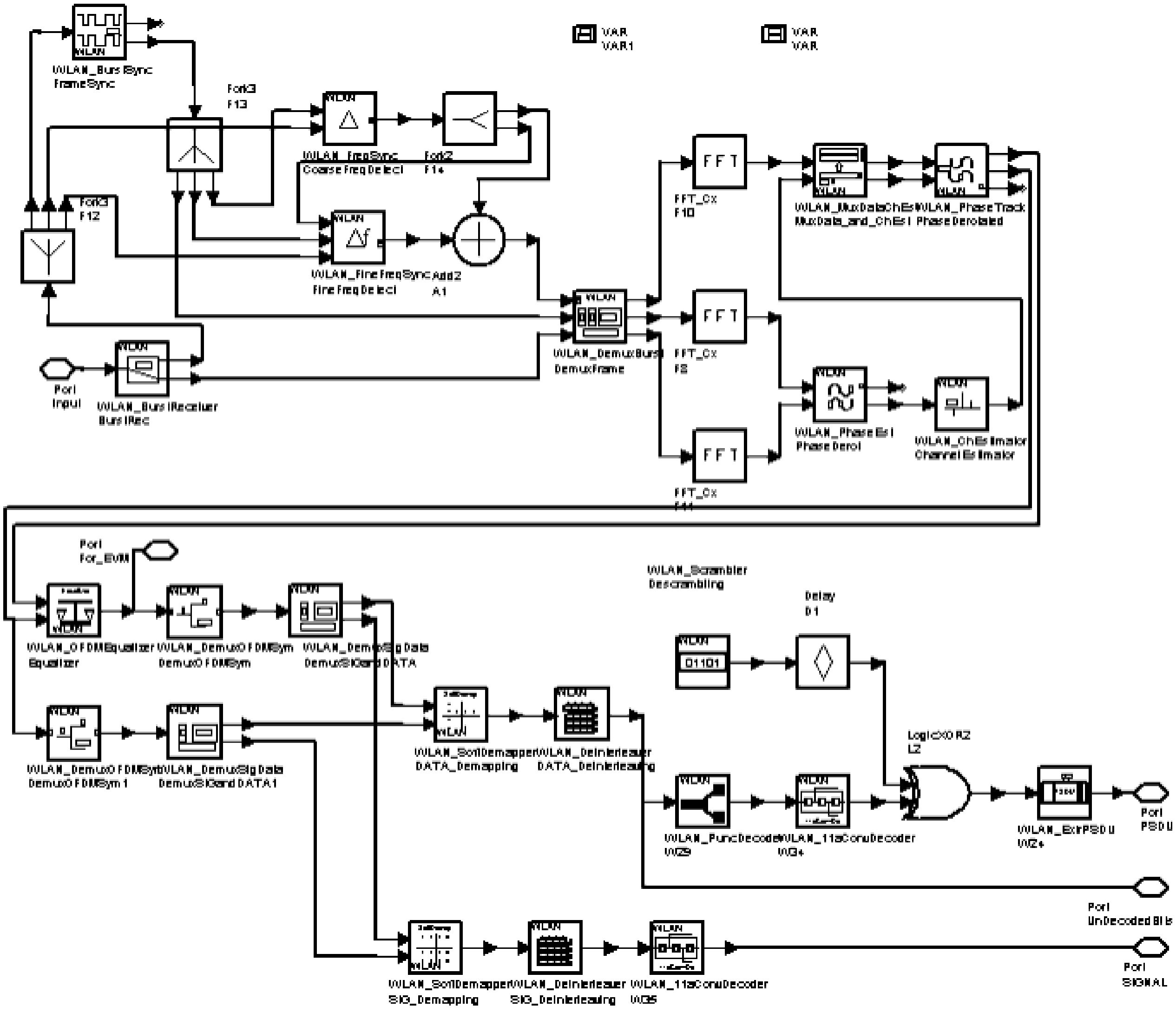
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | For_EVM | undemapped signal after FFT used for EVM | complex |
| 3 | UnDecodedBits | deinterleaved data bits before decoding | real |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

## Notes/Equations

1. This subnetwork implements an IEEE 802.11a baseband receiver with soft Viterbi decoding algorithm. The schematic is shown in the following figure.



**WLAN_80211aRx_Soft Schematic**

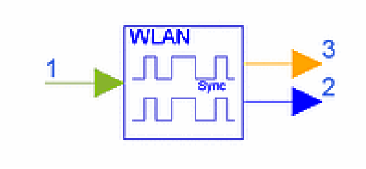
2. Receiver functions are implemented as specified in the IEEE 802.11a Standard.
    - Start of frame is detected. WLAN_BurstSync calculates the correlation between the received signal and the 10 short preambles, and selects the index with the maximum correlation value as the start of frame. The transition from short to channel estimation sequences is detected and time (with one sample resolution) is established (burst synchronization).
    - Coarse and fine frequency offsets are estimated. WLAN_FreqSync calculates the coarse frequency offset and makes coarse frequency synchronization using the 8th and 9th short preambles. WLAN_FineFreqSync calculates the fine frequency offset and makes fine frequency synchronization using the two long preambles.
    - The packet is derotated according to the estimated coarse and fine frequency offsets (coarse and fine frequency synchronization). The phase effect caused by the frequency offset is compensated by WLAN_DemuxBurst. WLAN_DemuxBurst outputs two long preambles and the OFDM symbols for DATA demodulation. The two long preamble outputs are used for channel estimation.
    - Complex channel response coefficients are estimated for each subcarrier

(channel estimation). The phases of the two long preambles are aligned by WLAN_PhaseEst before the channel estimator. WLAN_ChEstimator performs channel estimation for 52 subcarriers by combining the two long preambles.
- Each data OFDM symbol is transformed into subcarrier received values, pilot subcarrier phases are estimated, subcarrier values are derotated according to estimated phase. WLAN_PhaseTrack implements these functions. WLAN_MuxDataChEst only duplicates the estimated complex channel response coefficients the number of OFDM symbols for DATA and SIGNAL times.
- Each subcarrier value is divided with a complex estimated channel response coefficient (phase tracking, phase synchronization, and equalization). This simple one-tap frequency domain channel response compensation is implemented by WLAN_OFDMEqualizer.
- After equalization, WLAN_DemuxOFDMSym demultiplexes 52 subcarriers into 48 data and 4 pilot subcarriers. The demodulated burst is then demultiplexed into SIGNAL and PSDU parts in WLAN_DemuxSigData.
- The demodulated SIGNAL and DATA (such as QPSK, 16-QAM, and 64-QAM modulation) are demapped by WLAN_SoftDemapper that has three modes:
  - when DecoderType = Hard, if $b < 0$, -1.0 is output, otherwise 1.0 is output
  - when DecoderType = Soft, if $b < -1.0$, -1.0 is output; if $b > 1.0$, 1.0 is output
  - when DecoderType = CSI, b is multiplied by CSI ($= |H(i)|^2$) and output. Estimated channel impulse responses (H(i)) in WLAN_ChEstimator is the CSI ( *channel status information* ) here [2].
- The demapped SIGNAL and DATA bits are deinterleaved and decoded.
- Demodulated SIGNAL and PSDU bits are output.
- The equalized receiver signal (burst) is output for the EVM measurement.
- The deinterleaved PSDU signal is output, which is the signal before decoding.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. M.R.G. Butler, S. Armour, P.N. Fletcher, A.R. Nix, D.R. Bull, "Viterbi Decoding Strategies for 5 GHz Wireless LAN Systems," VTC 2001 Fall. IEEE VTS 54th.

# WLAN_BurstSync



**Description Burst synchronizer**
**Library WLAN, Receiver**
**Class SDFWLAN_BurstSync**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | int | [0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signals for synchronization | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | correlation for OFDM symbol synchronization | real |
| 3 | index | synchronization index | int |

## Notes/Equations

1. This model is used to calculate the correlation of the input signal that is used in OFDM system timing synchronization.
   Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols, $N_{SYM}$ is:

   $$N_{SYM} = Ceiling\ ((16 + 8 \times \ Length + 6) / N_{DBPS})$$

   where $N_{DBPS}$ is determined by data rate according to the following table.

   **Rate-Dependent Values**

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

After determining $N_{SYM}$, the number of input tokens $N_{total}$ can be calculated:

$$N_{total} = (2^{Order} + 2^{Order - 2}) \times 4 + (2^{Order} + GI) \times (N_{SYM} + 1) + Idle$$

where *idle* is Idle parameter; and GI (GuardInterval parameter) is defined as:

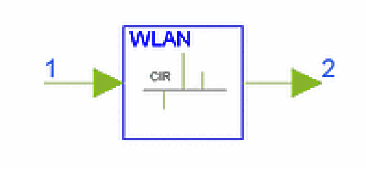
if GuardType=T/32, $GI = 2^{Order - 5}$
if GuardType=T/16, $GI = 2^{Order - 4}$
if GuardType=T/8, $GI = 2^{Order - 3}$
if GuardType=T/4, $GI = 2^{Order - 2}$
if GuardType=T/2, $GI = 2^{Order - 1}$
if GuardType=UserDefined, GI is determined by GuardInterval.

2. 10 short preambles are used to generate the correlation values for burst synchronization. $(2^{Order} + 2^{Order-2}) \times 4$ + Idle correlation values are calculated and the maximum value is selected; the index for synchronization corresponds to the maximum correlation value. The $(2^{Order} + 2^{Order-2}) \times 4$ + Idle correlation values are output at *output* pin 2, the synchronization index is output at *index* pin 3. The maximum delay range detected by this model is $(2^{Order} + 2^{Order-2}) \times 4$ + Idle.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_ChEstimator



**Description** **Channel estimator**
**Library** **WLAN, Receiver**
**Class** **SDFWLAN_ChEstimator**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | output signals from FFT | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | Coef | channel coefficient in active subcarriers | complex |

## Notes/Equations

1. The model is used to calculate channel estimation based on the pilot channel and output the active subcarriers estimated channel impulse response (CIR).
2. This model uses long preambles to estimate the CIRs. The estimated CIRs are calculated using active subcarrier pilot channels. The long training symbol include 52 subcarriers, given by

   $L_{0, \dots, 51}$ = {1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,
   1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1}

   Set $x_0, x_1, \dots, x_{51}$ are the input signals, $h_0, h_1, \dots, h_{51}$ are the estimated CIR. The estimated CIR can be calculated as follows:

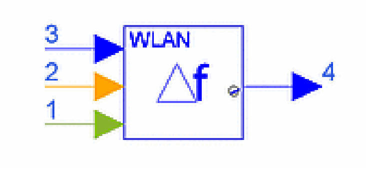
   $$h_i = \frac{x_i}{L_i}$$

   where $i = 0, 1, \dots, 51$.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_FineFreqSync



**Description Fine carrier frequency synchronizer**
**Library WLAN, Receiver**
**Class SDFWLAN_FineFreqSync**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Length | octet number of PSDU | 256 | | int | (0, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal for fine frequency synchronization | complex |
| 2 | index | synchronization index | int |
| 3 | CoarseF | coarse carrier frequency offset | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | FineF | fine carrier frequency offset | real |

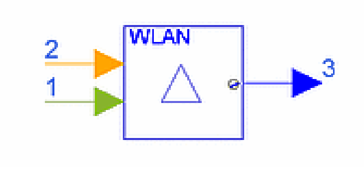
## Notes/Equations

1. This model is used to estimate and output the fine carrier frequency offset between transmitter and receiver after coarse carrier frequency offset detection.
2. Two long preambles are used to calculate the fine carrier frequency offset between transmitter and receiver after coarse carrier frequency offset detection. The WLAN_DemuxBurst model will use the coarse and fine carrier frequency offsets detected in WLAN_FreqSync and WLAN_FineFreqSync models to remove carrier frequency offset in the receiver.
   Input index pin 2 determines the starting point of the two long preambles. The coarse frequency offset in CoarseF pin3 is used to derotate the preambles. A maximum likelihood algorithm is used to estimate the fine carrier frequency offset.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_FreqSync



**Description Coarse carrier frequency synchronizer**
**Library WLAN, Receiver**
**Class SDFWLAN_FreqSync**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Length | octet number of PSDU | 256 | | int | (0, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal for frequency synchronization | complex |
| 2 | index | synchronization index | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | CoarseF | coarse carrier frequency offset | real |

## Notes/Equations

1. This model is used to calculate the carrier frequency offset between the transmitter and the receiver and output the coarse carrier frequency offset.
2. Two short preambles ( *t* 9 and *t* 10) are used to calculate the carrier frequency offset; WLAN_DemuxBurst will use this coarse carrier frequency offset to remove it in the receiver.
   Input at *index* pin 2 is used to determine the starting point of the 10 short preambles. The maximum likelihood algorithm is used to calculate the offset.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_OFDMEqualizer

**Description OFDM equalizer by the channel estimation**
**Library WLAN, Receiver**
**Class SDFWLAN_OFDMEqualizer**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of active carriers in one OFDM symbol | 52 | int | (0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | data in the active carriers in OFDM symbol | complex |
| 2 | Coef | frequency channel impulse response(CIR) estimation | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | output data after channel equalization | complex |

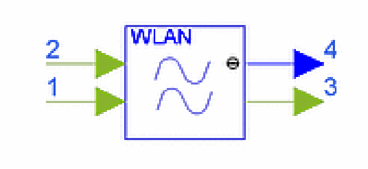
## Notes/Equations

1. This model is used to perform channel equalization using the channel estimation in each active carrier.
2. The OFDM channel equalization algorithm is:
$$a(i) = \frac{x(i)}{h(i)}$$
where *h* ( *i* ) is the channel estimation, *x* ( *i* ) is the received signal in active carriers, *a* ( *i* ) is the equalized output signal.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_PhaseEst



**Description Phase estimator**
**Library WLAN, Receiver**
**Class SDFWLAN_PhaseEst**

### Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | LPrmbl1 | first long preamble signals from FFT | complex |
| 2 | LPrmbl2 | second long preamble signals from FFT | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | channel coefficient in active subcarriers | complex |
| 4 | theta | phase difference between two long preambles | real |

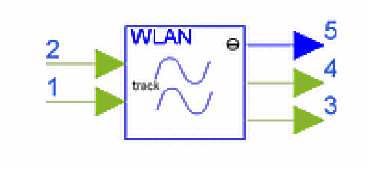
### Notes/Equations

1. This model is used to estimate the phase difference between the two long input preambles.
2. According to IEEE 802.11a standard, there are two long preambles in every burst, which is used for channel estimation. The maximum likelihood algorithm is used to calculate the phase offset between the two long preambles. The detected phase offset is used to correct the second long preamble so that both long preambles have the same phase. A combined long preamble is then output and used in the WLAN_ChEstimator model.
   The detected phase offset is output at *theta* pin 4.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_PhaseTrack



**Description Phase tracker in OFDM de-modulation**
**Library WLAN, Receiver**
**Class SDFWLAN_PhaseTrack**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| Phase | initial phase of pilots | 0 | int | [0, 126] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | all sub-carriers in one OFDM symbol | complex |
| 2 | chl | estimated channel impulse response | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | active sub-carriers after removing null sub-carriers | complex |
| 4 | Coef | channel coefficient in active subcarriers | complex |
| 5 | theta | phase difference between current CIR and estimated CIR | real |

## Notes/Equations

1. The model is used to track the phase caused by doppler shift in OFDM demodulation systems, remove the null carrier in one OFDM symbol, and update the estimated CIR using the phase offset detected in the phase tracking algorithm.
2. According to IEEE 802.11a standard, the positions from 27 to 37 and the 0 position are set to zero. These 12 subcarriers are set to zero in the WLAN_LoadIFFTBuff model. In the receiver, these 12 zero subcarriers will be removed which is the inverse procedure of WLAN_LoadIFFTBuff.
   Signals from *chl* pin 2 are output directly at *Coef* pin 4. The 12 zero subcarrier signals will be removed from 64 point signals and form 52 active subcarriers signals that are output at *output* pin 3.
   Assume

   $x(0), x(1), \ldots, x(2^{Order}-1)$ are input signals
   $y(0), y(1), \ldots, y(51)$ are output signals.

Then

$$y(i) = x(2^{Order} - 26 + i) \quad i = 0, 1, \ldots, 25$$
$$y(i+26) = x(i+1) \quad i = 0, 1, \ldots, 25$$

At the same time, this model uses the four pilots to obtain the estimated CIR of the four subcarriers. The maximum likelihood algorithm is used to detect the phase offset θ between input *chl* pin 2 and the current estimated CIR. The phase offset θ is output at *theta* pin 5. The estimated CIRs from input *chl* pin 2 are updated by phase offset θ .

Set $h_0, h_1, \ldots, h_{51}$ and $h'_0, h'_1, \ldots, h'_{51}$ are the input estimated and updated CIRs, respectively.

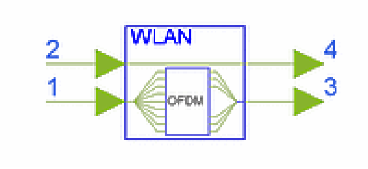
$$h'_i = h_i \times e^{j\theta}$$

The updated CIRs $h'_0, h'_1, \ldots, h'_{51}$ are output at *Coef* pin 4.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_RmvNullCarrier



**Description Null sub-carriers remover in OFDM**
**Library WLAN, Receiver**
**Class SDFWLAN_RmvNullCarrier**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | all sub-carriers in one OFDM symbol | complex |
| 2 | chl | estimated channel impulse response | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | active sub-carriers after removing null sub-carriers | complex |
| 4 | Coef | channel coefficient in active subcarriers | complex |

## Notes/Equations

1. This model is used to remove the null carrier in one OFDM symbol. (It does not have the phase tracking functionality of the WLAN_PhaseTrack model.)
2. According to IEEE 802.11a standard, the 27 to 37 and the 0 positions are set to zero; these 12 subcarriers are set to zero in the WLAN_LoadIFFTBuff model. In the receiver, these zero subcarriers will be removed (the inverse procedure of WLAN_LoadIFFTBuff).
   Input *chl* pin 2 signals are output directly at *Coef* pin 4. 12 zero subcarrier signals will be removed from the 64 point signals to form 52 active subcarriers signals output at *output* pin 3.
   Assume

   $x(0), x(1), \ldots, x(2^{Order}-1)$ are input signals
   $y(0), y(1), \ldots, y(51)$ are output signals.

   Then

   $$y(i) = x(2^{Order} - 26 + i) \quad i = 0, 1, \ldots, 25$$
   $$y(i+26) = x(i+1) \quad i = 0, 1, \ldots, 25$$

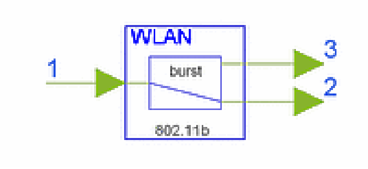
## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# 11b Receivers

- *WLAN 11bBurstRec* (wlan)
- *WLAN 11bBurstSync* (wlan)
- *WLAN 11bCIREstimator* (wlan)
- *WLAN 11bDemuxBurst* (wlan)
- *WLAN 11bDescrambler* (wlan)
- *WLAN 11bDFE* (wlan)
- *WLAN 11b Equalizer* (wlan)
- *WLAN 11bFreqEstimator* (wlan)
- *WLAN 11bPreamble* (wlan)
- *WLAN 11bRake* (wlan)
- *WLAN 11b Rake* (wlan)
- *WLAN CCKDemod* (wlan)
- *WLAN CCK RF Rx DFE* (wlan)
- *WLAN CCK RF Rx Rake* (wlan)
- *WLAN CCK Rx DFE* (wlan)
- *WLAN CCK Rx Rake* (wlan)
- *WLAN Despreader* (wlan)
- *WLAN FcCompensator* (wlan)
- *WLAN HeaderDemap* (wlan)
- *WLAN PhaseRotator* (wlan)
- *WLAN PrmblDemap* (wlan)
- *WLAN RecFilter* (wlan)

# WLAN_11bBurstRec



**Description 11b burst receiver**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bBurstRec**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0μsec | sec | real | [10μsec, 1000μsec] |

## Pin Inputs

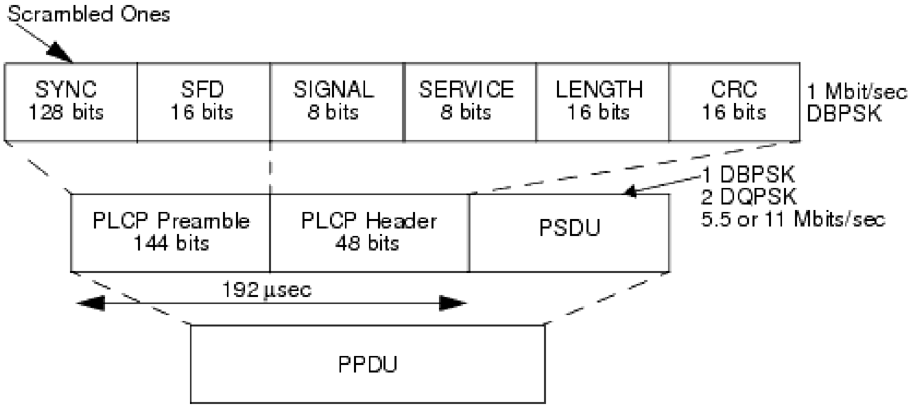
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | input burst data | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | BurstOut | output burst data | complex |
| 3 | SyncOut | output signal for 11b burst and frequency synchronization | complex |

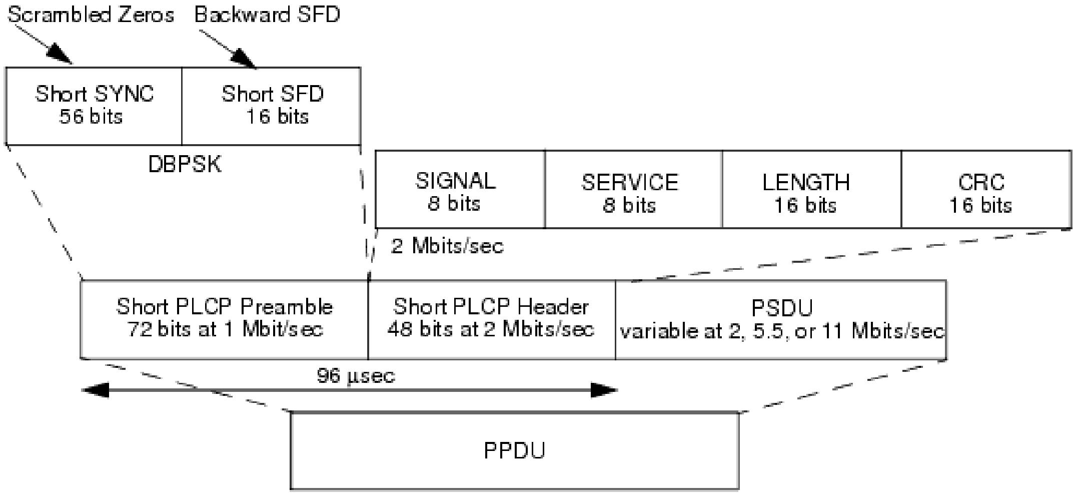
## Notes/Equations

1. This model is used to output signals for 802.11b burst (or timing) synchronization and carrier frequency synchronization, which includes the Preamble.
2. Rate, ModType, PLCPType, Octets, RampTime, PwrType, SampsPerChip and IdleInterval parameters are used to specify the number of complex signals in one burst.
   The following figure illustrates the format for the interoperable (long) PPDU, including the High Rate PLCP preamble, the High Rate PLCP header, and the PSDU.



**Long PLCP PPDU Format**

The following figure illustrates the format of the PPDU, with HR/DSSS/short.



**Short PLCP PPDU Format**

In the WLAN Design Library, one 802.11b burst consists of Idle, Ramp (Transmit power-on and power-down ramp), PLCP Preamble (Long or Short), PLCP Header (Long or Short) and PSDU.

The length of PLCP Preamble (mPLCP) is determined by the PLCPType parameter:

$$mPLCP = \begin{cases} 144 \times 11 \times SampsPerChip & ifPLCPType = Long \\ 72 \times 11 \times SampsPerChip & ifPLCPType = Short \end{cases}$$

The length of Header (mHeader) is determined by the PLCPType parameter:

$$mHeader = \begin{cases} 48 \times 11 \times SampsPerChip & ifPLCPType = Long \\ 24 \times 11 \times SampsPerChip & ifPLCPType = Short \end{cases}$$

The length of PSDU (mPSDU) is determined by the Rate parameter:

$$mPSDU = \begin{cases} Octets \times 8 \times 11 \times SampsPerChip & ifRate = 1Mbps \\ Octets \times 4 \times 11 \times SampsPerChip & ifRate = 2Mbps \\ Octets \times 2 \times 8 \times SampsPerChip & ifRate = 5.5Mbps \\ Octets \times 1 \times 8 \times SampsPerChip & ifRate = 11Mbps \end{cases}$$

The length of Ramp (mRamp) can be calculated as follows:

$$mRamp = \begin{cases} 0 & ifPwrType = None \\ 2 \times RampTime \times 11 \times 10^6 \times SampsPerChip & ifPwrType = Linear, Cosine \end{cases}$$

The length of Idle (mIdle) can be calculated as follows:

$$mIdle = IdleInterval \times 11 \times 10^6 \times SampsPerChip$$

So, the total number of one 802.11b burst is:

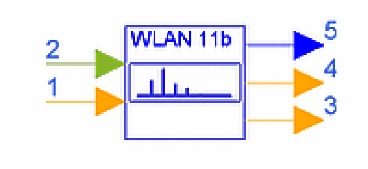
$$Total = mIdle + mRamp + mPLCP + mHeader + mPSDU$$

All input data (BurstIn pin) is output at pin 2 (BurstOut pin).
Pin 3 (SyncOut) outputs the 11b burst and frequency synchronization signal, which includes the PLCP Preamble. In the WLAN Design Library, Idle is added before and after the actual 802.11b burst; the transmit power-on ramp is also added before the actual 802.11b. So, the half Idle and transmit power-on ramp model must be discarded in order to output the PLCP Preamble for burst and frequency synchronization. SyncOut outputs mPLCP+3 × 11 × SampsPerChip samples from mIdle/2+mRamp/2 point in the input burst (BurstIn input pin).
This model is used before WLAN_11bBurstSync and WLAN_FreqEstimator in an 802.11b receiver.

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_11bBurstSync



**Description 11b burst synchronizer**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bBurstSync**
**Derived From WLAN_11bBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | Ref | reference signal for synchronization | int |
| 2 | SyncIn | input signals for synchronization | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | index | finger index | int |
| 4 | max_index | synchronization index | int |
| 5 | corr | correlation for synchronization | real |

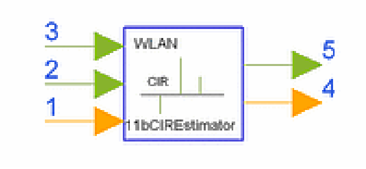
## Notes/Equations

1. This model is used to implement the burst synchronization for 802.11b.
2. The correlation values between the received and the reference signals are calculated for synchronization. The number of preamble symbols used for the correlation calculation is set by SyncEstWindow and the search range is set by MaxSearchDelay. The maximum correlation value is searched for and the corresponding index is selected as the burst start point. The index corresponding to the maximum correlation value is output at pin max_index.
3. All correlation values are output at pin corr. After output, all correlation values are rearranged from maximum to minimum. The corresponding indexes are also rearranged and output at pin index for the multi-path search.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bCIREstimator



**Description 11b channel estimator**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bCIREstimator**
**Derived From WLAN_11bBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| MaxSearchDelay | time range for searching fading path | 3.0usec | sec | real | [3usec, 10usec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| CIREstWindow | number of preamble symbols used to estimate channel | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | Index | samples index of channel delay | int |
| 2 | Ref | preamble symbols used to estimate channel | complex |
| 3 | PLCP | 11b received PLCP signal suffered from channel distortion | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | FingerIdx | samples index of channel delay | int |
| 5 | CIR | channel CIR | complex |

## Notes/Equations

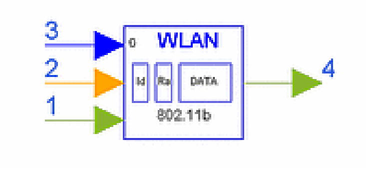
1. This model is used to estimate the fading channel impulse response.
   Each firing, the position of each resolved channel is input from pin Index; the preambles affected by the fading channels are received by pin PLCP; and, the prime preamble bits are received by pin Ref. The channels are then estimated and output at pin CIR.
   Each firing, PLCP, Index and Ref pins consume tokens separately as follows:
   - (MaxSearchDelay × 11E6+PLCPLength × 11) × SampsPerChip
   - (MaxSearchDelay × 11E6) × SampsPerChip
   - PLCPLength is 144 if PCLPType is set to Long, otherwise 72.
     At the same time, FingerNumber tokens are produced from pin CIR and FingerIdx.

2. MaxSearchDelay specifies the range for searching fading channels; the value must be set larger than the sum of system delay and maximum multipath delay.
3. FingerNumber specifies the number of estimated channels output at pin CIR.
4. CIREstWindow specifies the number of preamble symbols used to estimate the CIR of channels.

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_11bDemuxBurst



**Description 11b burst demultiplexer and frequency compensator**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bDemuxBurst**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 |  | enum |  |
| ModType | modulation type: CCK, PBCC | CCK |  | enum |  |
| PLCPType | PLCP preamble type: Long, Short | Long |  | enum |  |
| Octets | octet number of PSDU | 100 |  | int | (0, 2312] |
| RampTime | power on and off ramp time | 2.0µsec | sec | real | [0µsec, 1000µsec] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None |  | enum |  |
| SampsPerChip | number of samples per chip | 2 |  | int | [2, 8] |
| IdleInterval | idle time | 50.0µsec | sec | real | [10µsec, 1000µsec] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | input burst data | complex |
| 2 | Index | burst synchronization index | int |
| 3 | DeltaF | estimated frequency offset | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | BurstOut | output burst data | complex |

## Notes/Equations

1. This model implements compensation for carrier frequency offset. Each firing, the frequency offset detected by WLAN_11bFreqEstimator is input to pin DeltaF, and the burst with frequency offset is input to BurstIn. The burst is then compensated by multiplying exp(-j × 2 × PI × DeltaF × n), where n is the number of burst samples from 1 to entire burst length, and is output to pin BurstOut.
2. Rate, ModType, PLCPType, Octets, RampTime, PwrType, SampsPerChip and IdleInterval parameters are used to specify the number of complex signals in one burst. In the WLAN Design Library, one 802.11b burst consists of Idle, Ramp (transmit power-on and power-down ramp), PLCP Preamble (Long or Short), PLCP Header (Long or Short) and PSDU.

The length of PLCP Preamble (mPLCP) is determined by PLCPType parameter:

$$mPLCP = \begin{cases} 144 \times 11 \times SampsPerChip & ifPLCPType = Long \\ 72 \times 11 \times SampsPerChip & ifPLCPType = Short \end{cases}$$

The length of Header (mHeader) is determined by PLCPType parameter:

$$mHeader = \begin{cases} 48 \times 11 \times SampsPerChip & ifPLCPType = Long \\ 24 \times 11 \times SampsPerChip & ifPLCPType = Short \end{cases}$$

The length of PSDU (mPSDU) is determined by Rate parameter:

$$mPSDU = \begin{cases} Octets \times 8 \times 11 \times SampsPerChip & ifRate = 1Mbps \\ Octets \times 4 \times 11 \times SampsPerChip & ifRate = 2Mbps \\ Octets \times 2 \times 8 \times SampsPerChip & ifRate = 5.5Mbps \\ Octets \times 1 \times 8 \times SampsPerChip & ifRate = 11Mbps \end{cases}$$

The length of Ramp (mRamp) can be calculated as follows:

$$mRamp = \begin{cases} 0 & ifPwrType = None \\ 2 \times RampTime \times 11 \times 10^6 \times SampsPerChip & ifPwrType = Linear, Cosine \end{cases}$$

The length of Idle (mIdle) can be calculated as follows:

$$mIdle = IdleInterval \times 11 \times 10^6 \times SampsPerChip$$

So, the total of one 802.11b burst (at BurstIn) is:

$$Total = mIdle + mRamp + mPLCP + mHeader + mPSDU$$

The actual burst includes PLCP Preamble, PLCP Header, and a 3-part PSDU. So, WLAN_11bDemuxBurst outputs the actual number of 802.11b bursts (mActual) which is calculated as follows:

$$mActual = mPLCP + mHeader + mPSDU$$

3. The transmitter transmits burst-by burst in ADS. The burst sequence is a continuous stream. (The 802.11 burst is transmitted burst-by-burst.) This model includes a frequency compensator. The transmitted consecutive bursts are independent.
   The estimated frequency offset ($\Delta f_i$) of each received burst is input at DeltaF. To avoid estimated frequency offset impacting the next bursts in the frequency compensator, the FreqOffset parameter in this model is set as the actual frequency offset between the transmitter and the receiver. When the ith burst is processed, the actual phase of previous i-1 bursts is calculated and removed by FreqOffset. The ith estimated frequency offset ($\Delta f_i$) is used to compensate for the phase in current burst only.
   Assume $x_0$, $x_1$, ..., $x_{Total}$ -1 sequences are the BurstIn burst signals after removing the actual phase of previous *i* -1 bursts caused by the frequency offset from

FreqOffset. The $y_0, y_1, \ldots, y_{Total-1}$ sequences (whose phase caused by frequency offset) are removed:

$$y_k = x_k \times e^{-j2\pi\Delta f_i k T_{samples}}$$

where

$\Delta f_i$ is frequency offset of ith received burst which is the input in *DeltaF*,

$T_{sample} = \frac{1}{11MHz \times SampsPerChip}$ is the sample time interval in 802.11b system.

After frequency offset compensation, the actual 802.11b burst is output. Index pin 2 inputs the start of a detected 802.11b burst (including ramps and idle) from WLAN_11bBurstSync. The ramps and idle must be discarded when the actual burst is output; the equation is

$$z_k = y_{k + Index + \frac{mRamp}{2} + \frac{mIdle}{2}}, k = 0, \ldots, mActual - 1$$

$z_0, z_1, \ldots, z_{mActual-1}$ sequences are output.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_11bDescrambler

**Description 11b descrambler**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bDescrambler**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| InitState | initial state of scrambler | 1 1 0 1 1 0 0 | int array | 0 or 1<br>array size is 7 |
| Octets | number of octets in PSDU | 100 | int | (0, 2336] |

## Pin Inputs

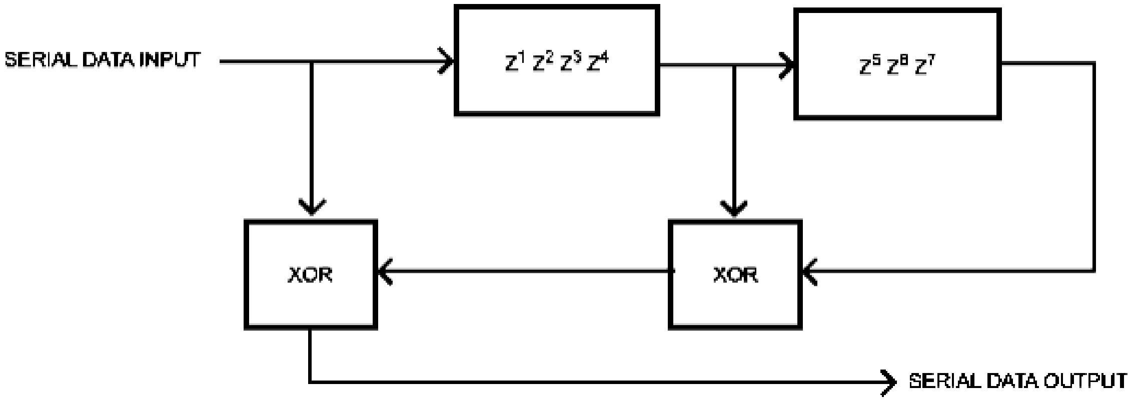
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | bits to be descrambled | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | descrambled bits | int |

## Notes/Equations

1. This component descrambles all input data bits. Each firing, when Octets bytes are consumed and produced, the descrambler is reset to its initial state.
2. The generator polynomial of the scrambler is $G(z) = z^{-7} + z^{-4} + 1$; the corresponding descrambler is illustrated in the following figure.



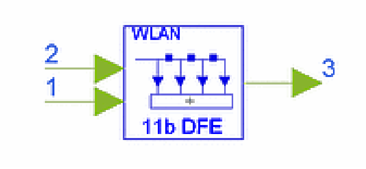
**Data Desrambler**

The feed-through configuration of the scrambler and descrambler is self-synchronizing, which requires no prior knowledge of the transmitter initialization of the scrambler for receiver processing.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_11bDFE



**Description decision feedback equalizer for 11b**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bDFE**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | enum | |
| ModType | modulation type: CCK, PBCC | CCK | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |
| Algorithm | algorithm for equalizer: RLS, LMS | RLS | enum | |
| FwdTaps | number of forward taps in equalizer | 5 | int | [1, 256] |
| FbkTaps | number of feedback taps in equalizer | 3 | int | [1, 128] |
| Alpha | scale factor for LMS algorithm | 0.0004 | real | (0.0, 1.0) |
| Lambda | weighting factor for RLS algorithm | 0.999 | real | (0.0, 1.0) |
| Delta | small positive constant for RLS algorithm | 0.001 | real | (0.0, 10.0] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | input signal before equalizer | complex |
| 2 | PrmblIn | input training preamble for equalizer | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | BurstOut | output data after equalizer | complex |

## Notes/Equations

1. This model implements decision feedback equalization for input 802.11b burst.
2. Rate, ModType, PLCPType and Octets parameters are used to specify the number of complex signals in one burst. In the WLAN Design Library, one real 802.11b burst consists of PLCP Preamble (Long or Short), PLCP Header (Long or Short) and PSDU. The length of PLCP Preamble (mPLCP) is determined by the PLCPType parameter:

$$mPLCP = \begin{cases} 144 \times 11 & if PLCPType = Long \\ 72 \times 11 & if PLCPType = Short \end{cases}$$

The length of Header (mHeader) is determined by the PLCPType parameter:

$$mHeader = \begin{cases} 48 \times 11 & ifPLCPType = Long \\ 24 \times 11 & ifPLCPType = Short \end{cases}$$

The length of PSDU (mPSDU) is determined by Rate parameter:

$$mPSDU = \begin{cases} Octets \times 8 \times 11 & ifRate = 1Mbps \\ Octets \times 4 \times 11 & ifRate = 2Mbps \\ Octets \times 2 \times 8 & ifRate = 5.5Mbps \\ Octets \times 1 \times 8 & ifRate = 11Mbps \end{cases}$$

So, the total of one 802.11b burst is:

$$Total = mPLCP + mHeader + mPSDU$$

Each firing, Total tokens are input at BurstIn and Total tokens are output at BurstOut.

3. Pin 1 (BurstIn) input

   $$x_0, x_1, \ldots, x_{Total-1}$$

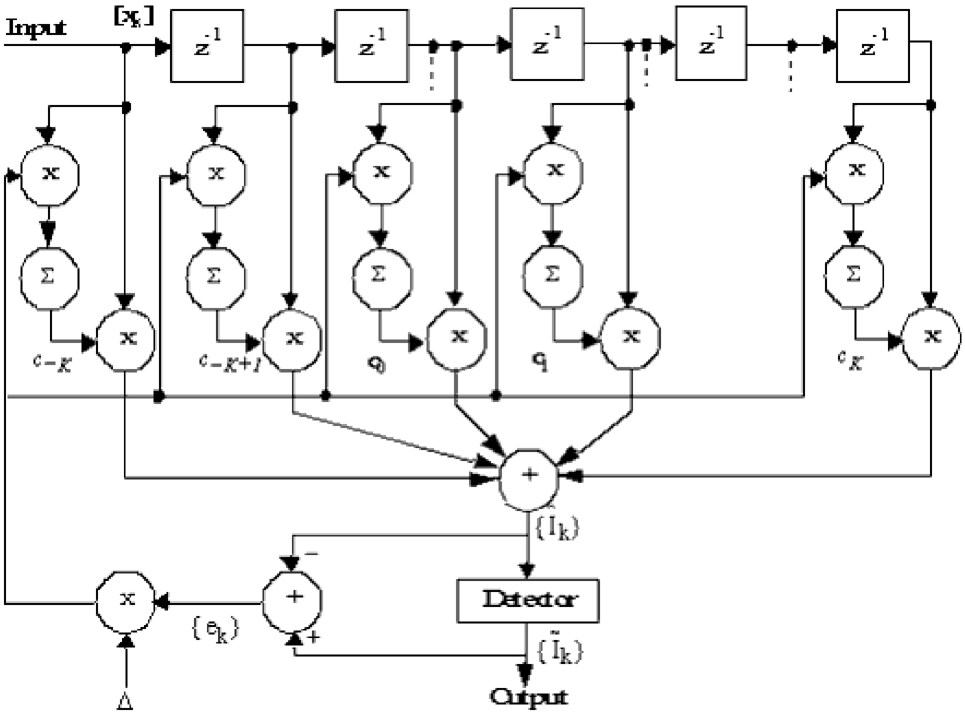
   sequences to be equalized, Pin 2 (PrmblIn) input mPLCP training sequences

   $$p_0, p_1, \ldots, p_{mPLCP-1}$$,

   which is used to train the decision feedback equalizer. Pin 3 (BurstOut) output equalized sequence

   $$y_0, y_1, \ldots, y_{Total-1}$$.

   Pin 2 (PrmblIn) must be connected with pin 3 (PrmblChip) of WLAN_11bPreamble.
4. This model implements a decision feedback equalizer. The equalizer works in training and tracking modes.
   In the training mode, the training sequence (from pin2 PrmblIn) is used as training sequence. The number of training sequences is mPLCP. The error signal $e_k$ (see note 5) is from the training signal. After the training mode, the decision feedback equalizer coefficient is converged and the equalizer enters into the tracking mode. The error signal $e_k$ is from the decision signal of the equalized signal.
5. This model supports the LMS and RLS algorithms to equalize the input signal. Users can select one algorithm to equalize the received signal according to various test cases. The following figure illustrates the decision feedback equalizer.



Decision Feedback Equalizer

## LMS Algorithm

The criterion most commonly used in the optimization of the equalizer coefficients is the minimization of the mean square error (MSE) between the desired equalizer output and the actual equalizer output.

MSE minimization can be accomplished recursively by use of the stochastic gradient algorithm introduced by Widrow, called the LMS algorithm. This algorithm is described by the coefficient update equation

$$C_{k+1} = C_k - \alpha e_k X_k{}^*$$

where

$C_k$ is the vector of the equalizer coefficients at the kth iteration

$X_k$ represents the signal vector.

## Recursive Least-Squares (Kalman) Algorithm

The convergence rate of the LMS algorithm is slow because a single parameter α controls the rate of adaptation. A fast converging algorithm is obtained if a recursive least squares (RLS) criterion for adjustment of the equalizer coefficients.

The RLS iteration algorithm follows.

Calculate output:

$$\hat{I}_k = C_{k-1} \times X_k$$

Calculate error:

$$e_k = I_k - \hat{I}_k$$

Calculate Kalman gain vector:

$$K_k = \frac{P_{k-1} \times X_k^*}{\lambda + X_k^T \times P_{k-1} \times X_k^*}$$

Update inverse of the correlation matrix:

$$P_k = \frac{1}{\lambda} \times \left[ P_{k-1} - K_k \times X_k^T \times P_{k-1} \right]$$

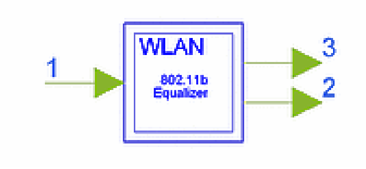
Update coefficients:

$$C_k = C_{k-1} + P_k \times X_k^* \times e_k$$

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.
4. John G. Proakis, Adaptive Equalization for TDMA Digital Mobile Radio, IEEE Trans. on Vehicular Technology, page 333-341, Vol. 40, No.2, May 1991.

# WLAN_11b_Equalizer



**Description 11b receiver with equalizer**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11b_Equalizer**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| EquAlgorithm | algorithm for equalizer: RLS, LMS | RLS | | enum | |
| FwdTaps | number of forward taps in equalizer | 5 | | int | [1, 256] |
| FbkTaps | number of feedback taps in equalizer | 3 | | int | [1, 128] |
| EquAlpha | scale factor for LMS algorithm | 0.0004 | | real | (0.0, 1.0) |
| Lambda | weighting factor for RLS algorithm | 0.999 | | real | (0.0, 1.0) |
| Delta | small positive constant for RLS algorithm | 0.001 | | real | (0.0, 10.0] |
| DownSamplePhase | phase of down sample model with range from 0 to SampsPerChip-1 | 0 | | int | [0.0, SampsPerChip-1] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

### Pin Inputs

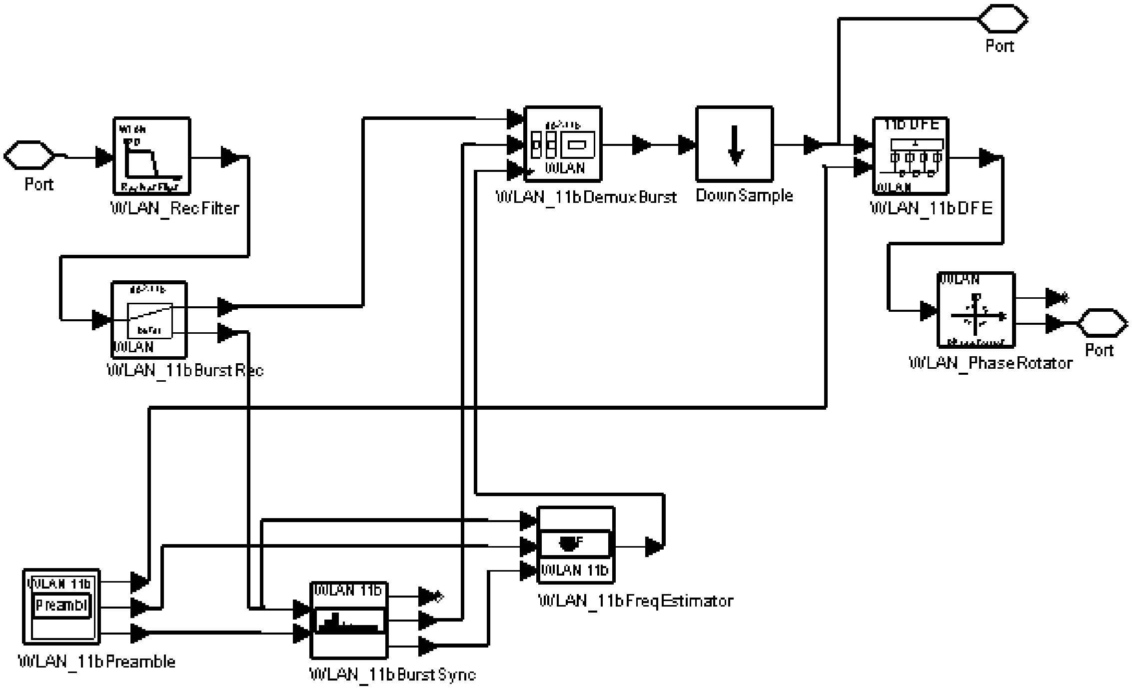
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | received burst | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | AfterEqu | 11b burst after equalizer | complex |
| 3 | BeforeEqu | 11b burst before equalizer | complex |

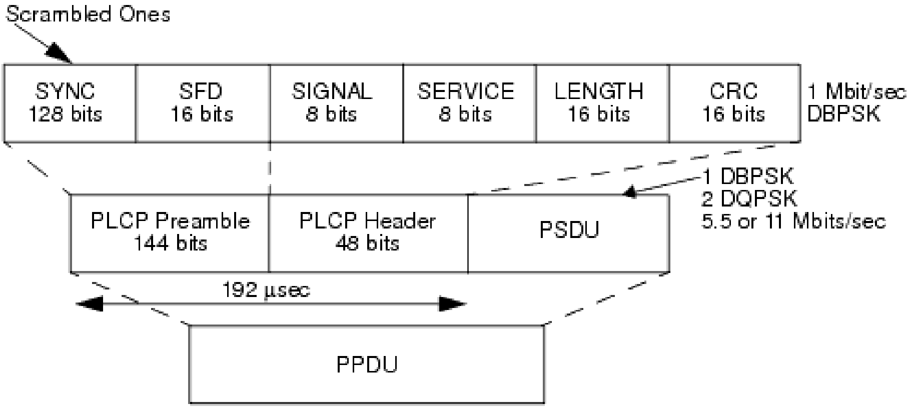
## Notes/Equations

1. This subnetwork implements an 802.11b receiver with decision feedback equalizer. It performs burst synchronization, carrier frequency synchronization and decision feedback equalization. The schematic for this subnetwork is shown in the following figure.

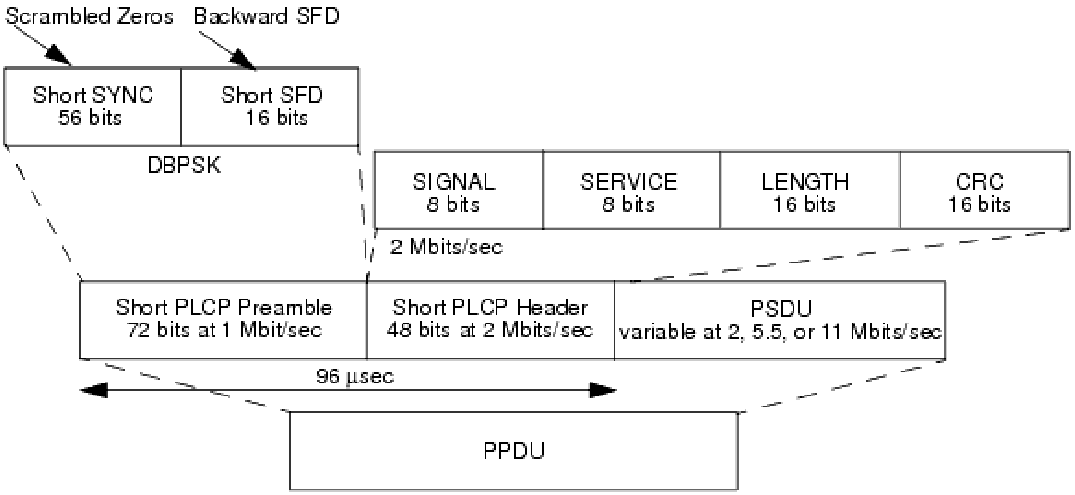


**WLAN_11b_Equalizer Schematic**

2. This subnetwork outputs both the full PPDU frame before and after the equalizer. So, the constellation can be shown before or after the decision feedback equalizer. The PPDU frames are illustrated in the following figures; one PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

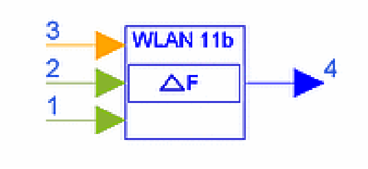


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.
4. John G. Proakis, Adaptive Equalization for TDMA Digital Mobile Radio, IEEE Trans. on Vehicular Technology, page 333-341, Vol. 40, No.2, May 1991.

# WLAN_11bFreqEstimator



**Description 11b frequency offset estimator**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bFreqEstimator**
**Derived From WLAN_11bBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| FingerNumber | number of rake receiver fingers | 3 |  | int | [1, 30] |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 |  | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SampsPerChip | number of samples per chip | 2 |  | int | [2, 8] |
| PLCPType | PLCP preamble type: Long, Short | Long |  | enum |  |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SyncIn | input signal for frequency synchronization | complex |
| 2 | Ref | reference signal for frequency estimation | complex |
| 3 | index | synchronization index | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | DeltaF | coarse carrier frequency offset | real |

## Notes/Equations

1. This model is used to estimate the carrier frequency offset between the transmitter and the receiver for 802.11b.
2. Before frequency offset estimation, channel estimation is completed and the received signals are rake combined in this model to ensure the excellent SNR. The finger number for rake combination is set by FingerNumber.
3. The maximum likelihood principle is applied to the frequency offset estimation. The received signals are differentiated and denoted as $Y[k]$ and the reference signal is denoted as d[k]; the estimate of frequency offset is as follows:

3. $$\Delta f = \frac{1}{2\pi T} arc\tan\left(\frac{\sum_{k=0}^{N-1} Im\{Y[k] \times d^*[k]\}}{\sum_{k=0}^{N-1} Re\{Y[k] \times d^*[k]\}}\right)$$

   where

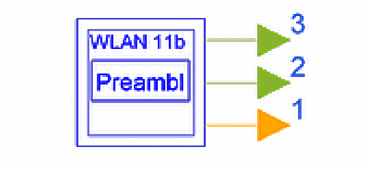
   *T* denotes the time duration of one symbol
   *N* denotes the number of symbols used for frequency offset estimation and set by the FreqEstWindow parameter.

   The result is output at DeltaF.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bPreamble



**Description Signal source of 11b preamble**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bPreamble**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum | [0, 1] |
| InitPhase | initial phase of DBPSK | PI / 4 | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | int array | {0, 1}† |

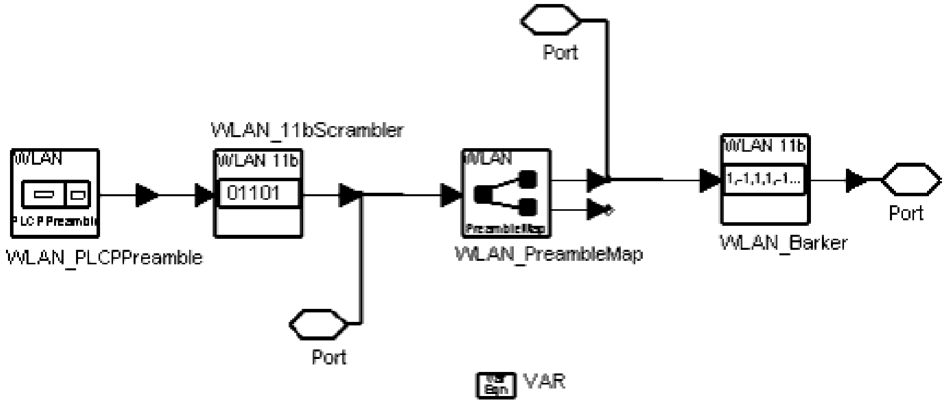
† for each array element: array size must be 7.

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PrmblBits | 802.11b preamble bits after scrambler | int |
| 2 | PrmblSym | 802.11b preamble after DBPSK modulation | complex |
| 3 | PrmblChip | 802.11b preamble after Barker spreader | complex |

## Notes/Equations

1. This subnetwork outputs the PLCP preamble bits, mapping signals and chip signals. The schematic for this subnetwork is shown in the following figure.



**WLAN_11bPreamble Schematic**

2. Both long and short PLCP preambles use 1 Mbit/s DBPSK modulation.

The InitPhase parameter specifies the initial phase of the DBPSK signal. DBPSK encoding is given in the following table.

**DBPSK Encoding**

| Bit Input | Phase Change (+jw) |
|---|---|
| 0 | 0 |
| 1 | п |

3. The 11-chip Barker sequence is used as the PN code sequence for the 1 and 2 Mbit/s modulation:

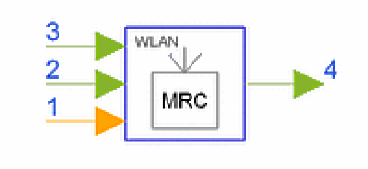
   +1, -1, +1, +1, -1, +1, +1, +1, -1, -1, -1

   The left-most chip is output first in time. The first chip is aligned at the start of a transmitted symbol. The symbol duration will be exactly 11 chips long.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bRake



**Description 11b rake combiner**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11bRake**
**Derived From WLAN_11bBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| IdleInterval | idle time | 50.0μsec | sec | real | [10μsec, 1000μsec] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | Index | samples index of channel delay | int |
| 2 | CIR | channel CIR | complex |
| 3 | BurstIn | 11b received signal | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | BurstOut | 11b multipath combined signal | complex |

## Notes/Equations

1. This model implements coherent receiving with maximal ratio combining (MRC). The necessary coefficients for rake combining, including multipath delay and channel impulse response, are received from the channel estimator component.
   Each firing, the position of each finger is input from pin Index, the CIR are received by pin CIR, at the same time, the burst whose frequency deviation has been compensated are received by pin BurstIn. Then, MRC are implemented.
   Each firing, the pin BurstIn, CIR and Index consume tokens separately as follows:
   (RampTime × 2+IdleInterval) × 11E6 × SampsPerChip+DataLength

DataLength can be calculated as:
DataLength = (NPLCP+NSYM) × SampsPerChip;
if PLCPType=Long (NPLCP=(144+48) × 11) else (NPLCP=(72+24) × 11);

if Rate=1 Mbps (NSYM=Octets × 8 × 11)
if Rate=2 Mbps (NSYM=Octets × 4 × 11)
if Rate=5.5 Mbps(NSYM=Octets × 8 × 2)
if Rate=11 Mbps ((NSYM=Octets × 8)

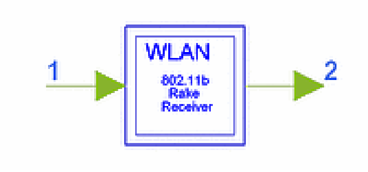
At the same time, DataLength tokens are produced from pin BurstOut.

2. FingerNumber specifies the number of combined paths.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11b_Rake



**Description 11b rake receiver**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_11b_Rake**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0μsec | sec | real | [10μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| CIREstWindow | number of preamble symbols used to estimate channel | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

### Pin Inputs

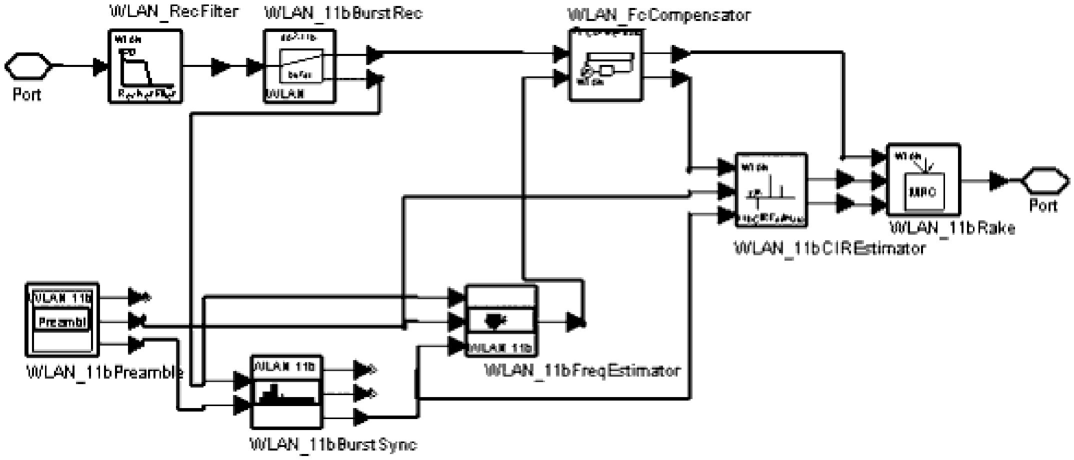
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | received burst | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | AfterRake | 11b burst after rake receiver | complex |

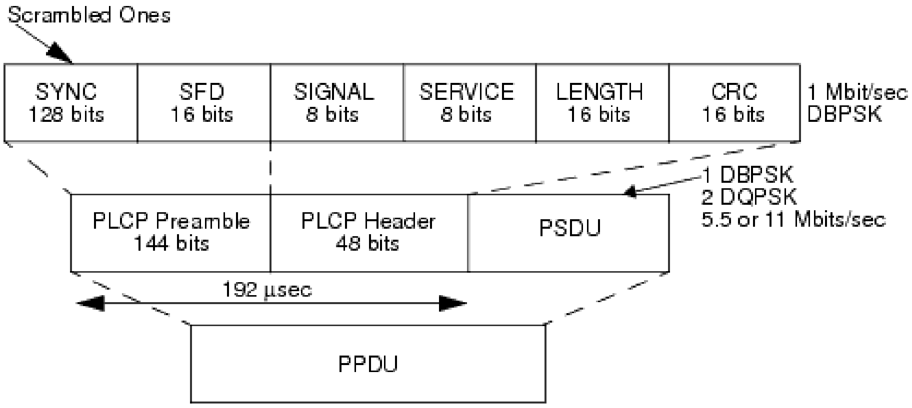
### Notes/Equations

1. This subnetwork implements IEEE 802.11b rake receiver. It performs burst synchronization, carrier frequency synchronization, channel estimation and rake combination. The schematic for this subnetwork is shown in the following figure.

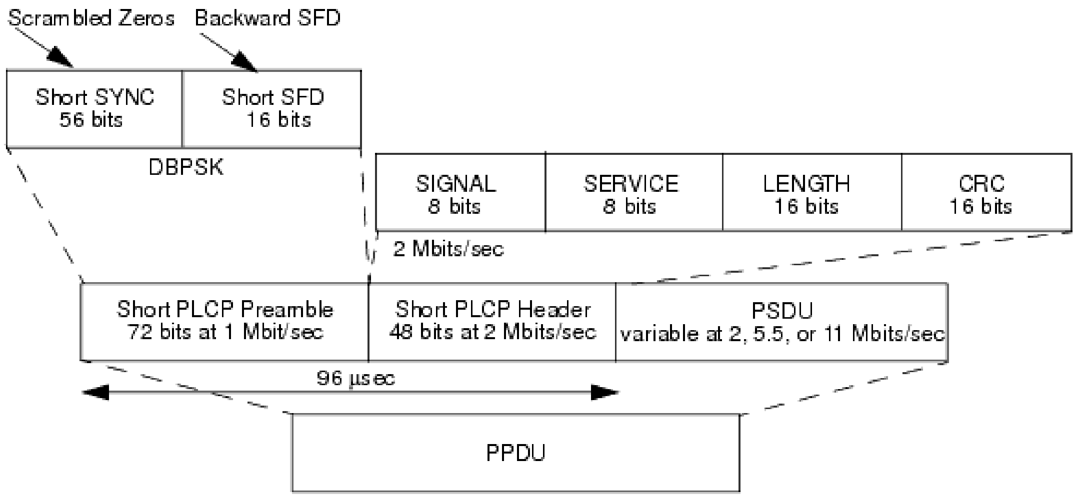


**WLAN_11b_Rake Schematic**

2. This subnetwork outputs the full PPDU frame after rake combination. The PPDU frames are illustrated in the following figures; one PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

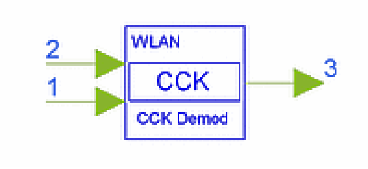


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.

# WLAN_CCKDemod



**Description 11b CCK demodulator**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_CCKDemod**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | CCKIn | input signal for CCK demodulator | complex |
| 2 | InitialPhase | initial phase | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | PSDU | signal demodulated to PSDU | complex |

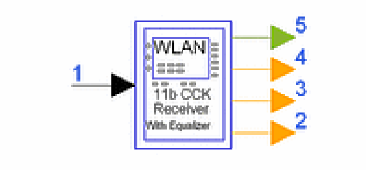
## Notes/Equations

1. This model implements 11b CCK demodulator. Each firing, the NSYM tokens are consumed by pin CCKIn and Octets × 8 tokens are produced by pin PSDU.
   NSYM can be calculated as follows:
   - if Rate=5.5 Mbps(NSYM=Octets × 8 × 2)
   - if Rate=11 Mbps((NSYM=Octets × 8)
2. The CCK demodulation is implemented with a correlator to correlate the 4 or 64 CCK complex code word. The correlator is followed by a biggest picker which finds the biggest of 4 or 64 correlator outputs depending on the rate, where 4 if rate is 5.5M and 64 if rate is 11M. This is translated into two or six data bits. The detected output is then processed through the differential phase decoder to demodulate the last two bits of the symbol.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_CCK_RF_Rx_DFE



**Description**: 802.11b CCK receiver with equalizer
**Library**: WLAN, 11b Receiver
**Class**: TSDFWLAN_CCK_RF_Rx_DFE

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp | Ohm | real | [-273.15, ∞] |
| RefFreq | internal reference frequency | 2400MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 | | real | (-∞, ∞) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312) |
| InitPhase | initial phase of DBPSK | 0.785 | | real | (0, 2п ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0usec | sec | real | [0usec, 1000usec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0usec | sec | real | [0usec, 1000usec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |

| | | | | | |
|---|---|---|---|---|---|
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | [0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0) |
| MaxSearchDelay | time range for searching fading path | 3.0usec | sec | real | [3usec, 10usec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| EquAlgorithm | algorithm for equalizer: RLS, LMS | RLS | | enum | |
| FwdTaps | number of forward taps in equalizer | 5 | | int | [1, 256] |
| FbkTaps | number of feedback taps in equalizer | 3 | | int | [1, 128] |
| EquAlpha | scale factor for LMS algorithm | 0.0004 | | real | (0.0, 1.0) |
| Lambda | weighting factor for RLS algorithm | 0.999 | | real | (0.0, 1.0) |
| Delta | small positive constant for RLS algorithm | 0.001 | | real | (0.0, 10.0) |
| DownSamplePhase | phase of down sample model with range from 0 to SampsPerChip-1 | 0 | | int | [0.0, SampsPerChip-1] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |
| † for each array element: array size must be 7. | | | | | |

### Pin Input

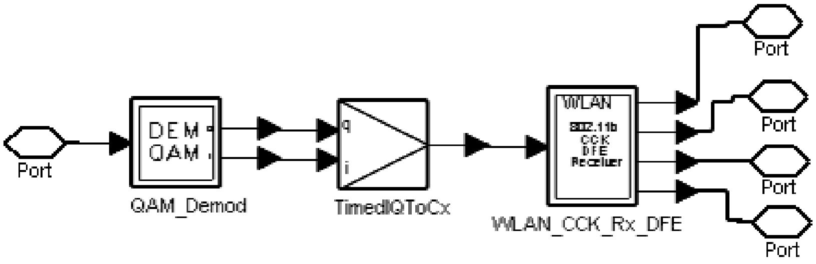
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |

### Pin Output

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | PSDU | demodulated PSDU after CCK receiver | int |
| 3 | Header | demodulated header after CCK receiver | int |
| 4 | Preamble | demodulated preamble after CCK receiver | int |
| 5 | BurstOut | 11b CCK burst after DFE receiver | complex |

### Notes/Equations

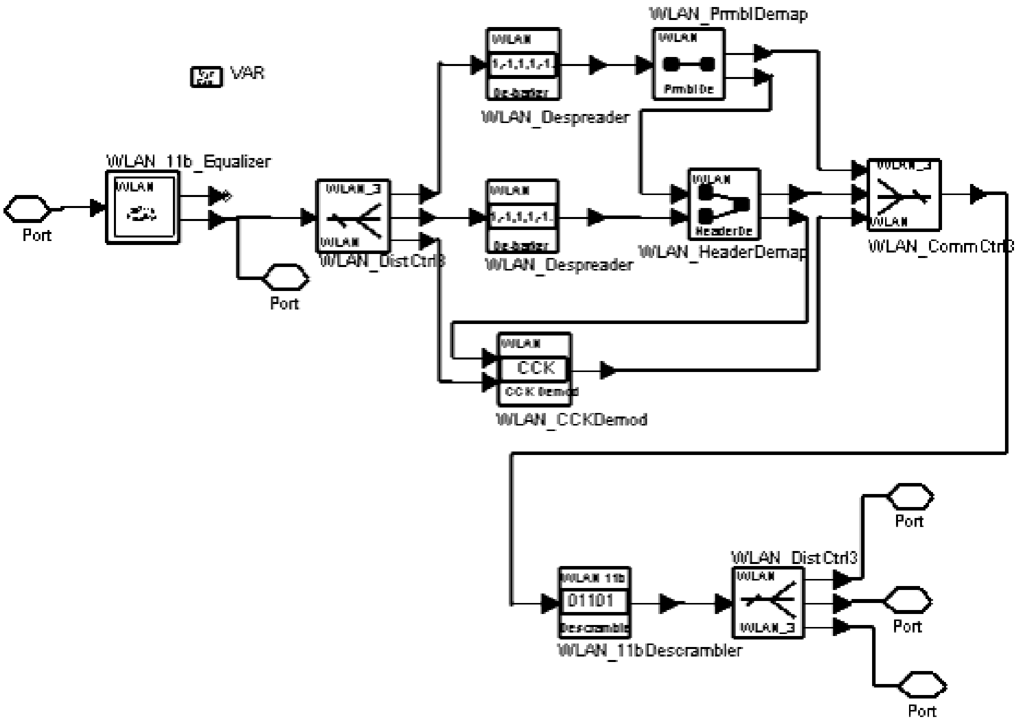
1. This subnetwork provides the full CCK receiver according to the IEEE 802.11b Standard; it can be configured in a top-level design using model parameters.
2. This subnetwork integrates an RF demodulator and baseband receiver. The schematic is shown in the following figure.



**WLAN_CCK_RF_Rx_DFE Schematic**

Receiver functions are implemented as specified in the IEEE 802.11b Standard.

- Start of frame is detected (WLAN_11bBurstSync).
- Carrier frequency offsets are estimated (WLAN_FreqEstimator).
- The packet is derotated according to estimated frequency offset (WLAN_11bDemuxBurst in the WLAN_11b_Equalizer subnetwork).
- The received packet is equalized by the decision feedback equalizer (WLAN_11bDFE in the WLAN_11b_Equalizer subnetwork).
- The equalized signal is demultiplexed into PSDU, Header and Preamble parts.
- Header and Preamble parts are Barker despread.
- PSDU parts are CCK demodulated.
- Despread Preamble and Header and CCK-demodulated PSDU are multiplexed into one bit stream; the bit stream is then descrambled.
- The descrambled bit stream is demultiplexed into PSDU, Preamble, and Header bits and output.
- The equalized burst (BurstOut) is output.  
  The WLAN_CCK_Rx_DFE receiver schematic is shown in the following figure.

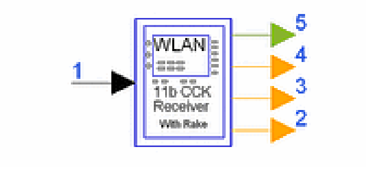


**WLAN_CCK_Rx_DFE Schematic**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.
4. John G. Proakis, Adaptive Equalization for TDMA Digital Mobile Radio, IEEE Trans. on Vehicular Technology, page 333-341, Vol. 40, No.2, May 1991.

# WLAN_CCK_RF_Rx_Rake



**Description:** 802.11b CCK Rake receiver
**Library:** WLAN, 11b Receiver
**Class:** TSDFWLAN_CCK_RF_Rx_Rake

## Parameters

| **Name** | **Description** | **Default** | **Unit** | **Type** | **Range** |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞ ) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞ ) |
| RTemp | physical temperature, in degrees C | DefaultRTemp | Ohm | real | [-273.15, ∞ ) |
| RefFreq | internal reference frequency | 2400MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 | | real | (-∞, ∞ ) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1} † |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0usec | sec | real | [0usec, 1000usec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0usec | sec | real | [10usec, 1000usec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| MaxSearchDelay | time range for searching fading path | 3.0usec | sec | real | [3usec, 10usec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| CIREstWindow | number of preamble symbols used to estimate channel | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

### Pin Inputs

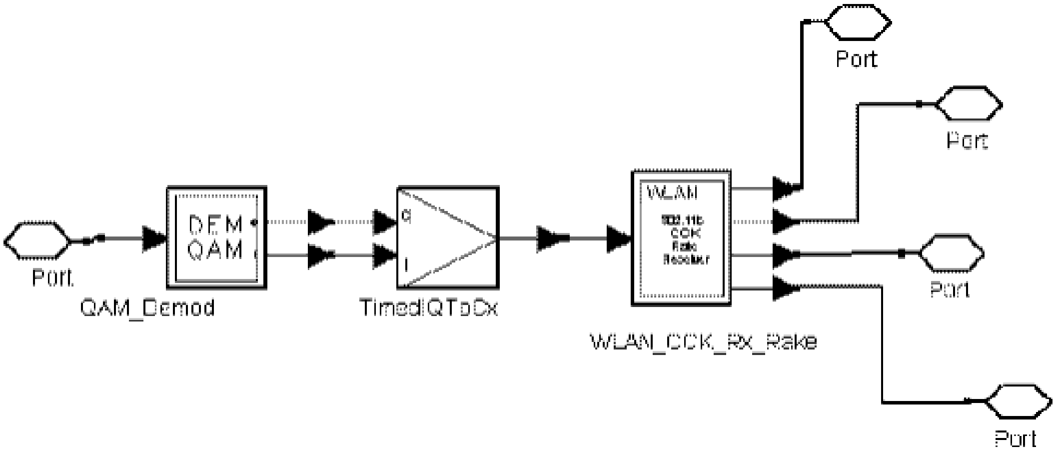
| **Pin** | **Name** | **Description** | **Signal Type** |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | PSDU | demodulated PSDU after CCK receiver | int |
| 3 | Header | demodulated header after CCK receiver | int |
| 4 | Preamble | demodulated preamble after CCK receiver | int |
| 5 | BurstOut | 11b CCK burst after Rake receiver | complex |

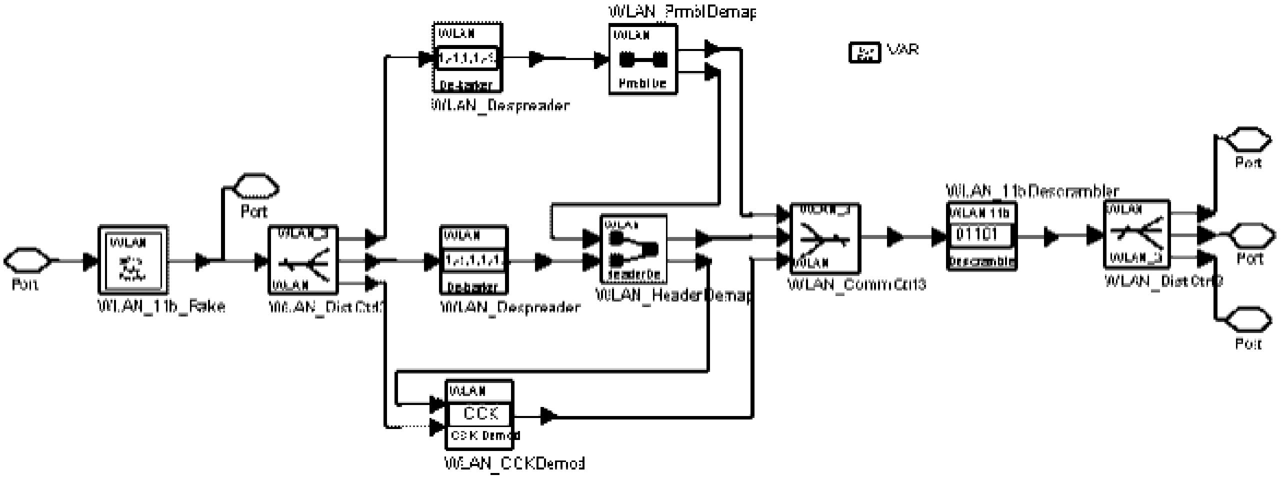
### Notes/Equations

1. This subnetwork provides the full CCK receiver according to the IEEE 802.11b Standard; it incorporates rake combination and can be configured in a top-level design using model parameters.
   This subnetwork integrates an RF demodulator and baseband receiver; the schematic is shown in the following figure.



**WLAN_CCK_RF_Rx_Rake Schematic**

2. Receiver functions are implemented as specified in the IEEE 802.11b Standard.
   - Start of frame is detected (WLAN_11bBurstSync).
   - Carrier frequency offsets are estimated (WLAN_FreqEstimator).
   - The packet is derotated according to estimated frequency offset (WLAN_FcCompensator).
   - The received packets are rake combined (WLAN_11bRake).
   - The rake combined signal is demultiplexed into PSDU, Header and Preamble parts.
   - Preamble and Header parts are Barker despread.
   - PSDU parts are CCK demodulated.
   - Preamble, Header, and CCK are then multiplexed into one bit stream; the bit stream is then descrambled.
   - The descrambled bit stream is demultiplexed into PSDU, Preamble, and Header bits; these parts are output.
   - The rake-combined burst is output.
     The WLAN_CCK_Rx_Rake receiver schematic is shown in the following figure.

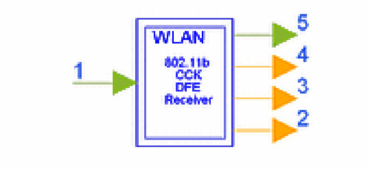


**WLAN_CCK_Rx_Rake Schematic**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.

# WLAN_CCK_Rx_DFE



**Description 802.11b CCK receiver with equalizer**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_CCK_Rx_DFE**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2п ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| EquAlgorithm | algorithm for equalizer: RLS, LMS | RLS | | enum | |
| FwdTaps | number of forward taps in equalizer | 5 | | int | [1, 256] |
| FbkTaps | number of feedback taps in equalizer | 3 | | int | [1, 128] |
| EquAlpha | scale factor for LMS algorithm | 0.0004 | | real | (0.0, 1.0) |
| Lambda | weighting factor for RLS algorithm | 0.999 | | real | (0.0, 1.0) |
| Delta | small positive constant for RLS algorithm | 0.001 | | real | (0.0, 10.0] |
| DownSamplePhase | phase of down sample model (0..SampsPerChip-1) | 0 | | int | [0.0, SampsPerChip-1] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

### Pin Inputs

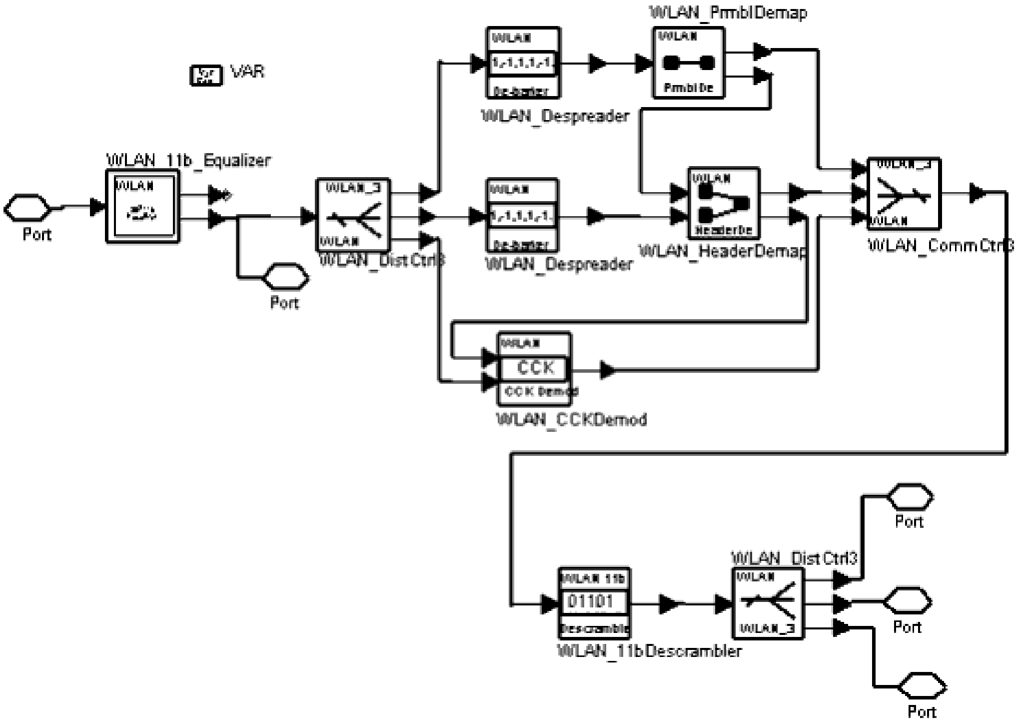
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | received burst | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | PSDU | demodulated PSDU after CCK receiver | int |
| 3 | Header | demodulated header after CCK receiver | int |
| 4 | Preamble | demodulated preamble after CCK receiver | int |
| 5 | BurstOut | 11b CCK burst after DFE receiver | complex |

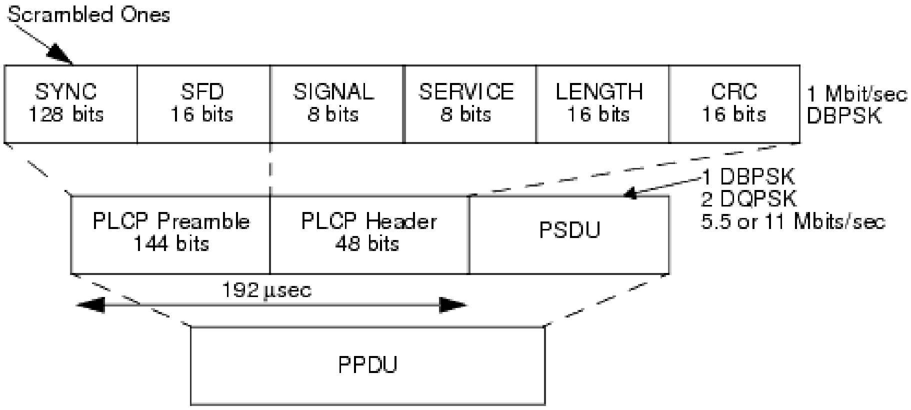
### Notes/Equations

1. This subnetwork implements an 802.11b CCK receiver with decision feedback equalizer. It performs burst synchronization, carrier frequency synchronization, decision feedback equalization, Barker despreading, CCK demodulation and descrambling. The schematic is shown in the following figure.

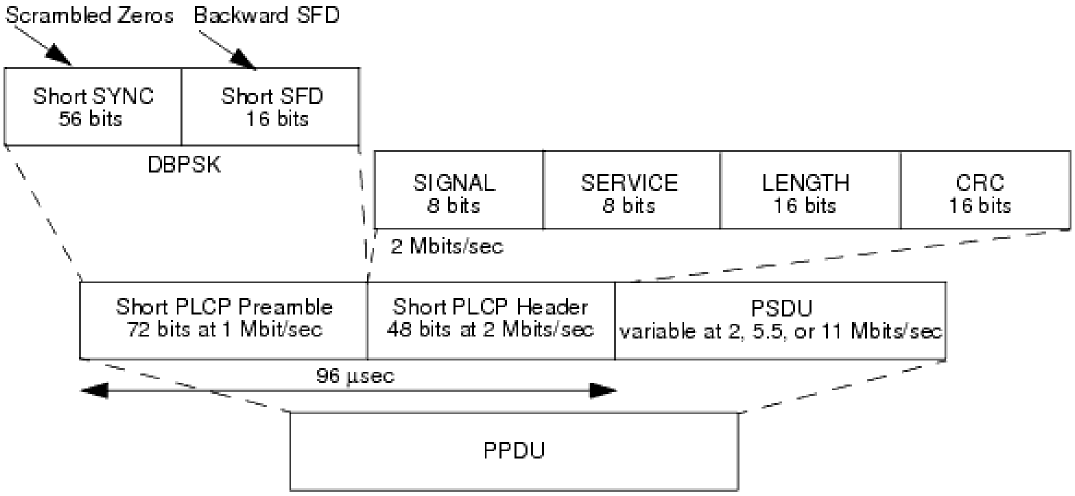


WLAN_CCK_Rx_DFE Schematic

2. This subnetwork outputs both the demodulated PSDU, Header, Preamble and the full PPDU frame after equalizer. The PPDU frames are illustrated in the following figures. One PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

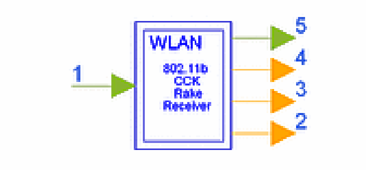


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.
4. John G. Proakis, Adaptive Equalization for TDMA Digital Mobile Radio, IEEE Trans. on Vehicular Technology, page 333-341, Vol. 40, No.2, May 1991.

# WLAN_CCK_Rx_Rake



**Description 802.11b CCK Rake receiver**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_CCK_Rx_Rake**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| SampsPerChip | number of samples per chip | 2 | | int | [2, 8] |
| IdleInterval | idle time | 50.0μsec | sec | real | [10μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| FingerNumber | number of rake receiver fingers | 3 | | int | [1, 30] |
| CIREstWindow | number of preamble symbols used to estimate channel | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqEstWindow | number of preamble symbols used to estimate frequency offset | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| SyncEstWindow | number of preamble symbols used for burst synchronization | 20 | | int | [1, 128] if PLCPType=Long<br>[1, 56] if PLCPType=Short |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

† for each array element: array size must be 7.

### Pin Inputs

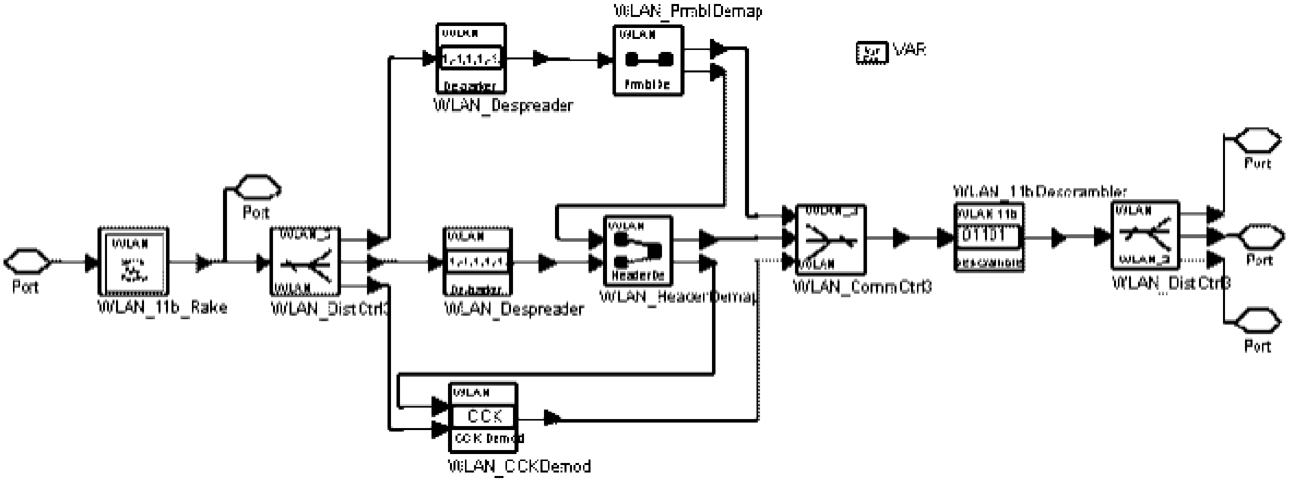
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | received burst | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | PSDU | demodulated PSDU after CCK receiver | int |
| 3 | Header | demodulated header after CCK receiver | int |
| 4 | Preamble | demodulated preamble after CCK receiver | int |
| 5 | BurstOut | 11b CCK burst after Rake receiver | complex |

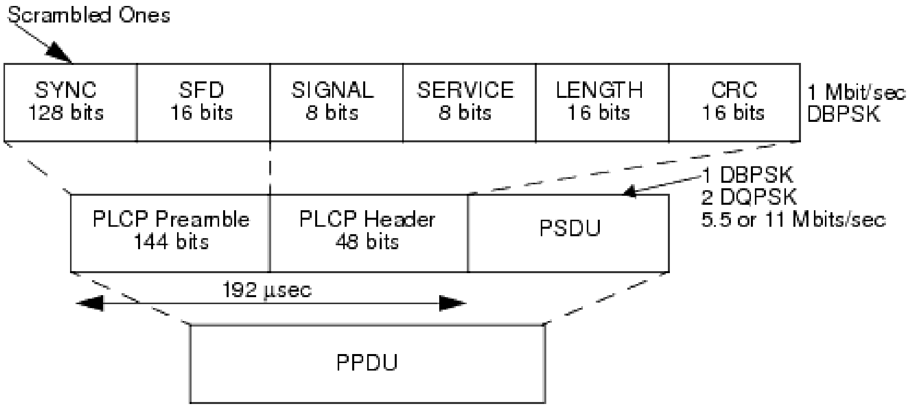
### Notes/Equations

1. This subnetwork implements an 802.11b CCK receiver with rake combination. It performs burst synchronization, carrier frequency synchronization, rake combination, Barker despreading, CCK demodulation and descrambling. The schematic is shown in the following figure.

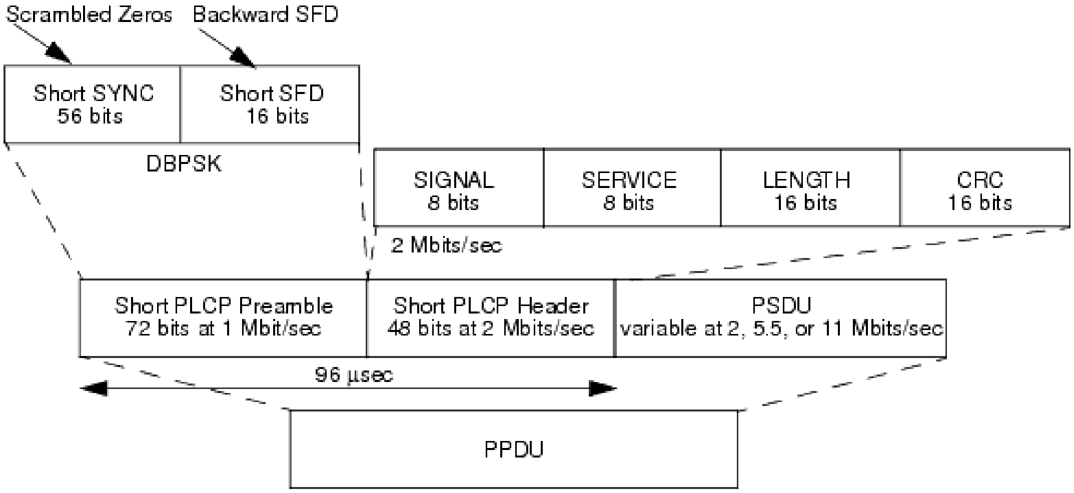


**WLAN_CCK_Rx_Rake Schematic**

2. This subnetwork outputs both the demodulated PSDU, Header, Preamble and the full PPDU frame after rake combination. The PPDU frames are illustrated in the following figures. One PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**



**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999
3. John G. Proakis, Digital Communications, Third Edition, page 636-679, McGraw-Hill, 1995.

# WLAN_Despreader

**Description 11b barker despreader**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_Despreader**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input Barker spreading signal | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signal after Braker de-spreading | complex |

## Notes/Equations

1. This model is used to implement Barker despreader for 802.11b. Each firing, 11 input tokens make the Barker despreader and one despread signal is output.
2. The 11-chip Barker sequence

$$\{B_0 B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}\}$$

is used as the PN code sequence for 1 and 2 Mbit/s modulation:

$\{B_0 B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}\}$ = +1, -1, +1, +1, -1, +1, +1, +1, -1, -1, -1

The Barker spreading signal sequence is

$$\{x_0 x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}.$$

The despreader calculates the despreading signal as follows, then outputs *y* .

$$y = \sum_{i=0}^{10} B_i \times x_i$$

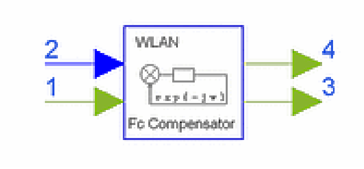
## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_FcCompensator



**Description 11b carrier frequency compensator**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_FcCompensator**
**Derived From WLAN_11bBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| MaxSearchDelay | time range for searching fading path | 3.0μsec | sec | real | [3μsec, 10μsec] |
| SampsPerChip | number of samples per chip | 2 |  | int | [2, 8] |
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 |  | enum |  |
| PLCPType | PLCP preamble type: Long, Short | Long |  | enum |  |
| Octets | octet number of PSDU | 100 |  | int | (0, 2312] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None |  | enum |  |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| ModType | modulation type: CCK, PBCC | CCK |  | enum |  |
| IdleInterval | idle time | 50.0μsec | sec | real | [10μsec, 1000μsec] |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | the 11b received signal | complex |
| 2 | DeltaF | estimated frequency offset | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | BurstOut | 11b signal compensated with freq offset | complex |
| 4 | PLCP | compensated PLCP is to be used as input of channel estimation model | complex |

## Notes/Equations

1. This model compensates for carrier frequency offset.
   Each firing, the frequency offset detected by WLAN_11bFreqEstimator is input at pin DeltaF and the frequency offset burst are input at pin BurstIn. The burst is then compensated by multiplying exp(-j × 2 × PI × DeltaF × n × Ts) and output at pin BurstOut, where n is the number of burst samples from 1 to entire burst length and Ts is sample time interval (described in note 3). Pin PLCP is the compensated PCLP signal.

   Each firing, the tokens consumed by BurstIn can be calculated as follows.

   (RampTime × 2+IdleInterval) × 11E6 × SampsPerChip+DataLength,

   where

   DataLength = (NPLCP+NSYM) × SampsPerChip

   if PLCPType=Long (NPLCP=(144+48) × 11)
   else (NPLCP=(72+24) × 11);
   if Rate=1M (NSYM=Octets × 8 × 11)
   if Rate=2M (NSYM=Octets × 4 × 11)
   if Rate=5.5M (NSYM=Octets × 8 × 2)
   if Rate=11M ((NSYM=Octets × 8)
   The same number of tokens are simultaneously produced by pin BurstOut.
2. The MaxSearchDelay value range for searching fading channels must be set larger than the sum of system delay and maximum multipath delay.
3. The transmitter transmits burst-by-burst in ADS. The burst sequence is a continuous stream. (The 802.11 burst is transmitted burst-by-burst.) The transmitted consecutive bursts are independent.
   The estimated frequency offset ($\Delta f_i$) of each received burst is input at DeltaF. To avoid estimated frequency offset impacting the next bursts in the frequency compensator, the FreqOffset parameter in this model is set as the actual frequency offset between transmitter and receiver. When the ith burst is processed, the actual phase of previous i-1 bursts is calculated and removed by FreqOffset. The ith estimated frequency offset ($\Delta f_i$) is used to compensate for the phase in current burst only.
   Assume $x_0$, $x_1$, ... , $x_{Total-1}$ sequences are the BurstIn burst signals after removing the actual phase of previous *i* -1 bursts caused by the actual frequency offset from FreqOffset. The $y_0$, $y_1$, ... , $y_{Total-1}$ sequences (whose phase caused by frequency offset) are removed:

   $$y_k = x_k \times e^{-j2\pi\Delta f_i kT_s}$$

   where

   $\Delta f_i$ is frequency offset of ith received burst, which is the input at *DeltaF*

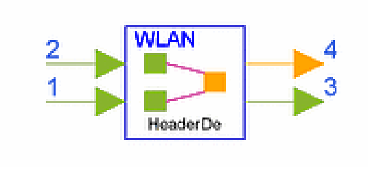
   $$T_s = \frac{1}{11MHz \times SampsPerChip}$$

   is the sample time interval in 802.11b system.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_HeaderDemap



**Description Header demapper**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_HeaderDemap**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | HeadIn | PLCP header modulation signal | complex |
| 2 | InitPhase | initial signal from preamble demapper | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | phase | last complex mapping signal | complex |
| 4 | HeadOut | PLCP header signal | int |

## Notes/Equations

1. This model outputs the PLCP header demapping signal and phase. The long PLCP header uses 1 Mbit/s DBPSK modulation; the short PLCP header uses 2 Mbit/s DQPSK modulation.
2. The initial phase of DBPSK or DQPSK signal is given by pin InitPhase. DBPSK encoding is listed in the following table.

### DBPSK Encoding

| Bit input | Phase change (+jw) |
|---|---|
| 0 | 0 |
| 1 | π |

DQPSK encoding is listed in the following table.

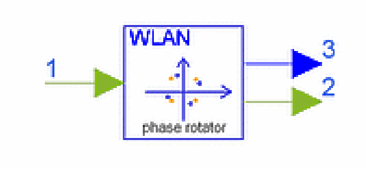
### DQPSK Encoding

| Dibit pattern (d0,d1) | Phase change (+jw) |
|---|---|
| 00 | 0 |
| 01 | π/2 |
| 11 | π |
| 10 | 3 × π/2 (-π/2) |

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_PhaseRotator



**Description 11b phase rotator after decision feedback equalizer**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_PhaseRotator**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | enum | |
| ModType | modulation type: CCK, PBCC | CCK | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | BurstIn | input signal before phase compensator | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | BurstOut | output data after phase compensator | complex |
| 3 | CommPhase | common phase for the equalized signal | real |

## Notes/Equations

1. This model implements phase rotator after decision feedback equalizer for equalized 802.11b burst.
2. Rate, ModType, PLCPType and Octets parameters are used to specify the number of complex signals in one burst. In the WLAN Design Library, one real 802.11b burst consists of PLCP Preamble (Long or Short), PLCP Header (Long or Short) and PSDU. The length of PLCP Preamble (mPLCP) is determined by PLCPType parameter:

$$mPLCP = \begin{cases} 144 \times 11 & ifPLCPType = Long \\ 72 \times 11 & ifPLCPType = Short \end{cases}$$

   The length of Header (mHeader) is determined by PLCPType parameter:

$$mHeader = \begin{cases} 48 \times 11 & ifPLCPType = Long \\ 24 \times 11 & ifPLCPType = Short \end{cases}$$

   The length of PSDU (mPSDU) is determined by Rate parameter:

$$mPSDU = \begin{cases} Octets \times 8 \times 11 & ifRate\ =\ 1Mbps \\ Octets \times 4 \times 11 & ifRate\ =\ 2Mbps \\ Octets \times 2 \times 8 & ifRate\ =\ 5.5Mbps \\ Octets \times 1 \times 8 & ifRate\ =\ 11Mbps \end{cases}$$

So, the total number of one 802.11b burst is as follows:

$$Total\ =\ mPLCP + mHeader + mPSDU$$

Each firing, Total number of tokens were input in BurstIn and Total number of tokens were output in BurstOut pin.

3. Pin 1 (BurstIn) equalized sequences $x$ 0, $x$ 1, ... , $x_{Total}$ -1 input; pin 2 (BurstOut) sequences $y$ 0, $y$ 1, ... , $y_{Total}$ -1 output after phase rotator removes the common phase; pin 3 (CommPhase) is the common phase of equalized sequence output.
4. This model must work behind WLAN_11bDFE when the equalizer converges. The constellation of CCK is the same as that of QPSK. The PSDU and Header parts equalized signals are used to detect the common phase, the Preamble part is not used it is used to train the coefficients of the equalizer and the equalizer algorithm does not converge. The common phase detection algorithm works after the equalizer converges; the common phase detection algorithm is as follows:

```
phase1 =0.0;
phase2 =0.0;
phase3 =0.0;
phase4 =0.0;
first_1 = 0;
second_2 = 0;
third_3 = 0;
fourth_4 = 0;
for (i=0; i<m_inlength-m_prmblength;i++)
{
   Complex xx;_
   xx = BurstIn%(Pin-);
     if ((real(xx)>0.0)&&(imag(xx)>0.0))
 {
   phase1+=arg(xx);
   first_1++;
 }
 if ((real(xx)<0.0)&&(imag(xx)>0.0))
 {
  phase2+=arg(xx);
  second_2++;
 }
 if ((real(xx)<0.0)&&(imag(xx)<0.0))
 {
   phase3+=arg(xx);
   third_3++;
 }
 if ((real(xx)>0.0)&&(imag(xx)<0.0))
 {
   phase4+=arg(xx);
     fourth_4++;
 }
}
phase1 = phase1/first_1;
```

```
phase1 = phase1-PI/4;
phase2 = phase2/second_2;
phase2 = phase2-3*PI/4;
phase3 = phase3/third_3;
phase3 = phase3+3*PI/4;
phase4 = phase4/forth_4;
phase4 = phase4+PI/4;
double phase;
phase =(phase1+phase2+phase3+phase4)/4.0;
```

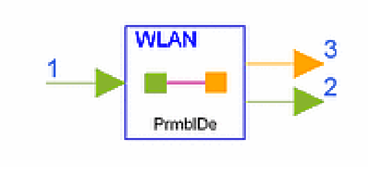
After getting the detected common phase φ above, the common phase can be removed by multiply $e^{-j\varphi}$.

$$y_k = x_k \times e^{-j\varphi}$$

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_PrmblDemap



**Description Preamble demapper**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_PrmblDemap**

### Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum | [0, 1] |
| InitPhase | initial phase of DBPSK | PI / 4 | real | [0, 2π ) |

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PrmblIn | DBPSK signal | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | phase | last DBPSK signal | complex |
| 3 | PrmblOut | PLCP preamble signal | int |

### Notes/Equations

1. This model outputs the PLCP preamble demapping bits and the last complex signal. Both long PLCP preamble and short PLCP preamble use 1 Mbit/s DBPSK modulation. The InitPhase parameter specifies the initial phase of the DBPSK signal. DBPSK encoding is given in the following table.

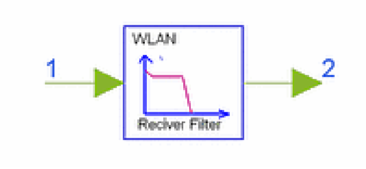
**DBPSK Encoding**

| Bit Input | Phase Change (+jw) |
|---|---|
| 0 | 0 |
| 1 | π |

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_RecFilter



**Description Receiver matched filter**
**Library WLAN, 11b Receiver**
**Class SDFWLAN_RecFilter**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| FilterType | receiver matched filter type: Rectangle, Gaussian, Root-Cosine, Off | Off | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| SampsPerChip | samples per chip | 2 | | int | [2, 8] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| RampTime | power on and off ramp time | 2.0usec | sec | real | [0usec, 1000usec] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| IdleInterval | idle time | 50.0usec | sec | real | [10usec, 1000usec] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | DataIn | input data | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | DataOut | output data | complex |

## Notes/Equations

1. This receiver model matches the transmitter filter. Each firing, data from the RF demodulator with ramp, PPDU, and idle is input; data is then filtered sample-by-sample.
   If FilterType=Off, the input signal will flow directly without filtering. If the transmitter shaping filter is set to Root-Cosine, set the receiver filter to Root-Cosine to obtain an ISI-free signal; if the transmitter shaping filter is set to another filter type, set the receiver filter to Off to avoid additional ISI.
   The impulse response of the rectangle filter can be given by

2. $$h(t) = \frac{\sin((\pi t)/T_s)}{(\pi t)/T_s}$$

   where *Ts* is symbol interval.
3. The impulse response of Gaussian filter can be given by

   $$g(t) = \left(erf\left(\pi \times BT \frac{Ts - 2t}{Ts\sqrt{\ln(4)}}\right) + erf\left(\pi \times BT \frac{Ts + 2t}{Ts\sqrt{\ln(4)}}\right)\right)$$

   and

   $$erf(t) = \frac{2}{\sqrt{\pi}}\int_0^t e^{-\tau^2} d\tau$$

   where *BT* is the BT parameter and *Ts* is the symbol interval.
4. The impulse response of raised-cosine rolloff filter can be given by

   $$h(t) = \left(\frac{\sin((\pi t)/(Ts))}{\pi t}\right)\left(\frac{\cos((\pi\alpha t)/(Ts))}{1 - ((4\alpha t)/(2Ts))^2}\right)$$

   where α is the Alpha parameter and *Ts* is the symbol interval.
5. The impulse response of the root raised-cosine filter has the following relationship with the raised-cosine filter:

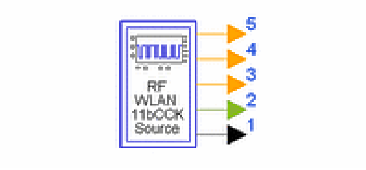
   $$h ( n ) = h\,1( n ) \text{ ƒ } h\,1( n )$$

   where *h(n)* is the impulse response of raised filter and *h1(n)* is root-raised.
6. This filter does not implement interpolating or decimating operations. The output signal has the same sample rate as the input signal.
7. Taps is the filter length and specifies the number of symbol periods to be used in the calculation of the symbol.
8. Alpha specifies the sharpness of a root cosine filter when FilterType=Root-Cosine.
9. BT is the Gaussian filter coefficient; it specifies the extent of signal filtering. *B* is the 3 dB bandwidth of the filter; *T* is the duration of the symbol period. Common values for BT are 0.3 to 0.5.
10. Rate specifies the transmitted data rate.
11. PLCPType specifies the format of the preamble/header sections of the framed signal.
12. Octets specifies the number of data bytes per burst (note that it is in bytes; to transform the value into bits, multiply by 8).
13. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
14. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.

# 11b Signal Sources

- *WLAN 11bCCK RF* (wlan)
- *WLAN 11bCCKSignalSrc* (wlan)
- *WLAN 11bCCKSignalSrc1* (wlan)
- *WLAN 11bMuxBurst* (wlan)
- *WLAN 11bPBCCSignalSrc* (wlan)
- *WLAN 11bScrambler* (wlan)
- *WLAN 11SignalSrc* (wlan)
- *WLAN 802 11bRF* (wlan)
- *WLAN Barker* (wlan)
- *WLAN CCKMod* (wlan)
- *WLAN CRC* (wlan)
- *WLAN HeaderMap* (wlan)
- *WLAN IdlePadding* (wlan)
- *WLAN MuxPLCP* (wlan)
- *WLAN PBCCConvCoder* (wlan)
- *WLAN PBCCMod* (wlan)
- *WLAN PLCPHeader* (wlan)
- *WLAN PLCPPreamble* (wlan)
- *WLAN PreambleMap* (wlan)
- *WLAN PSDUMap* (wlan)
- *WLAN TransFilter* (wlan)

# WLAN_11bCCK_RF



**Description RF Signal source of IEEE 802.11b with idle and CCK modulation**
**Library WLAN, 11b Signal Source**
**Class TSDFWLAN_11bCCK_RF**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| FCarrier | carrier frequency | 2400MHz | Hz | real | (0, ∞) |
| Power | modulator output power | 40mW | W | real | [0, ∞) |
| VRef | reference voltage for output power calibration | 0.1122V | V | real | (0, ∞) |
| PhasePolarity | if set to Invert, Q channel signal is inverted: Normal, Invert | Normal | | enum | |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| I_OriginOffset | I origin offset in percent with respect to output rms voltage | 0.0 | | real | (-∞, ∞) |
| Q_OriginOffset | Q origin offset in percent with respect to output rms voltage | 0.0 | | real | (-∞, ∞) |
| IQ_Rotation | IQ rotation, in degrees | 0.0 | | real | (-∞, ∞) |
| NDensity | noise spectral density at output, in dBm/Hz | -173.975 | | real | (-∞, ∞) |
| Type | type of bit sequence, random or pseudo random: Random, Prbs | Random | | enum | |
| ProbOfZero | probability of bit value being zero (used when Type=Random) | 0.5 | | real | [0, 1] |
| LFSR_Length | Linear Feedback Shift Register length (used when Type=Prbs) | 12 | | int | [2, 31] |
| LFSR_InitState | Linear Feedback Shift Register initial state (used when Type=Prbs) | 1 | | int | [1, pow (2, LFSR_Length) -1] |
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| ClocksBit | locked clocks bit: Not, Locked | Locked | | enum | |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Gaussian | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |

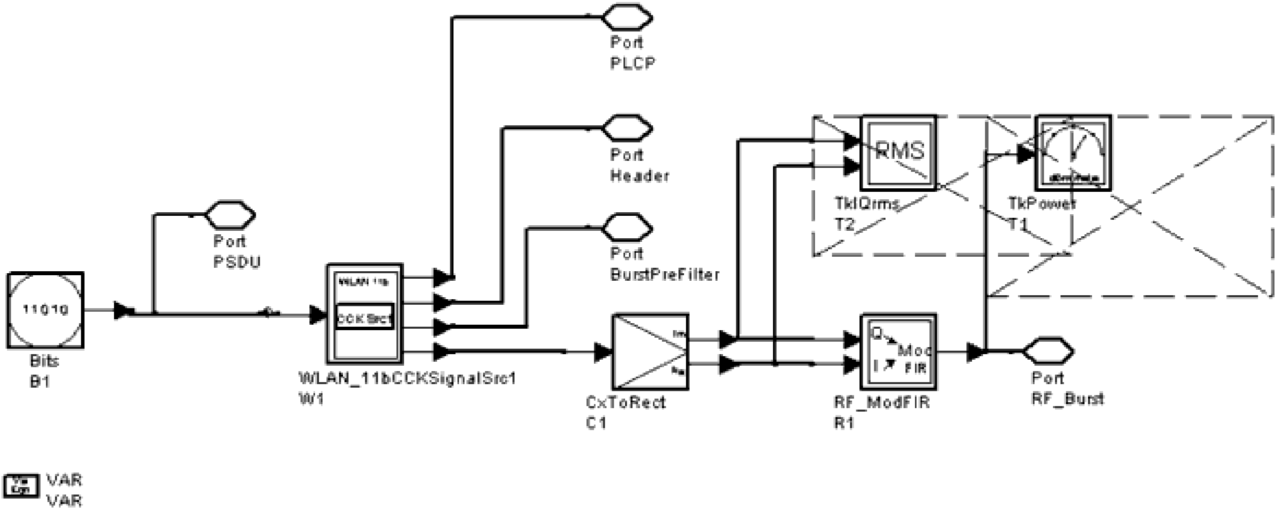
† for each array element: array size must be 7.

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Burst | RF signal of IEEE802.11b burst with idle | timed |
| 2 | BurstPreFilter | IEEE802.11b burst without idle | complex |
| 3 | Header | header bits | int |
| 4 | PLCP | PLCP bits | int |
| 5 | PSDU | PSDU bits | int |

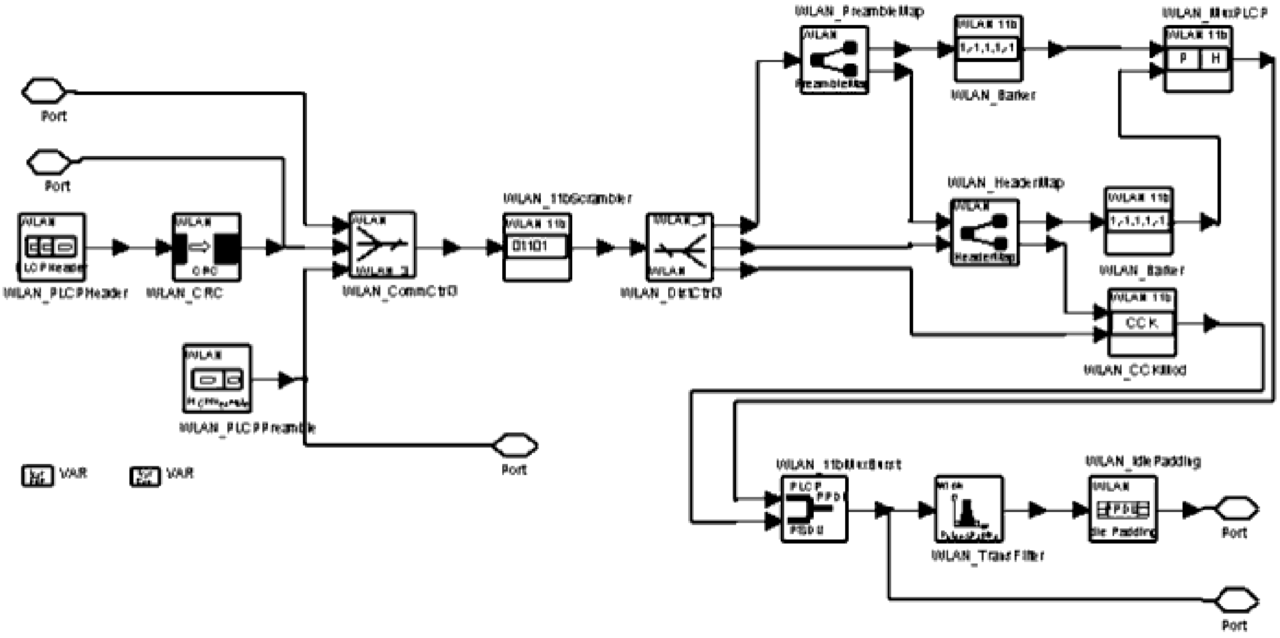
### Notes/Equations

1. This subnetwork is used to generate an RF signal; the baseband signal is sent to an IQ RF modulator and the RF signal is generated. For ease of testing, five signals are output:
    - RF_Burst is RF signal of IEEE802.11b burst with idle
    - BurstPreFilter is the signal generated before the shaping filter (note that this signal is digital)
    - Header outputs the header bits according to IEEE802.11b
    - PLCP outputs the PLCP bits according to IEEE802.11b
    - PSDU outputs the payload data bits that are used in the BER/PER test
2. This subnetwork integrates a baseband transmitter and RF modulator; the schematic is shown in the following figure.



**WLAN_11bCCK_RF Schematic**

In this subnetwork, the baseband generation block includes function blocks that are essential to the 11b baseband signal, such as preamble, header and PSDU generation, signal scrambling, DBPSK and DQPSK mapping, CCK modulating, ramp time and idle time attaching; pulse shaping is attached as the final block to reduce the transmitted bandwidth, thereby increasing spectral efficiency.
The IEEE 802.11b CCK baseband signal is generated by the WLAN_11bCCKSignalSrc1 subnetwork; the schematic is shown in the following figure.



**WLAN_11bCCKSignalSrc1 Schematic**

3. Model TkIQrms and TkPower are used to calibrate the output power. When these are activated and simulated, the result shown in "input IQ signal rms value" should be the VRef for WLAN_11bCCK_RF. If VRef is set correctly, the output power is the designated Power, as can be seen in the "Modulator output power in dBm".
4. The GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters are used to add certain impairments to the ideal output RF signal. Impairments are added in the order described here.

   The unimpaired RF I and Q envelope voltages have gain and phase imbalance applied. The RF is given by:

   $$V_{RF}(t) = A\left(V_I(t)\cos(\omega_c t) - gV_Q(t)\sin\left(\omega_c t + \frac{\phi\pi}{180}\right)\right)$$

   where $A$ is a scaling factor that depends on the Power and ROut parameters specified by the designer, $V_I(t)$ is the in-phase RF envelope, $V_Q(t)$ is the quadrature phase RF envelope, $g$ is the gain imbalance

   $$g = 10^{\frac{GainImbalance}{20}}$$

   and, φ (in degrees) is the phase imbalance.

   Next, the signal $V_{RF}(t)$ is rotated by IQ_Rotation degrees. The I_OriginOffset and Q_OriginOffset are then applied to the rotated signal. Note that the amounts specified are percentages with respect to the output rms voltage. The output rms voltage is given by

   $$\sqrt{2 \times ROut \times Power}$$
5. The PhasePolarity parameter is used to invert the polarity of the Q channel signal before modulation. Depending on the configuration and number of mixers in the transmitter and receiver, the output of the demodulator may be inverted. If such a configuration is used, the Q channel signal can be correctly recovered by setting this

parameter to Invert.

6. The VRef parameter is used to calibrate the modulator. VRef is the input voltage value that results in an instantaneous output power on a matched load equal to P. In order to get an average output power on a matched load equal to P, the input rms voltage must equal VRef. Therefore, in order to calibrate the modulator, VRef must be set to the input rms voltage.
7. Rate is used to determine the transmitted data rate. It can be chosen from the lists of 5.5 Mbps and 11 Mbps.
8. PLCPType is used to select the format of the preamble/header sections of the framed signal, Long and Short can be selected.
9. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
10. ClocksBit enables users to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame; it is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator, and the designer can set this bit. If ClocksBit=Locked, the clock bit is 1 (otherwise it is 0).
11. InitPhase specifies the initial phase of the DBPSK signal. The default value is set to PI/4.0.
12. ScramblerInit indicates the initial state of scrambler, in WLAN 11b specification, this value is set to 1101100.
13. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.
14. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
15. OverSampling indicates the oversampling ratio of transmission signal. For example, if OverSampling = Ratio_4, the transmission signal is upsampled with 4 times. Oversampling ratios from 2 to 9 are supported.
16. IdleInterval indicates the idle time added between two consecutive bursts, which is in [0, 1000 μsec].
17. FilterType specifies a baseband filter that is used to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:
    - NoneFilter No transmitter filter is used.
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine just how much of the inter-symbol interference can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian-shaped in both the time and frequency domains; it does not ring like the root-cosine filters ring. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root-Cosine Root-cosine filters (also referred to as square root raised-cosine filters) have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI does exist at all times except at symbol (decision) times.
    - Ideal-Lowpass In the frequency domain, this filter appears as a lowpass,

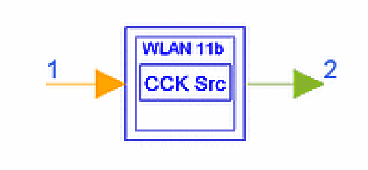
rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.

18. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    - The Gaussian filter has a rapidly decaying impulse response, so a filter length of 6 is recommended. Greater lengths have negligible effects on the accuracy of the signal.
    - The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended. Beyond this, the ringing has negligible effects on the accuracy of the signal.
    - The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
      For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
19. Alpha is to set the sharpness of a root-cosine filter when FilterType=Root-Cosine.
20. BT is the Gaussian filter coefficient. The B is the 3 dB bandwidth of the filter and T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_11bCCKSignalSrc



**Description Signal source of IEEE 802.11b with idle and CCK modulation**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11bCCKSignalSrc**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| ClocksBit | locked clocks bit: Not, Locked | Locked | | enum | |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Gaussian | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |

† for each array element: array size must be 7.

## Pin Inputs

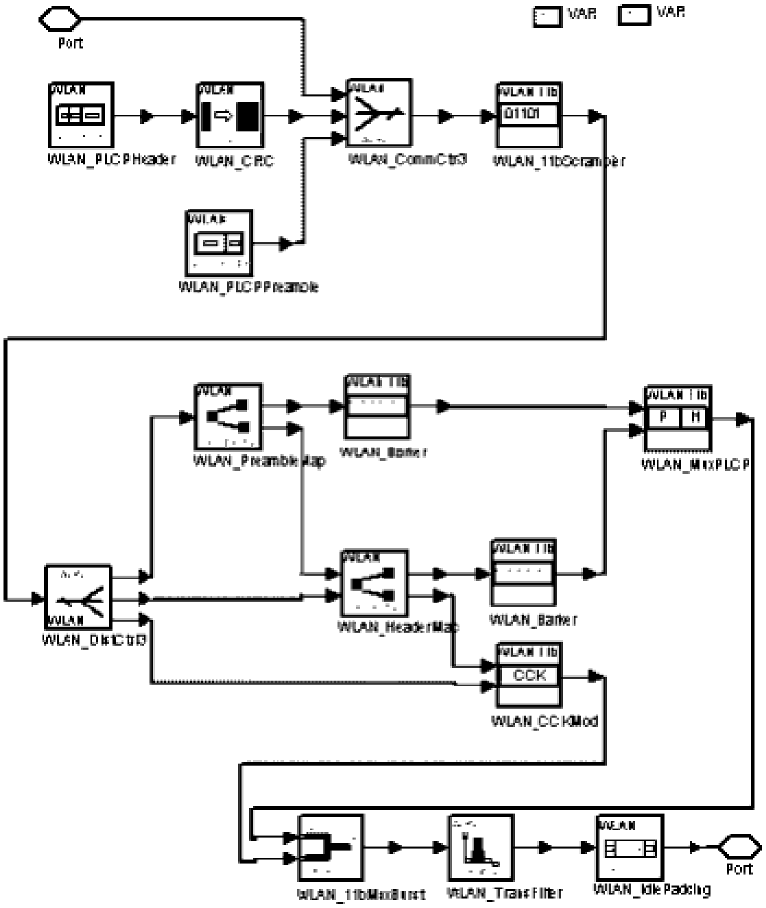
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE802.11b burst | complex |

## Notes/Equations

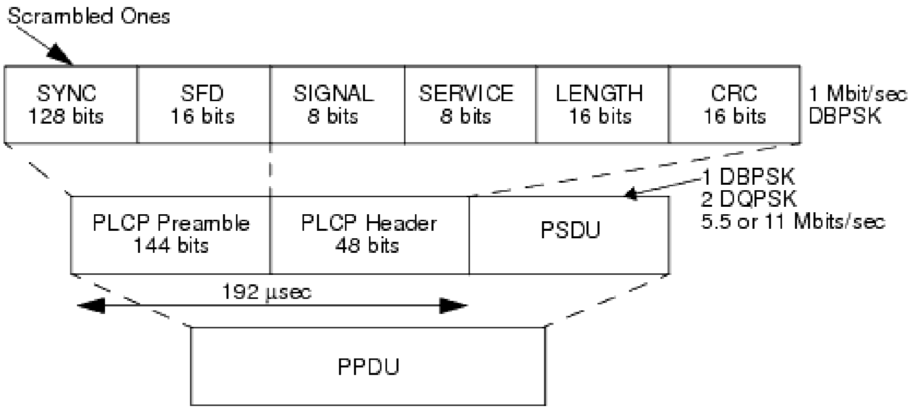
1. This model is used to generate a CCK baseband signal according to IEEE 802.11b. Functions are implemented that are essential to an 11b baseband signal including preamble, header and PSDU generation, signal scrambling, DBPSK and DQPSK mapping, CCK modulation, ramp time and idle time attaching; pulse shaping is attached to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The schematic is shown in the following figure.



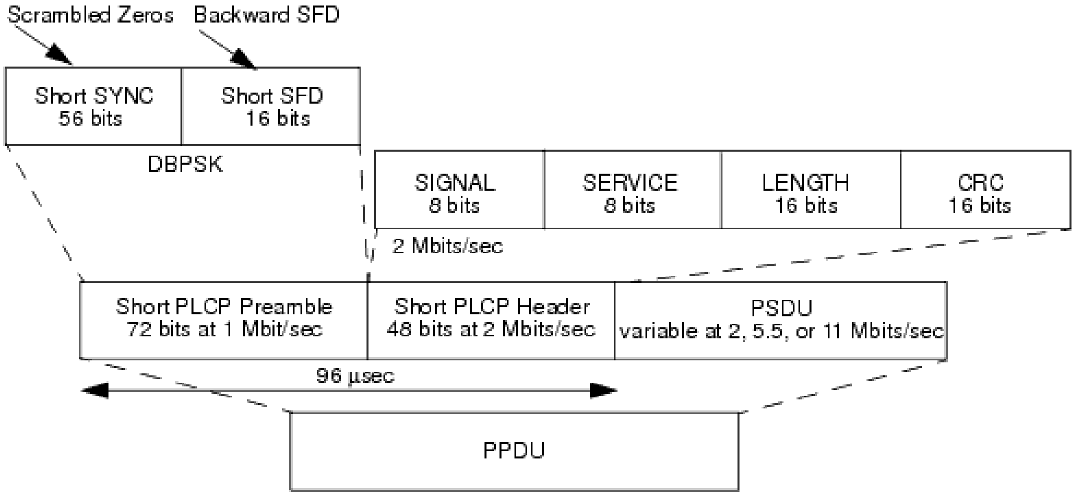
**WLAN_80211bCCKSignalSrc Schematic**

2. Rate is used to determine the transmitted data rate. It can be chosen from the lists of 5.5 Mbps and 11 Mbps.
3. PLCPType is used to select the format of the preamble/header sections of the framed signal, Long and Short can be selected.
4. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
5. ClocksBit enables users to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame. This bit is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator, and the designer can set this bit. If ClocksBit=Locked, the clock bit is 1 (otherwise it is 0).
6. The InitPhase parameter specifies the initial phase of the DBPSK signal. The default value is PI/4.0.
7. ScramblerInit indicates the initial state of scrambler, in WLAN 11b specification, this value is 1101100.
8. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.

9. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
10. OverSampling indicates the oversampling ratio of transmission signal. For example, if OverSampling = Ratio_4, the transmission signal is upsampled with 4 times. Oversampling ratios ranging from 2 to 9 are supported.
11. IdleInterval indicates the idle time added between two consecutive bursts, which is in [0, 1000 μsec].
12. FilterType specifies a baseband filter to be applied to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:
    - NoneFilter No transmitter filter is used.
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine just how much of the ISI can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian shaped in both the time and frequency domains, and it does not ring like root-cosine filters ring. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root-Cosine Root-cosine filters (also referred to as square root raised-cosine, filters) have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI does exist at all times except the symbol (decision) times.
    - Ideal-Lowpass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.
13. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    The Gaussian filter has a rapidly decaying impulse response, so a filter length of 6 is recommended. Greater lengths have negligible effects on the accuracy of the signal.
    The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended; beyond this, the ringing has negligible effects on the accuracy of the signal.
    The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
    For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
14. Alpha is to set the sharpness of a root-cosine filter when FilterType=Root-Cosine.
15. BT is the Gaussian filter coefficient. B is the 3 dB bandwidth of the filter and T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.
16. As illustrated in the following figures, one PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

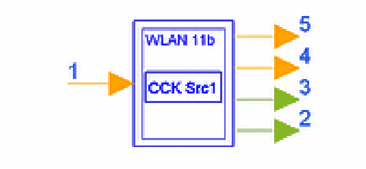


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bCCKSignalSrc1



**Description Signal source of IEEE 802.11b with idle and CCK modulation**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11bCCKSignalSrc1**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| ClocksBit | locked clocks bit: Not, Locked | Locked | | enum | |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Gaussian | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000) |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |

† for each array element: array size must be 7.

## Pin Inputs

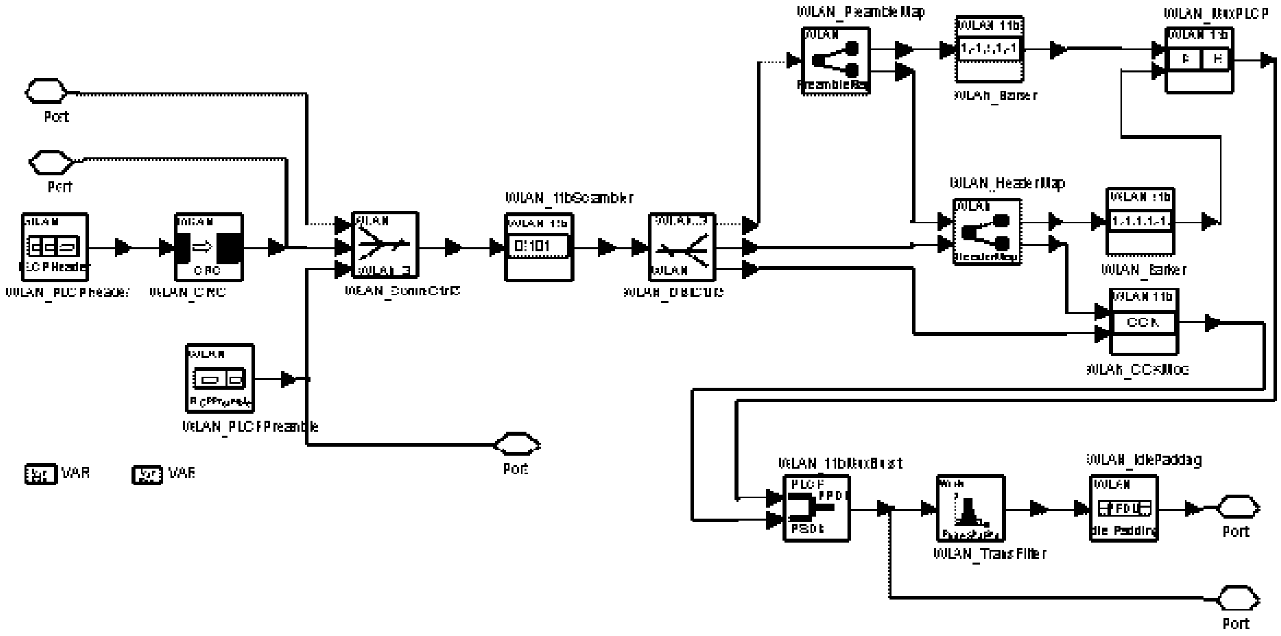
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE802.11b burst with idle | complex |
| 3 | BurstPreFilter | IEEE802.11b burst without idle | complex |
| 4 | Header | header bits | int |
| 5 | PLCP | PLCP bits | int |

## Notes/Equations

1. This model is used to generate a CCK baseband signal according to IEEE 802.11b. Functions are implemented that are essential to an 11b baseband signal including preamble, header and PSDU generation, signal scrambling, DBPSK and DQPSK mapping, CCK modulation, ramp time and idle time attaching; pulse shaping is attached as the final block to reduce the transmitted bandwidth, thereby increasing spectral efficiency.
   The schematic for this subnetwork is shown in the following figure.



**WLAN_11bCCKSignalSrc1 Schematic**

2. Rate is used to determine the transmitted data rate. It can be chosen from the lists of 5.5 Mbps and 11 Mbps.
3. PLCPType is used to select the format of the preamble/header sections of the framed signal, Long and Short can be selected.
4. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
5. ClocksBit enables users to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame. This bit is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator, and the designer can set this bit. If ClocksBit=Locked, the clock bit is 1 (otherwise it is 0).
6. The InitPhase parameter specifies the initial phase of the DBPSK signal. The default value is set to PI/4.0.
7. ScramblerInit indicates the initial state of scrambler, in WLAN 11b specification, this value is set to 1101100.
8. PwrType specifies the pattern for generating the ramp signal: None, Linear, or

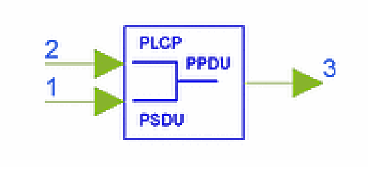
Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.

9. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
10. OverSampling indicates the oversampling ratio of transmission signal. For example, if OverSampling = Ratio_4, it means the transmission signal is upsampled with 4 times. There are 8 kinds of oversampling ratios, ranged from 2 to 9, to be supported.
11. IdleInterval indicates the idle time added between two consecutive bursts, which is in [0, 1000 µsec].
12. FilterType specifies a baseband filter that is used to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:
    - NoneFilter No transmitter filter is used.
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine just how much of the inter-symbol interference can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian-shaped in both the time and frequency domains; it does not ring like the root-cosine filters ring. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root-Cosine Root-cosine filters (also referred to as square root raised-cosine filters) have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI does exist at all times except at symbol (decision) times.
    - Ideal-Lowpass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.
13. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    - The Gaussian filter has a rapidly decaying impulse response, so a filter length of 6 is recommended; greater lengths have negligible effects on the accuracy of the signal.
    - The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended; beyond this, the ringing has negligible effects on the accuracy of the signal.
    - The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
      For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
14. Alpha is to set the sharpness of a root-cosine filter when FilterType=Root-Cosine.
15. BT is the Gaussian filter coefficient; B is the 3 dB bandwidth of the filter and T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_11bMuxBurst



**Description 11b burst multiplexer**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11bMuxBurst**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |

## Pin Inputs

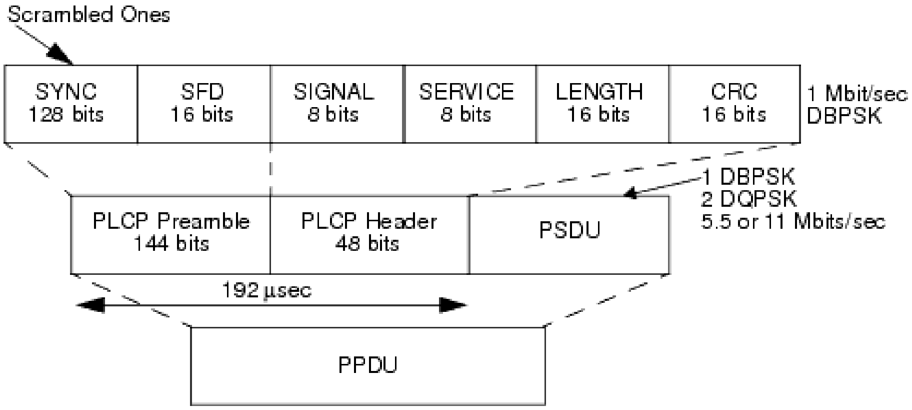
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU | complex |
| 2 | PLCP | PLCP preamble and header | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | Burst | 11b burst signal | complex |

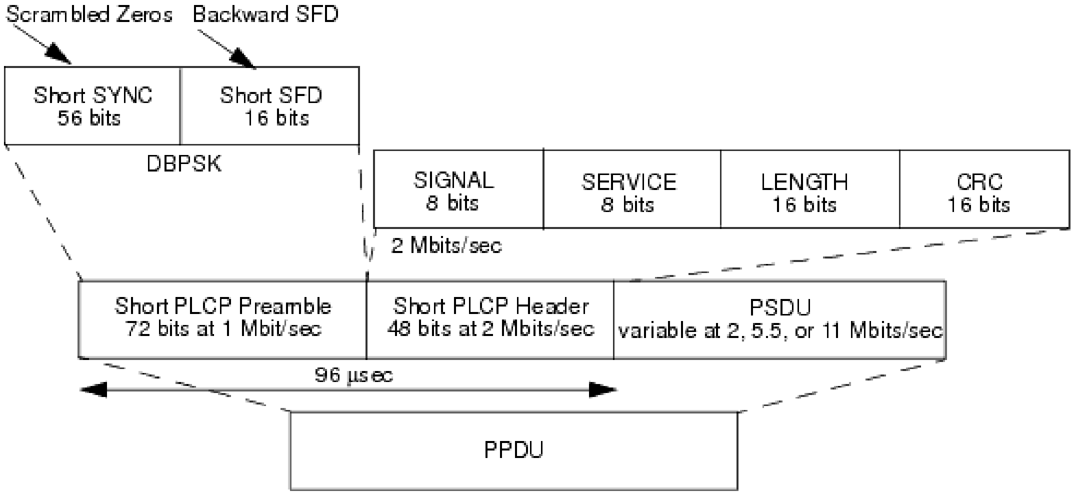
## Notes/Equations

1. This model multiplexes the PLCP including preamble and header and PSDU into a signal burst. This burst is the frame format time. Two different preambles and headers are defined:
    - The mandatory supported long preamble and header, illustrated in the following figure, inter-operates with the current 1 Mbit/s and 2 Mbit/s DSSS specification (as described in the IEEE Standard 802.11, 1999 Edition).



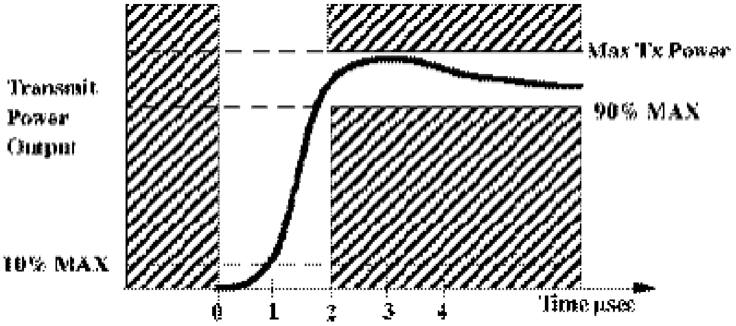
**802.11b Signal Burst with Long PLCP Preamble**

- The optional short preamble and header, illustrated in the following figure, is intended for applications where maximum throughput is desired and inter-operability with legacy and non-short-preamble capable equipment is not a consideration. That is, it is expected to be used only in networks of like equipment that can use the optional mode.

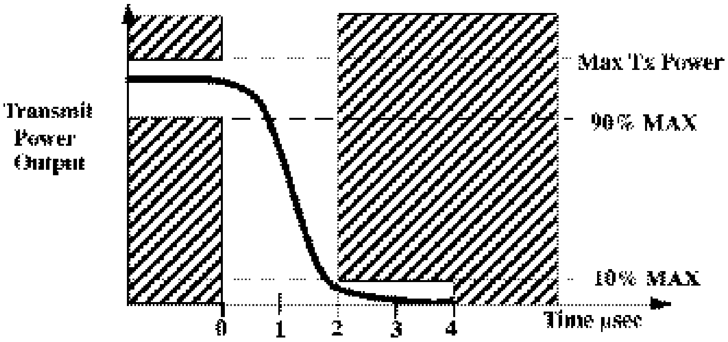


**802.11b Signal Burst with Short PLCP Preamble**

2. Transmit power-on and power-down ramps are implemented as illustrated in the following figures. The RampTime setting is used when PwrType is set to Linear or Cosine.



**Transmit Power-On Ramp**

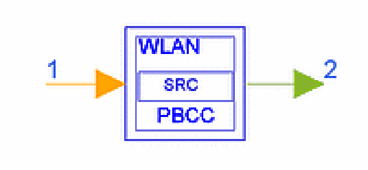


Transmit Power-Down Ramp

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bPBCCSignalSrc



**Description Signal source of IEEE 802.11b with idle and PBCC modulation**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11bPBCCSignalSrc**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| ClocksBit | locked clocks bit: Not, Locked | Locked | | enum | |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2π ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0μsec | sec | real | [0μsec, 1000μsec] |
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Root-Cosine | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |

† for each array element: array size must be 7.

## Pin Inputs

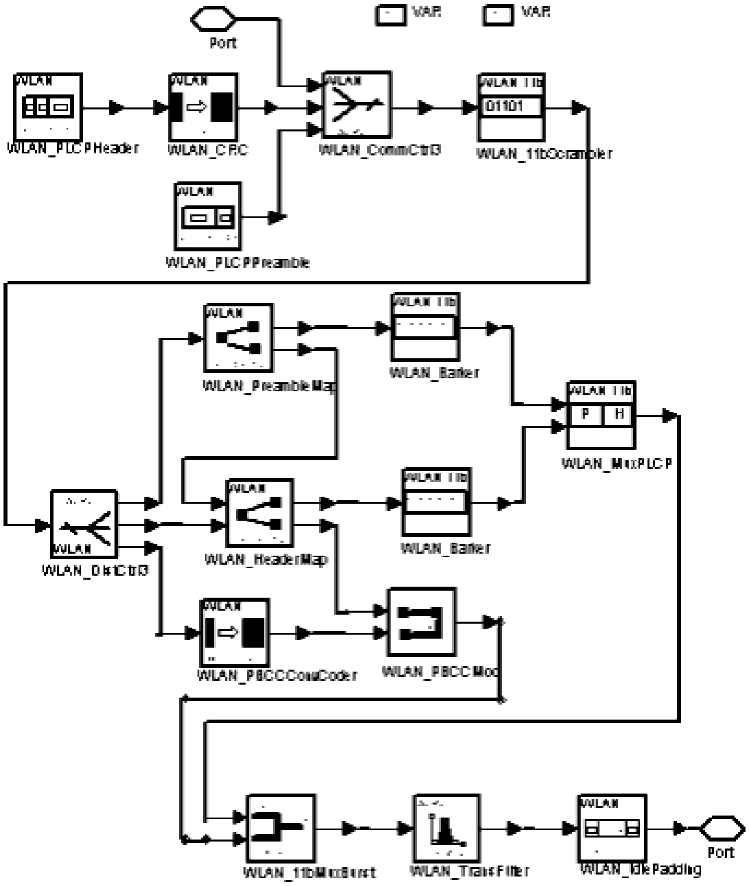
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE802.11 burst | complex |

## Notes/Equations

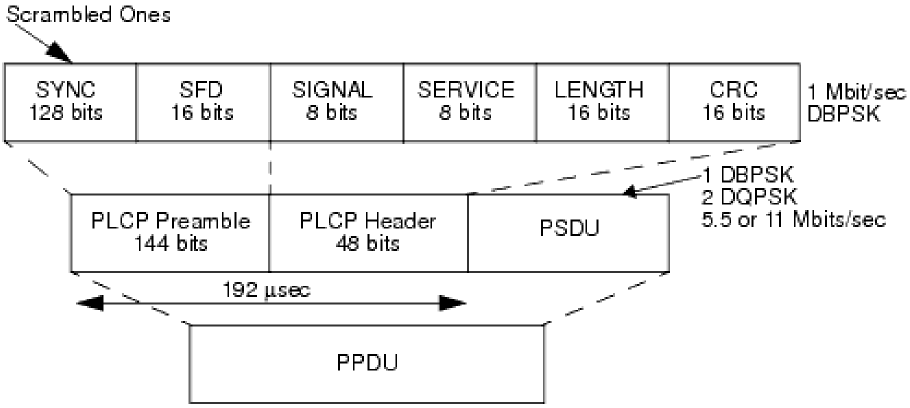
1. This model is used to generate a 5.5 or 11 Mbps DSS/PBCC signal source according to IEEE 802.11b. It performs PLCP preamble and header generation, DBPSK/DQPSK modulation, Barker spreading, PBCC convolutional encoding, multiplexing, and shaping.
   The schematic is shown in the following figure.



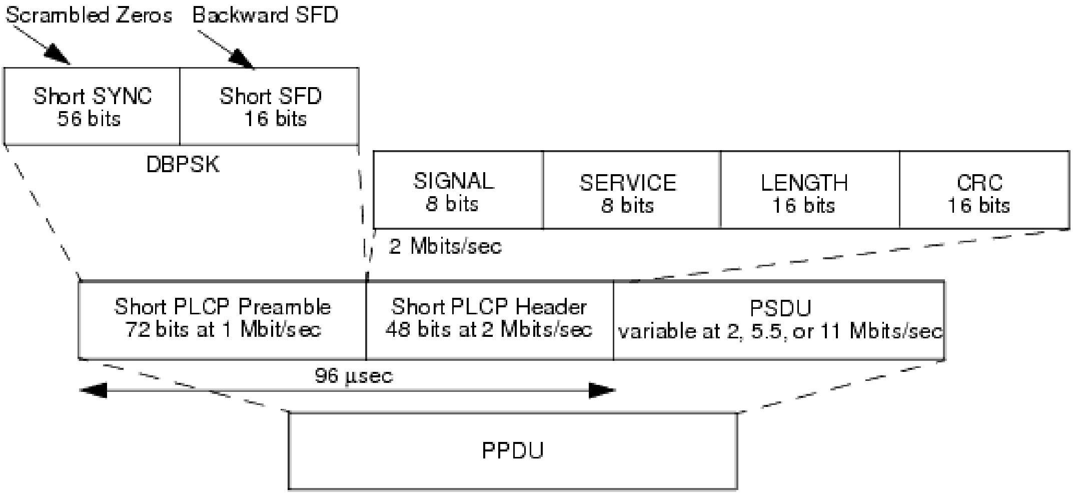
**WLAN_11bPBCCSignalSrc Schematic**

2. Rate is used to determine the transmitted data rate.
3. PLCPType is used to select the format (Long or Short) of the preamble/header sections of the framed signal.
4. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
5. ClocksBit enables users to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame; it is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator, and the designer can set this bit. If ClocksBit=Locked, the clock bit is 1 (otherwise it is 0).
6. InitPhase specifies the initial phase of the DBPSK signal. The default value is set to PI/4.0.
7. ScramblerInit indicates the initial state of scrambler, in WLAN 11b specification, this value is set to 1101100.
8. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.
9. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.

10. OverSampling indicates the oversampling ratio of transmission signal. For example, if OverSampling = Ratio_4, it means the transmission signal is upsampled with 4 times. There are 8 kinds of oversampling ratios, ranged from 2 to 9, to be supported.
11. IdleInterval indicates the idle time added between two consecutive bursts, which is in [0, 1000 μsec].
12. FilterType specifies a baseband filter to be applied to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:
    - NoneFilter No transmitter filter is used.
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine how much of the inter-symbol interference can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian-shaped in both the time and frequency domains; it does not ring like root-cosine filters ring. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root-Cosine Root-cosine (also referred to as square root raised-cosine) filters have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI exists at all times except at symbol (decision) times.
    - Ideal-Lowpass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.
13. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    - The Gaussian filter has a rapidly decaying impulse response, so the filter length of 6 is recommended. Greater lengths have negligible effects on the accuracy of the signal.
    - The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended. Beyond this, the ringing has negligible effects on the accuracy of the signal.
    - The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
      For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
14. Alpha specifies the sharpness of a root-cosine filter when FilterType=Root-Cosine.
15. BT is the Gaussian filter coefficient. B is the 3 dB bandwidth of the filter; T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.
16. As illustrated in the following figures, one PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

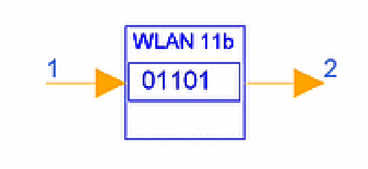


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_11bScrambler



**Description 11b scrambler**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11bScrambler**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| InitState | initial state of scrambler | 1 1 0 1 1 0 0 | int array | 0 or 1 array size is 7 |
| Octets | octet number of PSDU | 100 | int | (0, 2336] |

## Pin Inputs

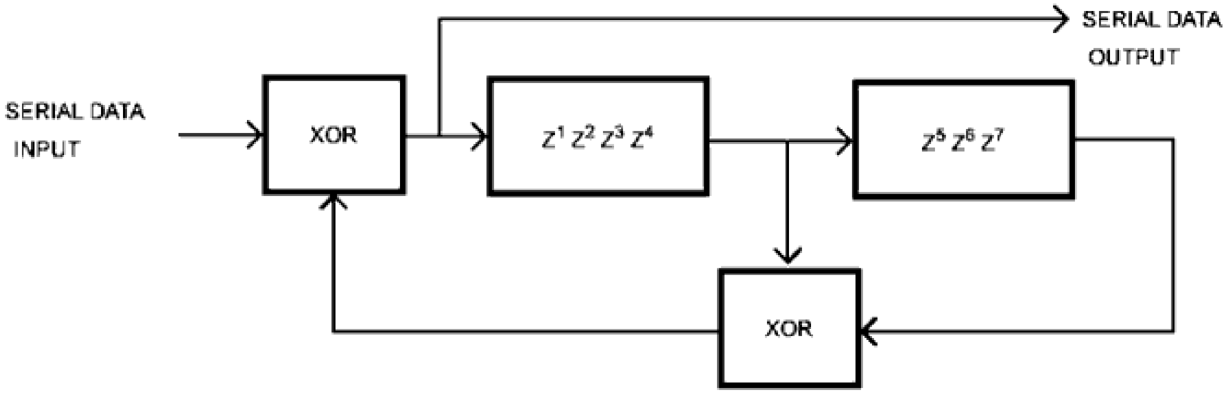
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | bits to be scrambled | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | scrambled bits | int |

## Notes/Equations

1. This component scrambles all data bits transmitted. Each firing, when Octets bytes are consumed and produced, the scrambler is reset to its initial state.
2. The generator polynomial is $G(z) = z^{-7} + z^{-4} + 1$. The following figure illustrates the scrambler.



**Data Scrambler**

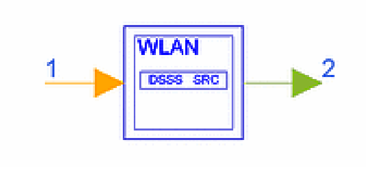
The feed-through configuration of the scrambler and descrambler is self-synchronizing, which requires no prior knowledge of the transmitter initialization of the scrambler for receiver processing.

3. The scrambler also generates the PLCP SYNC field.
   For a long preamble, the SYNC field consists of 128 bits of scrambled 1 bits. The initial state of the scrambler (seed) is [1101100], where the left end bit specifies the value to place in the first delay element $Z^{1}$, and the right end bit specifies the value to place in the last delay element $Z^{7}$.
   For a short preamble, the short SYNC field consists of 56 bits of scrambled 0 bits. The initial state of the scrambler (seed) is [001 1011].

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_11SignalSrc



**Description Signal source of IEEE 802.11 with idle**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_11SignalSrc**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2 | Mbps_1 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| ClocksBit | locked clocks bit: Not, Locked | Locked | | enum | |
| InitPhase | initial phase of DBPSK | 0.785 | | real | [0, 2п ) |
| ScramblerInit | initial state of scrambler | 1 1 0 1 1 0 0 | | int array | {0, 1}† |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| RampTime | power on and off ramp time | 2.0µsec | sec | real | [0µsec, 1000µsec] |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0µsec | sec | real | [0µsec, 1000µsec] |
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Root-Cosine | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000) |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |

† for each array element: array size must be 7.

## Pin Inputs

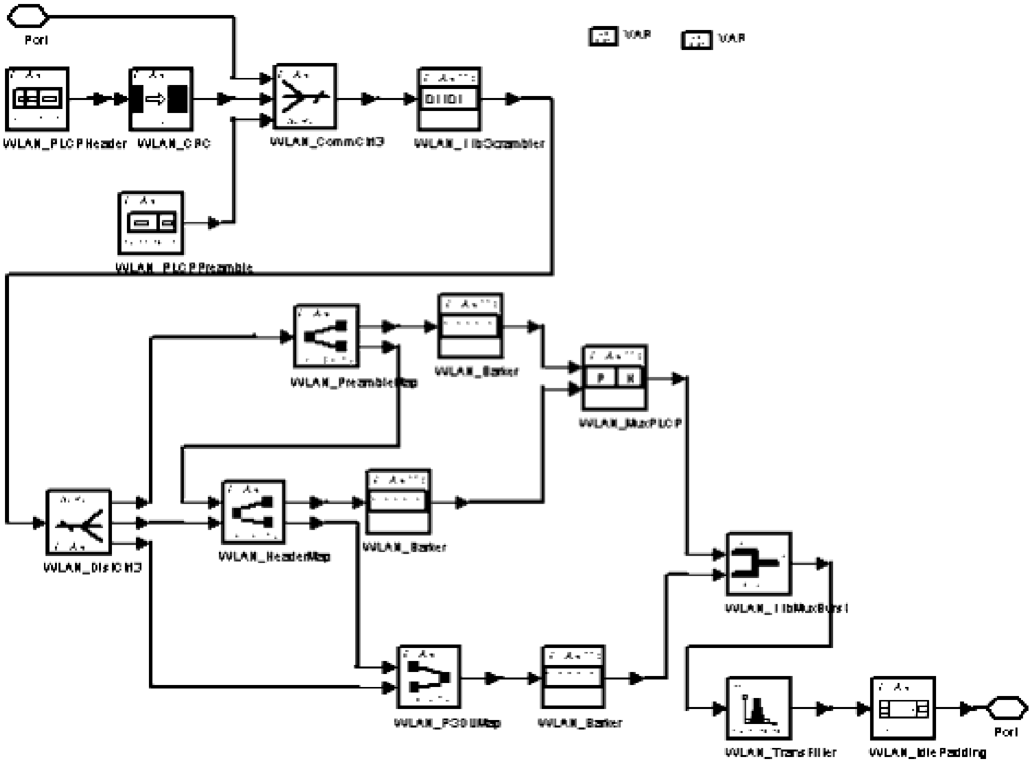
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE 802.11 burst | complex |

### Notes/Equations

1. This subnetwork generates an HR/DSSS and HR/DSSS/short IEEE 802.11b signal source. It performs PLCP preamble and header generation, DBPSK/DQPSK modulation, Barker spreading, multiplexing, and shaping.
   The schematic is shown in the following figure.



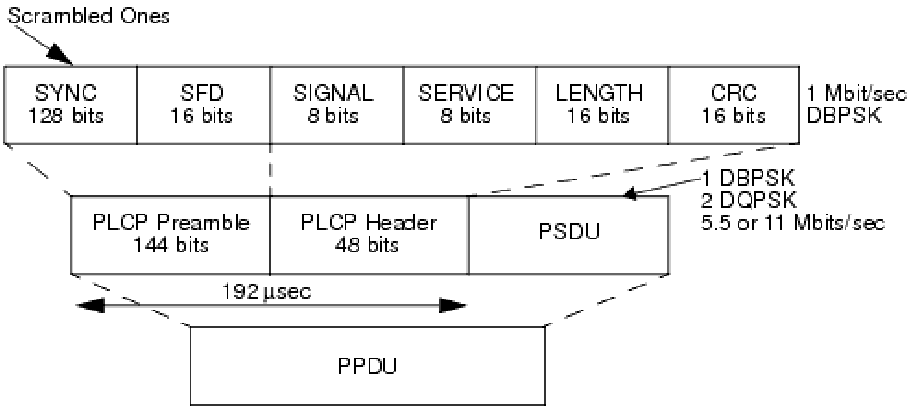
**WLAN_11SignalSrc Schematic**

2. Rate is used to determine the transmitted data rate.
3. PLCPType is used to select the format (Long or Short) of the preamble/header sections of the framed signal.
4. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
5. ClocksBit enables users to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame; it is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator, and the designer can set this bit. If ClocksBit=Locked, the clock bit is 1 (otherwise it is 0).
6. InitPhase specifies the initial phase of the DBPSK signal.
7. ScramblerInit indicates the initial state of scrambler; in the WLAN 11b specification, this value is 1101100.
8. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.
9. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
10. OverSampling indicates the oversampling ratio of transmission signal. For example, if OverSampling = Ratio_4, it means the transmission signal is upsampled with 4 times.
11. IdleInterval indicates the idle time added between two consecutive bursts, which is in [0, 1000 μsec].
12. FilterType specifies which baseband filter is applied to reduce the transmitted

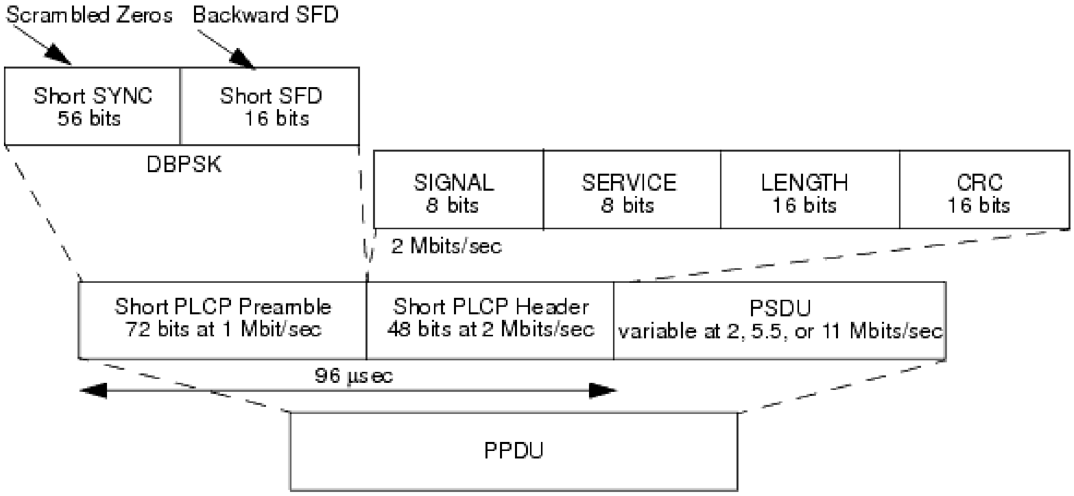
bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:

- NoneFilter No transmitter filter is used.
- Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine just how much of the inter-symbol interference can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian shaped in both the time and frequency domains, and it does not ring like the root-cosine filters do. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
- Root-Cosine Root-cosine filters (also referred to as square root raised cosine filters) have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI exists at all times except at symbol (decision) times.
- Ideal-Lowpass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.

13. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    - The Gaussian filter has a rapidly decaying impulse response, so the filter length of 6 is recommended. Greater lengths have negligible effects on the accuracy of the signal.
    - The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended. Beyond this, the ringing has negligible effects on the accuracy of the signal.
    - The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
      For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
14. Alpha is to set the sharpness of a root-cosine filter when FilterType=Root-Cosine.
15. BT is the Gaussian filter coefficient. The B is the 3 dB bandwidth of the filter and T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.
16. As illustrated in the following figures, one PPDU frame includes PLCP Preamble, PLCP Header, and PSDU.



**Long PLCP PPDU Format**

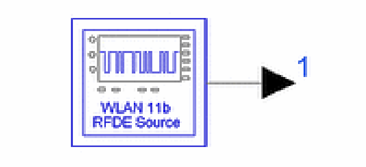


**Short PLCP PPDU Format**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999

# WLAN_802_11bRF



**Description WLAN 802.11b signal source**
**Library WLAN, 11b Signal Source**
**Class TSDFWLAN_802_11bRF**
**Derived From baseARFsource**

## Parameters

| Name | Description | Default | Sym | Unit | Type | Range |
|---|---|---|---|---|---|---|
| ROut | Source resistance | DefaultROut |  | Ohm | real | (0, ∞) |
| RTemp | Temperature | DefaultRTemp |  | Celsius | real | [-273.15, ∞) |
| TStep | Expression showing how TStep is related to the other source parameters | 1/11 MHz/OversamplingRatio |  |  | string |  |
| FCarrier | Carrier frequency: CH1_2412.0M, CH3_2422.0M, CH5_2432.0M, CH6_2437.0M, CH7_2442.0M, CH9_2452.0M, CH11_2462.0M, CH13_2472.0M | CH1_2412.0M |  | Hz | real enum |  |
| Power | Power | 0.04 |  | W | real | [0, ∞) |
| MirrorSpectrum | Mirror spectrum about carrier? NO, YES | NO |  |  | enum |  |
| GainImbalance | Gain imbalance, Q vs I | 0.0 |  | dB | real | (-∞, ∞) |
| PhaseImbalance | Phase imbalance, Q vs I | 0.0 |  | deg | real | (-∞, ∞) |
| I_OriginOffset | I origin offset (percent) | 0.0 |  |  | real | (-∞, ∞) |
| Q_OriginOffset | Q origin offset (percent) | 0.0 |  |  | real | (-∞, ∞) |
| IQ_Rotation | IQ rotation | 0.0 |  | deg | real | (-∞, ∞) |
| OversamplingRatio | Oversampling ratio | 2 | S |  | int | [2, 9] |
| DataRate | Data rate (Mbps): Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_11 | R |  | enum |  |
| Modulation | Modulation format: CCK, PBCC | CCK |  |  | enum |  |
| PreambleFormat | Preamble/header format: Long, Short | Long | H |  | enum |  |
| ClkLockedFlag | Lock header clock? NO, YES | YES |  |  | enum |  |
| PwrRamp | RF power ramp shape: None, Linear, Cosine | None | P |  | enum |  |
| IdleInterval | Burst idle interval | 10.0 μsec | I | sec | real | [0, 1000μsec] |
| FilterType | Shaping filter type: NoneFilter, Gaussian, Root Cosine, Ideal Lowpass | Gaussian |  |  | enum |  |
| RRC_Alpha | RRC roll-off factor | 0.2 |  |  | real | (0.0, 1.0] |
| GaussianFilter_bT | Gaussian filter bT coefficient | 0.3 |  |  | real | (0.0, 1.0] |
| FilterLength | Filter length (chips) | 6 |  |  | int | [1, 200] |
| DataType | Payload data type: PN9, PN15, FIX4, _4_1_4_0, _8_1_8_0, _16_1_16_0, _32_1_32_0, _64_1_64_0 | PN9 |  |  | enum |  |
| DataLength | Data length (bytes per burst) | 100 | L |  | int | [1, 2312] |

## Pin Outputs

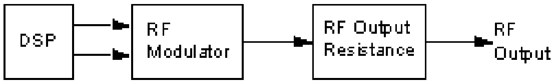
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF | RF output | timed |

## Notes/Equations

1. This WLAN signal source generates an IEEE 802.11b and 802.11g DSSS/CCK/PBCC RF signal.
   To use this source, RF carrier frequency (FCarrier) and power (Power) must be set.

RF impairments can be introduced by setting the ROut, RTemp, MirrorSpectrum, GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters.
802.11b/g signal characteristics an be specified by setting the OversamplingRatio, DataRate, Modulation, PreambleFormat, ClkLockedFlag, PwrRamp, IdleInterval, FilterType, RRC_Alpha, GaussianFilter_bT, FilterLength, DataType, and DataLength parameters.

2. This signal source includes a DSP section, RF modulator, and RF output resistance as illustrated in the following figure.

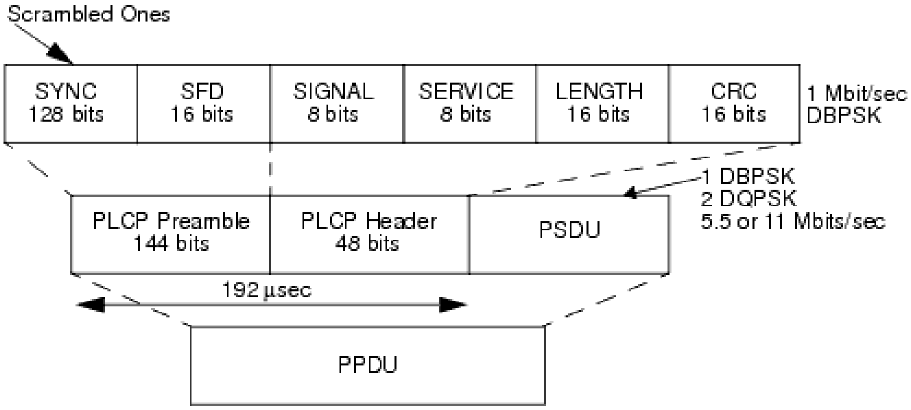


**Source Block Diagram**

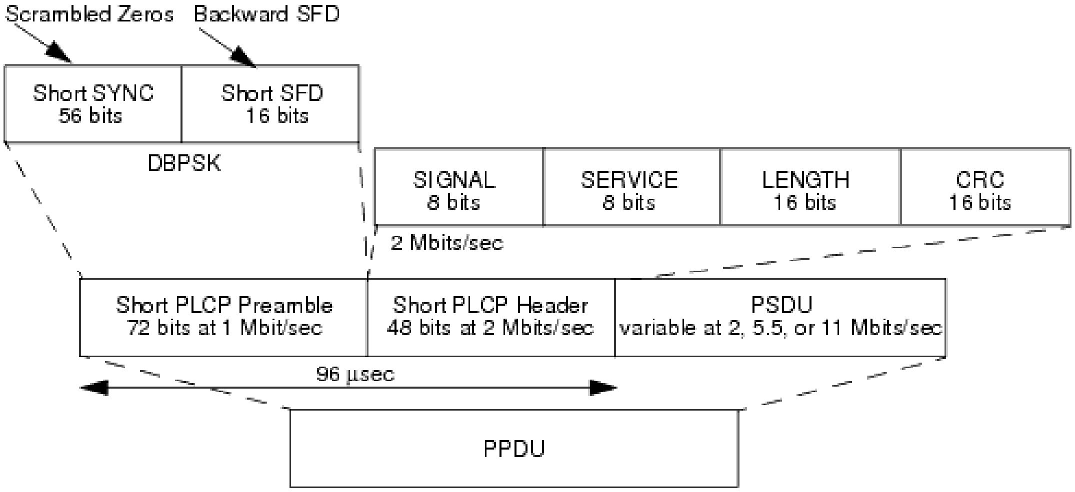
The ROut and RTemp parameters are used by the RF output resistance. The FCarrier, Power, MirrorSpectrum, GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters are used by the RF modulator. The remaining signal source parameters are used by the DSP block.
The RF output from the signal source is at the frequency specified (FCarrier), with the specified source resistance (ROut) and with power (Power) delivered into a matched load of resistance ROut. The RF signal has additive Gaussian noise power set by the resistor temperature (RTemp).

3. This WLAN 802.11b signal source model is compatible with the Agilent Signal Studio Software for 802.11 WLAN Agilent E4438C ESG Vector Signal Generator Option 417 for transmitter test.
Details regarding Signal Studio for WLAN 802.11 are included at the website http://www.agilent.com/find/signalstudio .

4. The 802.11b baseband signal source frame structure is illustrated in the following figures. Each frame is separated by an IdleInterval; one 802.11b frame consists of PLCP Preamble, PLCP Header and Data (PSDU) parts. (PPDU means *physical layer protocol data units*; SFD means *start frame delimiter*; CRC means *cyclic redundancy code*; PLCP means *physical layer convergence procedure*; PSDU means *PLCP service data units*.)



**Long PLCP Frame Format**



**Short PLCP Frame Format**

5. Parameter Details
   - ROut is the RF output source resistance.
   - RTemp is the RF output source resistance temperature in Celsius and sets the noise density in the RF output signal to (k(RTemp+273.15)) Watts/Hz, where k is Boltzmann's constant.
   - FCarrier is the RF output signal frequency.
   - Power is the RF output signal power. The Power of the signal is defined as the average frame power and excludes the idle interval time intervals.
   - MirrorSpectrum is used to mirror the RF_out signal spectrum about the carrier. This is equivalent to conjugating the complex RF envelope voltage.
     Depending on the configuration and number of mixers in an RF transmitter, the RF output signal from hardware RF generators can be inverted. If such an RF signal is desired, set this parameter to YES.
   - GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation are used to add certain impairments to the ideal output RF signal. Impairments are added in the order described here.
     The unimpaired RF I and Q envelope voltages have gain and phase imbalance applied. The RF is given by:

$$V_{RF}(t) = A\left(V_I(t)\cos(\omega_c t) - gV_Q(t)\sin\left(\omega_c t + \frac{\phi\pi}{180}\right)\right)$$

     where *A* is a scaling factor based on Power and ROut parameters specified by the designer, *V* I( *t* ) is the in-phase RF envelope, *V* Q( *t* ) is the quadrature phase RF envelope, *g* is the gain imbalance

$$g = 10^{\frac{GainImbalance}{20}}$$

     and, φ (in degrees) is the phase imbalance.
   - Next, the signal *V* RF( *t* ) is rotated by IQ_Rotation degrees. The I_OriginOffset and Q_OriginOffset are then applied to the rotated signal. Note that the amounts specified are percentages with respect to the output rms voltage. The output rms voltage is given by sqrt(2 × ROut × Power).
   - OversamplingRatio sets the oversampling ratio of 802.11b RF signal source.

Eight oversampling ratios (2, 3, 4, 5, 6, 7, 8, 9) are supported by this source. If OversamplingRatio = 4, the simulation RF bandwidth is larger than the signal bandwidth by a factor of 4 (e.g. for Bandwidth=11 MHz, the simulation RF bandwidth = 11 MHz × 4 = 44 MHz).

- DataRate specifies the data rate; 1, 2, 5.5, and 11 Mbps can be implemented in this source. All data rates are defined in 802.11b specification.
- Modulation is defined as CCK or PBCC, which are two modulation formats in 802.11b. This parameter is useless for 1 Mbps and 2 Mbps data rates. For 5.5 Mbps and 11 Mbps data rates, the two different modulation formats available are CCK and PBCC.
- PreambleFormat is used to set the format of the framed signal preamble/header sections; refer to the previous figures.
- ClkLockedFlag is used to toggle the clock locked flag in the header. This is Bit 2 in the Service field of the PPDU frame. This bit is used to indicate to the receiver if the carrier and the symbol clock use the same local oscillator. If ClkLockedFlag=YES, this bit is set to 1; if ClkLockedFlag=NO, this bit is set to 0.
- PwrRamp is used to select shape of the RF burst in framed mode. Select the power up/down ramp type: Cosine ramp gives least amount of out-of-channel interference; None starts transmitting the signal at full power, and is the simplest power ramp to implement; Linear ramp shapes the burst in a linear fashion. The length (in microseconds) of the power up/down ramp is set to 2 μsec when PwrRamp is not None.
- IdleInterval sets an idle duration time between two consecutive bursts when generating the 802.11b signal source.
- FilterType can be used to specify that a baseband filter is applied to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not indicate the type of filter to be used, but the transmitted signal must meet the spectral mask requirements. Four options for baseband filtering are available:
    - None (no filter)
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine how much of the ISI can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian-shaped in time and frequency domains, and it does not ring as root cosine filters do.
      The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root Cosine (also referred to as square root raised-cosine) These filters have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one.
      Root cosine filters heavily filter the signal without blurring the symbols together at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI exists at all times except at the symbol (decision) times.
    - Ideal Low Pass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal*. The resulting ripple in the cut-off band is effectively minimized with a Hamming

window. A symbol length of 32 or greater is recommended for this filter.

- RRC_Alpha is used to set the sharpness of a root cosine filter when FilterType=Root Cosine.
- GaussianFilter_bT is the Gaussian filter coefficient. B is the 3 dB bandwidth of the filter; T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.
- FilterLength is used to set the number of symbol periods to be used in the calculation of the symbol.
- For DataType:
  - if PN9 is selected, a 511-bit pseudo-random test pattern is generated according to CCITT Recommendation O.153
  - if PN15 is selected, a 32767-bit pseudo-random test pattern is generated according to CCITT Recommendation O.151
  - if FIX4 is selected, a zero-stream is generated
  - if x_1_x_0 is selected, where x equals 4, 8, 16, 32, or 64, a periodic bit stream is generated, with the period being 2 x. In one period, the first x bits are 1s and the second x bits are 0s.
- DataLength is used to set the number of data bytes in a frame.

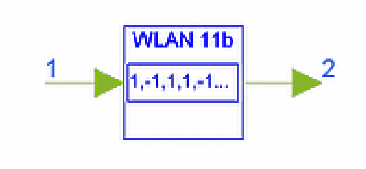
### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

   http://standards.ieee.org/getieee802/download/802.11b-1999.pdf
2. IEEE P802.11g/D8.2, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band," April 2003.

   http://standards.ieee.org/getieee802/download/802.11g-2003.pdf
3. CCITT, Recommendation O.151(10/92).
4. CCITT, Recommendation O.153(10/92).

# WLAN_Barker



**Description Barker spreader**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_Barker**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signals to be spread | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signals after spreading | complex |

## Notes/Equations

1. This component spreads the input symbols.
2. The 11-chip Barker sequence is used as the PN code sequence for the 1 and 2 Mbit/s modulation:

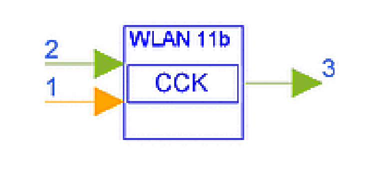
   +1, -1, +1, +1, -1, +1, +1, +1, -1, -1, -1

   The left-most chip is output first in time. The first chip is aligned at the start of a transmitted symbol. The symbol duration will be exactly 11 chips long.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_CCKMod



**Description CCK modulator**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_CCKMod**

### Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_11 | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input bits to be CCK modulated | int |
| 2 | InitialPhase | initial phase | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | output signals after modulation and spreading | complex |

### Notes/Equations

1. This component is used as a CCK modulator.
   For CCK modulation modes, the spreading code length is 8 and is based on complementary codes. The chip rate is 11 Mchip/s. The symbol duration is exactly 8 complex chips. The following formula is used to derive the CCK code words that are used for spreading 5.5 and 11 Mbit/s:

$$c = \{e^{j(\varphi 1+\varphi 2+\varphi 3+\varphi 4)}, e^{j(\varphi 1+\varphi 3+\varphi 4)}, e^{j(\varphi 1+\varphi 2+\varphi 4)}, -e^{j(\varphi 1+\varphi 4)}, e^{j(\varphi 1+\varphi 3+\varphi 3)}, e^{j(\varphi 1+\varphi 3)}, -e^{j(\varphi 1+\varphi 2)}, e^{j\varphi 1}\}$$

   where C is the code word C = {c0 to c7}.
   φ1, φ2, φ3, and φ4 are defined differently for 5.5 Mbit/s and 11 Mbit/s.
   This formula creates 8 complex chips (c0 to c7), where c0 is transmitted first in time.
   This is a form of the generalized Hadamard transform encoding, where φ1 is added to all code chips, φ2 is added to all odd code chips, φ φ3 is added to all odd pairs of code chips, and φ4 is added to all odd quads of code chips.
   φ1 modifies the phase of all code chips of the sequence and is DQPSK encoded for 5.5 and 11 Mbit/s. This takes the form of rotating the whole symbol by the appropriate amount relative to the phase of the preceding symbol. Note that chip c7 of the symbol defined above is the chip that indicates the symbol phase and is

transmitted last.

### CCK 5.5 Mbit/s Modulation

At 5.5 Mbit/s 4 bits (d0 to d3; d0 first in time) are transmitted per symbol.

Data bits d0 and d1 encode φ1 based on DQPSK. DQPSK encoding is given in the following table ( $j\omega$ is defined as counterclockwise rotation). The phase change for φ1 is relative to the phase φ1 of the preceding symbol. For the header to PSDU transition, the phase change for φ1 is relative to the phase of the preceding DQPSK (2 Mbit/s) symbol. That is, the phase of the last symbol of the CRC-16 is the reference phase for the first symbol generated from the PSDU octets.

All odd-numbered symbols generated from the PSDU octets are given an extra 180-degree rotation (in addition to the standard DQPSK modulation in the following table). The PSDU symbols are numbered starting with 0 for the first symbol to determine odd and even symbols (that is, the PSDU transmission starts on an even-numbered symbol).

**DQPSK Encoding**

| **Dibit pattern (d0, d1) (d0 is first in time)** | **Even symbols phase change ( $j\omega$ )** | **Odd symbols phase change ( $j\omega$ )** |
|---|---|---|
| 00 | 0 | π |
| 01 | π/2 | 3π/2 (−π/2) |
| 11 | π | 0 |
| 10 | 3π/2 (−π/2) | π/2 |

Data dibits d2 and d3 CCK encode the basic symbol according to the following table that is derived from the above formula by setting φ2 = (d2 × π) + π/2, φ3 = 0, and φ4 = (d3 × π). d2 and d3 are in the order shown, and the complex chips are shown c0 to c7 (left to right), with c0 transmitted first in time.

**5.5Mbit/s CCK Encoding**

| **d2, d3** | **c1** | **c2** | **c3** | **c4** | **c5** | **c6** | **c7** | **c8** |
|---|---|---|---|---|---|---|---|---|
| 00 | 1j | 1j | 1j | -1j | 1j | 1j | -1j | 1j |
| 01 | -1j | -1j | -1j | 1j | 1j | 1j | -1j | 1j |
| 10 | -1j | 1j | -1j | -1j | -1j | 1j | 1j | 1j |
| 11 | 1j | -1j | 1j | 1j | -1j | 1j | 1j | 1j |

### CCK 11 Mbit/s Modulation

At 11 Mbit/s, 8 bits (d0 to d7; d0 first in time) are transmitted per symbol.

The first dibit (d0, d1) encodes φ1 based on DQPSK. DQPSK encoding is given in the following table. The phase change for φ1 is relative to the phase φ1 of the preceding symbol. In the case of header to PSDU transition, the phase change for φ1 is relative to the phase of the preceding DQPSK symbol. All odd-numbered symbols of the PSDU are given an extra 180-degree rotation, in accordance with the DQPSK modulation shown in the following table. Symbol numbering starts with 0 for the first symbol of the PSDU.

Data dibits (d2, d3), (d4, d5), and (d6, d7) encode φ2, φ3, and φ4, respectively, based on QPSK as given in the following table.

Note that the following table is binary (not Grey) coded.

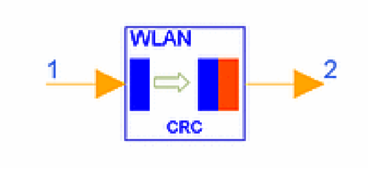
**QPSK Encoding**

| **Dibit pattern [di,d(i+1)] (di is first in time)** | **Phase** |
|---|---|
| 00 | 0 |
| 01 | п/2 |
| 10 | п |
| 11 | 3п/2(−п/2) |

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_CRC



**Description CRC calculation**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_CRC**

## Pin Inputs

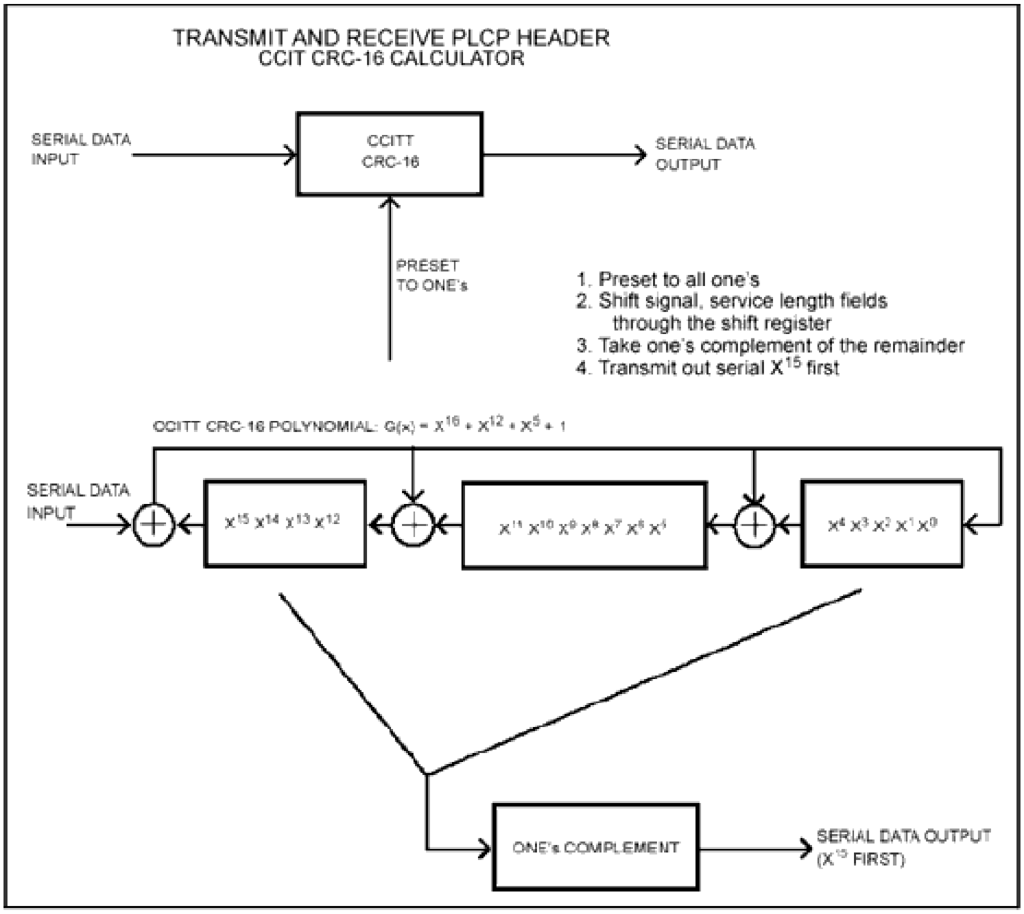
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be crc protected | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | PLCP header | int |

## Notes/Equations

1. The SIGNAL, SERVICE, and LENGTH fields are protected with a CCITT CRC-16 frame check sequence (FCS). The CCITT CRC-16 FCS is the one's complement of the remainder generated by the modulo 2 division of the protected PLCP fields by the polynomial $X^{16} + X^{12} + X^{5} + 1$.
2. Protected bits are processed in transmit order and FCS calculations are made prior to data scrambling as illustrated in the following figure.

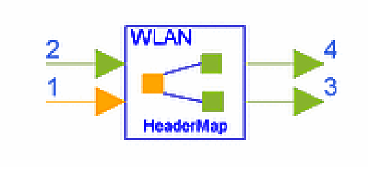


**CCITT CRC-16 Implementation**

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_HeaderMap



**Description Header mapper**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_HeaderMap**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | PLCP header signal | int |
| 2 | inphase | initial signal from preamble mapper | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | next | last complex mapping signal | complex |
| 4 | output | complex mapping signal | complex |

## Notes/Equations

1. This model outputs the PLCP header mapping signal and phase. The long PLCP preamble uses 1 Mbit/s DBPSK modulation; the short PLCP preamble uses 2 Mbit/s DQPSK modulation.
2. The initial phase of DBPSK or DQPSK signal is given by pin inphase. DBPSK encoding is given in the following table.

### DBPSK Encoding

| Bit input | Phase change (+jw) |
|---|---|
| 0 | 0 |
| 1 | π |

DQPSK encoding is given in the following table.

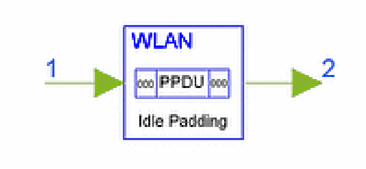
### DQPSK Encoding

| Dibit pattern (d0,d1) | Phase change (+jw) |
|---|---|
| 00 | 0 |
| 01 | π/2 |
| 11 | π |
| 10 | 3 × π/2 (−π/2) |

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_IdlePadding



**Description idle padding**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_IdlePadding**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| RampTime | power on and off ramp time | 2.0usec | sec | real | [0usec, 1000usec] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |
| OverSampling | sampling rate of pulse-shaping filter: Ratio_2, Ratio_3, Ratio_4, Ratio_5, Ratio_6, Ratio_7, Ratio_8, Ratio_9 | Ratio_2 | | enum | |
| IdleInterval | idle time | 50.0usec | sec | real | [0usec, 1000usec] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | DataIn | input data | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | DataOut | output data | complex |

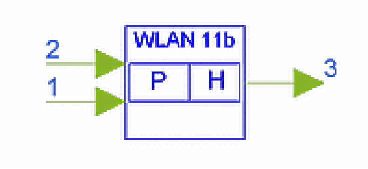
## Notes/Equations

1. This model is used to append the idle time between the consecutive packet (PPDUs).

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_MuxPLCP



**Description PLCP multiplexer**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_MuxPLCP**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum |

## Pin Inputs

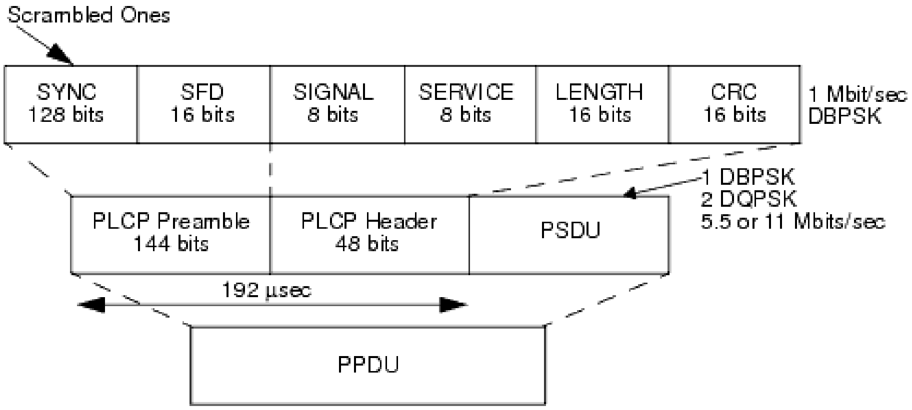
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | header | PLCP header | complex |
| 2 | preamble | PLCP preamble | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | PLCP | output PLCP | complex |

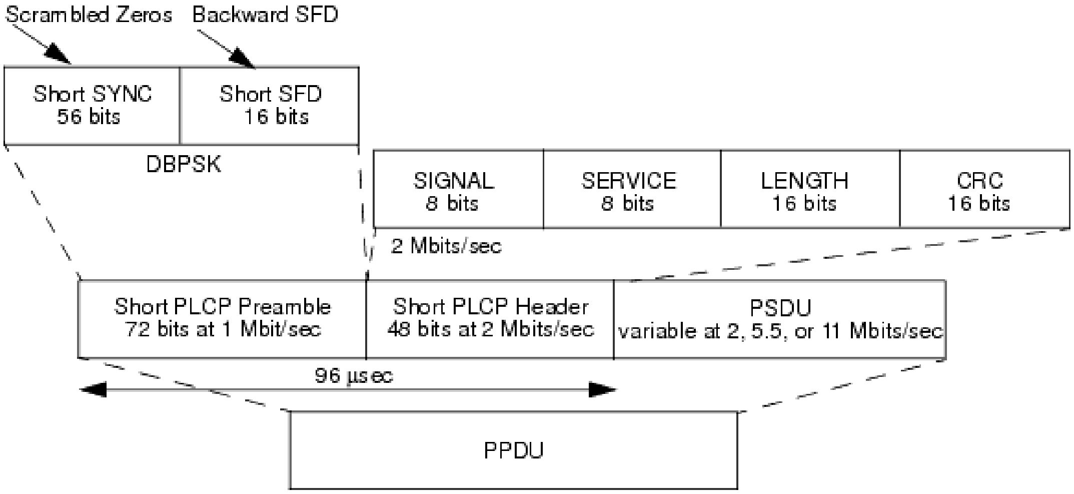
## Notes/Equations

1. This component is used to multiplex the PLCP preamble and header.
2. Two different preambles and headers are defined:
    - The mandatory supported long preamble and header which is interoperable with the current 1 Mbit/s and 2 Mbit/s DSSS specification (described in IEEE Standard 802.11, 1999 Edition) is illustrated in the following figure.



**Long PLCP PPDU Format**

- An optional short preamble and header illustrated in the following figure.



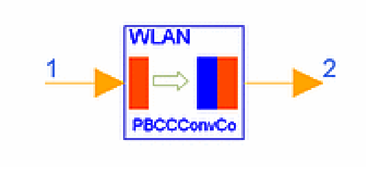
**Short PLCP PPDU Format**

The long PLCP preamble and header and short PLCP preamble use the 1 Mbit/s Barker code spreading with DBPSK modulation. The short PLCP header uses the 2 Mbit/s Barker code spreading with DQPSK modulation.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_PBCCConvCoder



**Description PBCC convolutional encoder**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PBCCConvCoder**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be convolutional coded | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after convolutional coded | int |

## Notes/Equations

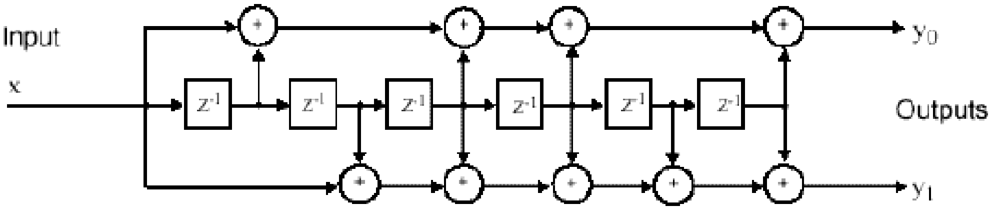
1. This model implements a PBCC convolutional encoder with appending 1 octet zeros. The DSSS/PBCC scheme uses a 64-state binary convolutional code (BCC). The output of the BCC is encoded jointly onto the I and Q channels.
2. A 64-state, rate s binary convolutional code is used. The generator matrix for the code is given as

$$G = [D^6 + D^4 + D^3 + D + 1, D^6 + D^5 + D^4 + D^3 + D^2 + 1];$$

in octal notation, it is given by

$$G = [133, 175].$$

The encoder block diagram, the following figure, consists of 6 memory elements. For every data bit input, 2 output bits are generated.



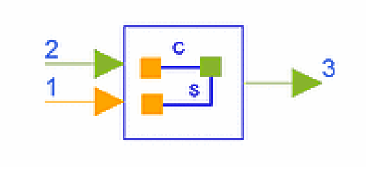
**PBCC Convolutional Encoder**

3. Since the system is frame (PPDU) based, the encoder must be in state zero (i.e., all memory elements contain zero at the beginning of each PPDU). The encoder must also be placed in a known state at the end of each PPDU to prevent the data bits near the end of the PPDU from being substantially less reliable than those early on in the PPDU. To place the encoder in a known state at the end of a PPDU, at least six deterministic bits must be input immediately following the last data bit input to the convolutional encoder. This is achieved by appending one octet containing all zeros to the end of the PPDU prior to transmission, and discarding the final octet of each received PPDU. In this manner, the decoding process can be completed reliably on the last data bits.

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_PBCCMod



**Description PBCC modulator**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PBCCMod**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_5.5, Mbps_11 | Mbps_5.5 | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Inputs

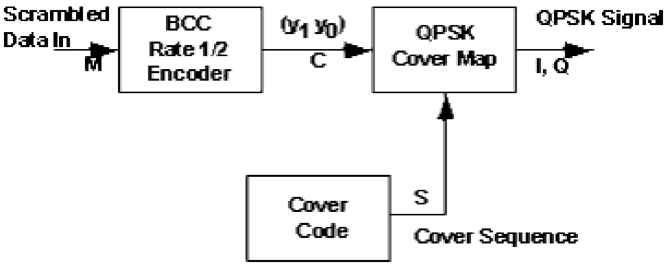
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | DataIn | the output of the PBCC binary convolution encoder | int |
| 2 | InitialPhase | initial phase | complex |

## Pin Outputs

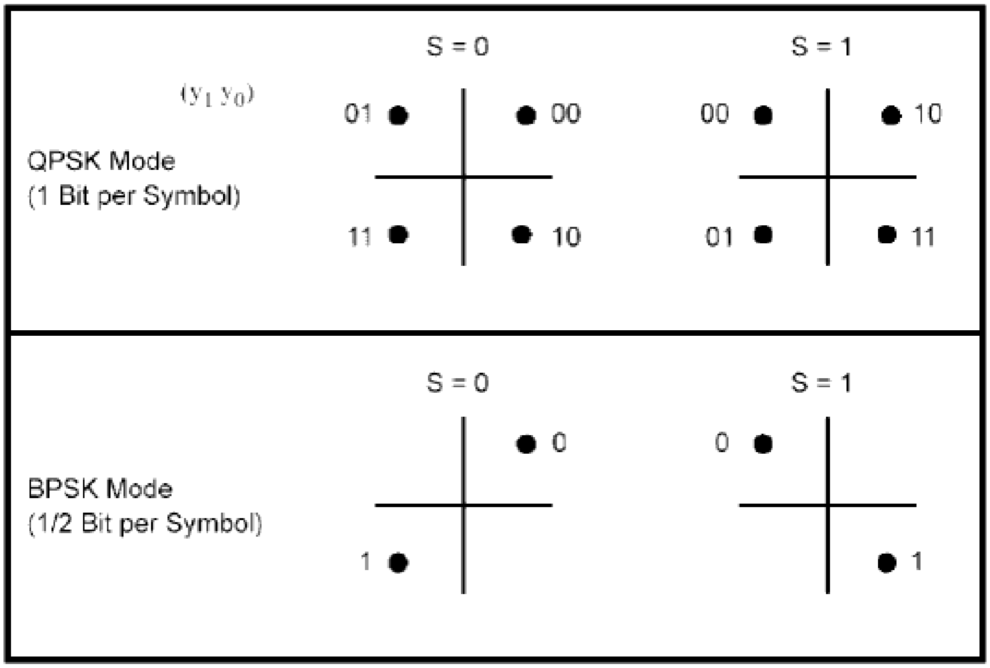
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | DataOut | the output of the PBCC modulator | complex |

## Notes/Equations

1. This model is used to map the output of the PBCC binary convolution encoder to a constellation using BPSK (for 5.5 Mbps) or QPSK (for 11 Mbps) modes.
   - For 5.5 Mbps, each firing, one token is consumed at InitialPhase, int(Octets × 8 × 2 tokens are consumed at DataIn and the same number of tokens is produced at DataOut.
   - For 11 Mbps, each firing, one token is consumed at InitialPhase, int(Octets) × 8 × 2 tokens are consumed at DataIn and int(Octets) × 8 tokens are produced at DataOut.
2. PBCC modulation is illustrated in the following figure.



3. Mapping from BCC outputs to PSK constellation points in BPSK and QPSK modes that are determined by a pseudo-random cover sequence illustrated in the following figure. (For more details regarding the pseudo-random cover sequence refer to [1].)



**PBCC Modulation**

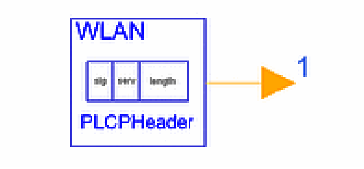
4. The InitialPhase input pin indicates the phase of the last chip of the PLCP header (that is, the last chip of the CRC check).

**Cover Code Mapping**

### References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_PLCPHeader



**Description 11b PLCP Header without CRC**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PLCPHeader**

## Parameters

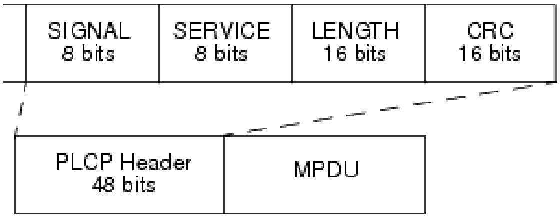
| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | enum | |
| ModType | modulation type: CCK, PBCC | CCK | enum | |
| ClocksBit | locked clocks bit: Not, Locked | Locked | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | PLCP header signal without CRC | int |

## Notes/Equations

1. This model outputs the PLCP header signal.
2. The header format is illustrated in the following figure.



**PLCP Header Format**

The 8-bit SIGNAL field indicates to the PHY the modulation for PSDU transmission and MPDU reception. The data rate is equal to the SIGNAL field value multiplied by 100 kbit/s. The high rate PHY supports four mandatory rates given by 8-bit words, which represent the rate in units of 100 kbit/s, where the lsb is transmitted in time:

- X'0A' (msb to lsb) for 1 Mbit/s
- X'14' (msb to lsb) for 2 Mbit/s
- X'37' (msb to lsb) for 5.5 Mbit/s
- X'6E' (msb to lsb) for 11 Mbit/s

Three bits are defined in the SERVICE field to support the high rate extension.

- The right-most bit 7 supplements the LENGTH field described in the PLCP Length field.
- Bit 3 indicates the modulation method: 0 = CCK; 1 = PBCC
- Bit 2 indicates that the transmit frequency and symbol clocks are derived from the same oscillator. This locked clocks bit is set by the PHY layer configuration.

The SERVICE field transmits b0 first in time, and is protected by the CCITT CRC-16 frame check sequence described in the PLCP CRC field. An IEEE 802.11-compliant device sets the values of bits b0, b1, b4, b5, and b6 to 0.

| **b0** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** | **b7** |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Locked clocks bit: 0 = not<br>1 = locked | Mod. selection bit:<br>0 = CCK<br>1 = PBCC | Reserved | Reserved | Reserved | Length extension bit |

The PLCP length field 16-bit integer indicates the number of microseconds required to transmit the PSDU. The transmitted value is determined by the Octets and Rate parameters.
Octets is converted to microseconds for inclusion in the PLCP Length field. The Length field is calculated as follows. Since there is an ambiguity in the number of octets that is described by a length in integer microseconds for any data rate over 8 Mbit/s, a length extension bit is placed at bit position b7 in the SERVICE field to indicate when the smaller potential number of octets is correct.

- 5.5 Mbit/s CCK Length = number of octets × 8 / 5.5, rounded up to the next integer.
- 11 Mbit/s CCK Length = number of octets × 8 / 11, rounded up to the next integer; the service field (b7) bit will indicate a 0 if rounding took less than 8/11 or a 1 if rounding took more than or equal to 8 / 11.
- 5.5 Mbit/s PBCC Length = (number of octets + 1) × 8 / 5.5, rounded up to the next integer.
- 11 Mbit/s PBCC Length = (number of octets + 1) × 8 / 11, rounded up to the next integer; the service field (b7) bit will indicate a 0 if rounding took less than 8/11, or a 1 if rounding took more than or equal to 8 / 11.

At the receiver, the number of octets in the MPDU is calculated as follows:

- 5.5 Mbit/s CCK Number of octets = Length × 5.5 / 8, rounded down to the next integer.
- 11 Mbit/s CCK Number of octets = Length × 11 / 8, rounded down to the next integer, minus 1 if the service field (b7) bit is a 1.
- 5.5 Mbit/s PBCC Number of octets = (Length × 5.5 / 8) - 1, rounded down to the next integer.
- 11 Mbit/s PBCC Number of octets = (Length × 11 / 8) - 1, rounded down to the next integer, minus 1 if the service field (b7) bit is a 1.

At the transmitter, LENGTH field and length extension bit values are calculated:

LENGTH'x = ((number of octets + P) × 8) / R
LENGTH = Ceiling (LENGTH'x)

If ((R == 11) && ((LENGTH - LENGTH'x) >= 8 / 11))

then Length Extension = 1
else Length Extension = 0
At the receiver, the number of octets in the MPDU is calculated:

Number of octets = Floor (Length × R / 8 - P - Length Extension)

where

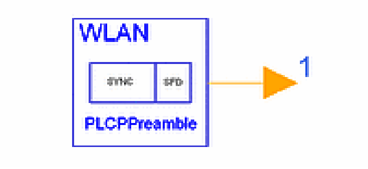
R is the data rate in Mbit/s
P = 0 for CCK; 1 for PBCC
Ceiling (X) returns the smallest integer value greater than or equal to X
Floor (X) returns the largest integer value less than or equal to X.

The least significant bit (lsb) are transmitted first in time.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_PLCPPreamble



**Description 11b PLCP preamble**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PLCPPreamble**

## Parameters

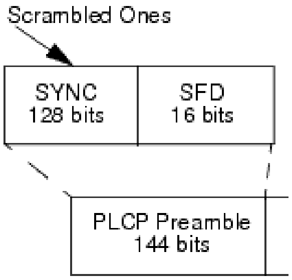
| Name | Description | Default | Type |
|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | PLCP preamble signal | int |

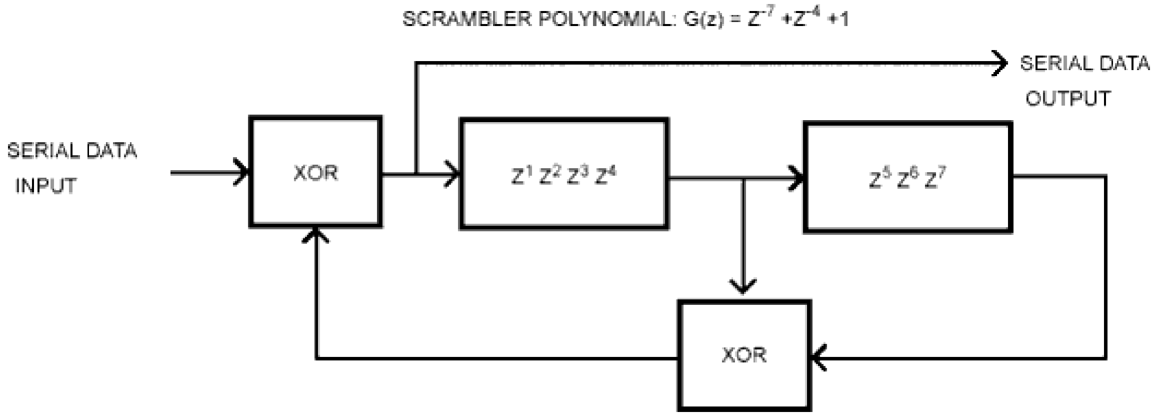
## Notes/Equations

1. This model outputs the PLCP preamble signal according to the PLCPType parameter. When PLCPType is set to *Long* , the long high rate PLCP preamble is output. The format is illustrated in the following figure.



**Long PLCP Preamble Format**

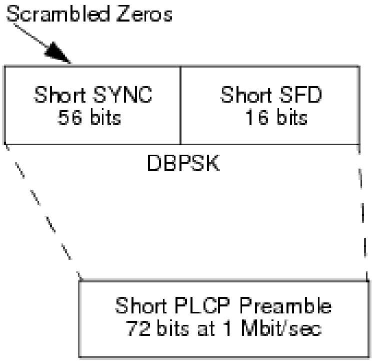
The SYNC field, which consists of 128 bits of scrambled 1 bits, is provided for receiver synchronization. The initial state of the scrambler (seed) is [1101100], where the left-most bit specifies the value for the first delay element Z $^1$ (see the following figure) and the right-most bit specifies the value of the last delay element in the scrambler.



**Data Scrambler**

The SFD indicates the start of PHY-dependent parameters within the PLCP preamble. The SFD is a 16-bit field, [1111 0011 1010 0000], where the right-most bit is transmitted first.

When PLCPType is set to *Short*, the short high rate PLCP preamble (illustrated in the following figure) is output.



**Short PLCP Preamble Format**

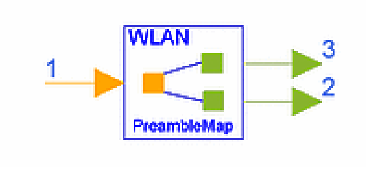
The short SYNC field, which consists of 56 bits of scrambled 0 bits, is provided for receiver synchronization. The initial state of the scrambler (seed) is [001 1011], where the left end bit specifies the value for the first delay element Z $^{1}$ and the right end bit specifies the value to put in the last delay element Z $^{7}$ .

The short SFD will be a 16-bit field and will be the time reverse of the field of the SFD in the long PLCP preamble. The field is the bit pattern 0000 0101 1100 1111. The right end bit will be transmitted first in time.

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.

# WLAN_PreambleMap



**Description Preamble mapper**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PreambleMap**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| PLCPType | PLCP preamble type: Long, Short | Long | enum | |
| InitPhase | initial phase of DBPSK | 3.1416 / 4 | real | [0, 2π ) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | preamble | PLCP preamble signal | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | next | last DBPSK signal | complex |
| 3 | output | complex DBPSK signal | complex |

## Notes/Equations

1. This model outputs the PLCP preamble mapping signal and phase. Both long PLCP and short PLCP preambles use 1 Mbit/s DBPSK modulation.
   The InitPhase parameter specifies the initial phase of the DBPSK signal. DBPSK encoding is given in the following table.

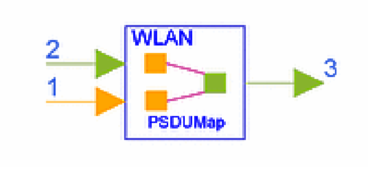
   **DBPSK Encoding**

   | Bit Input | Phase Change (+jw) |
   |---|---|
   | 0 | 0 |
   | 1 | π |

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_PSDUMap



**Description PSDU mapper**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_PSDUMap**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | data rate: Mbps_1, Mbps_2 | Mbps_1 | enum | |
| Octets | octet number of PSDU | 100 | int | (0, 2312] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU signal | int |
| 2 | inphase | initial phase from header mapper | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | complex mapping signal | complex |

## Notes/Equations

1. This model outputs a PSDU mapping signal. The basic access rate is based on 1 Mbit/s DBPSK modulation; the enhanced access rate is based on 2 Mbit/s DQPSK modulation.
   DBPSK encoding is given in the following table:

   **DBPSK Encoding**

   | Bit input | Phase change (+jw) |
   |---|---|
   | 0 | 0 |
   | 1 | π |

   DQPSK encoding is given in the following table:

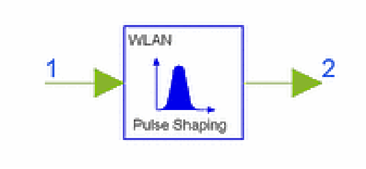
   **DQPSK Encoding**

| **Dibit pattern (d0,d1)** | **Phase change (+jw)** |
|---|---|
| 00 | 0 |
| 01 | π/2 |
| 11 | π |
| 10 | 3 × π/2 (−π/2) |

## References

1. IEEE Standard 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer Extension in the 2.4 GHz Band," 1999.
2. IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 1999.

# WLAN_TransFilter



**Description Pulse-shaping Filter**
**Library WLAN, 11b Signal Source**
**Class SDFWLAN_TransFilter**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| FilterType | pulse-shaping filter type: NoneFilter, Gaussian, Root-Cosine, Ideal-Lowpass | Gaussian | | enum | |
| Taps | number of taps | 6 | | int | [1, 1000] |
| Interpolation | interpolation ratio | 1 | | int | [1, ∞) |
| Decimation | decimation ratio | 1 | | int | [1, ∞) |
| Alpha | roll-off factor for root raised-cosine filter | 0.5 | | real | (0, 1.0] |
| BT | product of 3dB bandwidth and symbol time for Gaussian filter | 0.5 | | real | (0, 1.0] |
| Rate | data rate: Mbps_1, Mbps_2, Mbps_5.5, Mbps_11 | Mbps_5.5 | | enum | |
| ModType | modulation type: CCK, PBCC | CCK | | enum | |
| PLCPType | PLCP preamble type: Long, Short | Long | | enum | |
| Octets | octet number of PSDU | 100 | | int | (0, 2312] |
| RampTime | power on and off ramp time | 2.0μsec | sec | real | [0μsec, 1000μsec] |
| PwrType | power on and off ramp type: None, Linear, Cosine | None | | enum | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | DataIn | input data | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | DataOut | output data | complex |

## Notes/Equations

1. This model is the modulation pulse-shaping filter. Each firing, data with ramp and PPDU is input, then interpolated, filtered and decimated; Interpolation/Decimation data points are then output.
2. The impulse response of the rectangle filter can be given by

$$h(t) = \frac{\sin((\pi t)/T_s)}{(\pi t)/T_s}$$

where Ts is symbol interval.

3. The impulse response of Gaussian filter can be given by

$$g(t) = \left(erf\left(\pi \times BT\frac{Ts-2t}{Ts\sqrt{\ln(4)}}\right) + erf\left(\pi \times BT\frac{Ts+2t}{Ts\sqrt{\ln(4)}}\right)\right)$$

and

$$erf(t) = \frac{2}{\sqrt{\pi}}\int_0^t e^{-\tau^2}d\tau$$

where BT is the BT parameter and Ts is the symbol interval.

4. The impulse response of raised-cosine rolloff filter can be given by

$$h(t) = \left(\frac{\sin((\pi t)/(Ts))}{\pi t}\right)\left(\frac{\cos((\pi\alpha t)/(Ts))}{1-((4\alpha t)/(2Ts))^2}\right)$$

where α is the Alpha parameter and Ts is the symbol interval.

5. The impulse response of root raised-cosine filter has the following relationship with the raised-cosine filter

*h* ( *n* ) = *h* 1( *n* ) ƒ *h* 1( *n* )

where *h(n)* is the impulse response of raised filter and *h1(n)* is root-raised.

6. This filter efficiently implements rational sample rate changes. When the Decimation ratio is >1, the filter behaves exactly as if it were followed by a DownSample component; similarly, when the Interpolation ratio is set, the filter behaves as if it were preceded by an UpSample component. However, the implementation is much more efficient than it would be using UpSample and DownSample. Arbitrary sample-rate conversions by rational factors can be accomplished this way.
7. FilterType specifies a baseband filter that is used to reduce the transmitted bandwidth, thereby increasing spectral efficiency. The 802.11b specification does not specify what type of filter must be used, but the transmitted signal must meet the spectral mask requirements. FilterType options are:
    - NoneFilter No transmitter filter is used.
    - Gaussian The Gaussian filter does not have zero ISI. Wireless system architects must determine just how much of the inter-symbol interference can be tolerated in a system and combine that with noise and interference. The Gaussian filter is Gaussian-shaped in both the time and frequency domains; it does not ring like the root-cosine filters ring. The effects of this filter in the time domain are relatively short and each symbol interacts significantly (or causes ISI) with only the preceding and succeeding symbols. This reduces the tendency for particular sequences of symbols to interact which makes amplifiers easier to build and more efficient.
    - Root-Cosine Root-cosine filters (also referred to as square root raised-cosine filters) have the property that their impulse response rings at the symbol rate. Adjacent symbols do not interfere with each other at the symbol times because the response equals zero at all symbol times except the center (desired) one. Root-cosine filters heavily filter the signal without blurring the symbols together

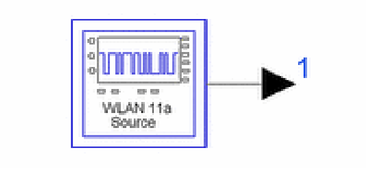
at the symbol times. This is important for transmitting information without errors caused by ISI. Note that ISI does exist at all times except at symbol (decision) times.
    - Ideal-Lowpass In the frequency domain, this filter appears as a lowpass, rectangular filter with very steep cut-off characteristics. The passband is set to equal the symbol rate of the signal. Due to a finite number of coefficients, the filter has a predefined length and is not truly *ideal* . The resulting ripple in the cut-off band is effectively minimized with a Hamming window. A symbol length of 32 or greater is recommended for this filter.

8. Taps is the filter length and determines how many symbol periods will be used in the calculation of the symbol. The filter selection influences the value of Taps.
    - The Gaussian filter has a rapidly decaying impulse response, so a filter length of 6 is recommended; greater lengths have negligible effects on the accuracy of the signal.
    - The root-cosine filter has a slowly decaying impulse response. A filter length of approximately 32 is recommended. Beyond this, the ringing has negligible effects on the accuracy of the signal.
    - The ideal lowpass filter also has a very slow decaying impulse response. A filter length of 32 or greater is recommended.
      For both root-cosine and ideal lowpass filters, the greater the filter length, the greater the accuracy of the signal.
9. Alpha is to set the sharpness of a root-cosine filter when FilterType=Root-Cosine.
10. BT is the Gaussian filter coefficient; B is the 3 dB bandwidth of the filter and T is the duration of the symbol period. BT determines the extent of the filtering of the signal. Common values for BT are 0.3 to 0.5.
11. Rate is used to determine the transmitted data rate.
12. PLCPType is used to select the format (Long or Short) of the preamble/header sections of the framed signal.
13. Octets indicates data bytes per burst (note that it is in bytes; to transform it into bits, multiply by 8).
14. RampTime specifies the length (in microseconds) of the power up/down ramp; it is used when PwrType is Linear or Cosine.
15. PwrType specifies the pattern for generating the ramp signal: None, Linear, or Cosine. The Cosine ramp gives the least amount of out-of-channel interference; None starts transmitting the signal at full power (it is the simplest power ramp to implement); and, the Linear ramp shapes the burst in a linear fashion.

# 80211a Signal Sources

- *WLAN 802 11aRF* (wlan)
- *WLAN 80211a RF* (wlan)
- *WLAN 80211a RF WithPN* (wlan)
- *WLAN 80211aSignalSrc* (wlan)
- *WLAN 80211aSignalSrc1* (wlan)
- *WLAN DATA* (wlan)
- *WLAN ExtrPSDU* (wlan)
- *WLAN LPreambleGen* (wlan)
- *WLAN PSDU* (wlan)
- *WLAN SIGNAL* (wlan)
- *WLAN SPreambleGen* (wlan)
- *WLAN Tail* (wlan)

# WLAN_802_11aRF



**Description** WLAN 802.11a signal source
**Library** WLAN, Signal Source
**Class** TSDFWLAN_802_11aRF
**Derived From** baseARFsource

## Parameters

| Name | Description | Default | Sym | Unit | Type | Range |
|---|---|---|---|---|---|---|
| ROut | Source resistance | DefaultROut | | Ohm | real | (0, ∞) |
| RTemp | Temperature | DefaultRTemp | | Celsius | real | [-273.15, ∞) |
| TStep | Expression showing how TStep is related to the other source parameters | 1/Bandwidth/ 2^OversamplingOption | | | string | |
| FCarrier | Carrier frequency: CH1_2412.0M, CH3_2422.0M, CH5_2432.0M, CH6_2437.0M, CH7_2442.0M, CH9_2452.0M, CH11_2462.0M, CH13_2472.0M, CH36_5180.0M, CH40_5200.0M, CH44_5220.0M, CH48_5240.0M, CH52_5260.0M, CH56_5280.0M, CH60_5300.0M, CH64_5320.0M, CH149_5745.0M, CH153_5765.0M, CH157_5785.0M, CH161_5805.0M | CH1_2412.0M | | Hz | real enum | (0, ∞) |
| Power | Power | 0.04 | | W | real | [0, ∞) |
| MirrorSpectrum | Mirror spectrum about carrier? NO, YES | NO | | | enum | |
| GainImbalance | Gain imbalance, Q vs I | 0.0 | | dB | real | (-∞, ∞) |
| PhaseImbalance | Phase imbalance, Q vs I | 0.0 | | deg | real | (-∞, ∞) |
| I_OriginOffset | I origin offset (percent) | 0.0 | | | real | (-∞, ∞) |
| Q_OriginOffset | Q origin offset (percent) | 0.0 | | | real | (-∞, ∞) |
| IQ_Rotation | IQ rotation | 0.0 | | deg | real | (-∞, ∞) |
| Oversampling Option | Oversampling ratio option: Option 0 for Ratio 1, Option 1 for Ratio 2, Option 2 for Ratio 4, Option 3 for Ratio 8, Option 4 for Ratio 16, Option 5 for Ratio 32 | Option 2 for Ratio 4 | S | | enum | |
| DataRate | Data rate (Mbps): Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_54 | R | | enum | |
| Bandwidth | Bandwidth | 20 MHz | B | Hz | real | (0, ∞) |
| IdleInterval | Burst idle interval | 4.0 μsec | I | sec | real | [0, 1000μsec] |
| DataType | Payload data type: PN9, PN15, FIX4, _4_1_4_0, _8_1_8_0, _16_1_16_0, _32_1_32_0, _64_1_64_0 | PN9 | | | enum | |
| DataLength | Data length (bytes per burst) | 100 | L | | int | [1, 4095] |
| GuardInterval | Guard interval (frac FFT size) | 0.25 | | | real | [0, 1] |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF | RF output | timed |

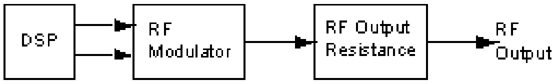
## Notes/Equations

1. This WLAN signal source generates an IEEE 802.11a and 802.11g OFDM RF signal.
   To use this source, the designer must set (as a minimum) RF carrier frequency (FCarrier) and power (Power).
   RF impairments can be introduced by setting the ROut, RTemp, MirrorSpectrum,

GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters.
802.11a/g signal characteristics can be specified by setting the OversamplingOption, DataRate, Bandwidth, IdleInterval, DataType, DataLength, and GuardInterval parameters.

> **Note**
> While WLAN_802_11a_RF generates the same 11a RF signal format as WLAN_80211aRF, their parameters are not the same.

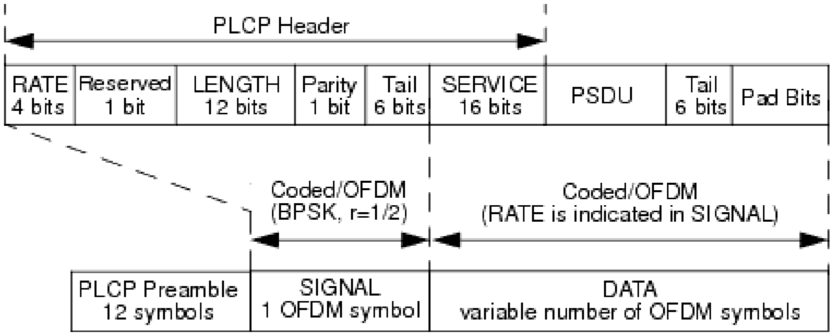
2. This signal source includes a DSP section, RF modulator, and RF output resistance as illustrated in the following figure.



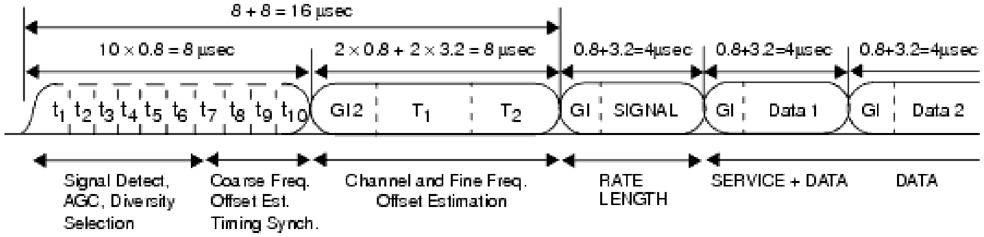
**Signal Source Block Diagram**

The ROut and RTemp parameters are used by the RF output resistance. The FCarrier, Power, MirrorSpectrum, GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters are used by the RF modulator. The remaining signal source parameters are used by the DSP block.
The RF output from the signal source is at the frequency specified (FCarrier), with the specified source resistance (ROut) and with power (Power) delivered into a matched load of resistance ROut. The RF signal has additive Gaussian noise power set by the resistor temperature (RTemp).

3. This WLAN 802.11a signal source model is compatible with the Agilent Signal Studio Software for 802.11 WLAN Agilent E4438C ESG Vector Signal Generator Option 417 for transmitter test.
Details regarding Signal Studio for WLAN 802.11 are included at the website * http://www.agilent.com/find/signalstudio]*
4. Regarding the WLAN 802.11a/g signal burst structure, one burst consists of four parts. Each burst is separated by an IdleInterval and is composed of the Short Preamble, Long Preamble, SIGNAL and DATA fields.
    - The Short Preamble field consists of 10 short preambles (8 µsec).
    - The Long Preamble field consists of 2 long preambles (8 µsec). The two preamble fields combined compose the PLCP Preamble that has a constant time duration (16 µsec) for all source parameter settings.
    - The SIGNAL field includes 802.11a/g bursts of information (such as data rate, payload data, and length).
    - The DATA field contains the payload data.
    Channel coding, interleaving, mapping and IFFT processes are also included in SIGNAL and DATA parts generation. The SIGNAL field and each individual Data field (part of the overall DATA field) have a time duration defined as the OFDM_SymbolTime and includes a GuardInterval. OFDM _SymbolTime depends on the Bandwidth (=64/Bandwidth).
    The burst structure is illustrated in the following figures. In these figures, PLCP means *physical layer convergence procedure*, PSDU means *PLCP service data units*, GI means *guard interval* ; GI is set to 0.25 and Bandwidth is set to 20 MHz (resulting in OFDM_SymbolTime = 4 µsec).



**802.11a/g Burst Format**



**OFDM Training Structure**

5. Parameter Details
   - ROut is the RF output source resistance.
   - RTemp is the RF output source resistance temperature in Celsius and sets the noise density in the RF output signal to (k(RTemp+273.15)) Watts/Hz, where k is Boltzmann's constant.
   - FCarrier is the RF output signal frequency.
   - Power is the RF output signal power. The Power of the signal is defined as the average burst power and excludes the idle interval time intervals.
   - MirrorSpectrum is used to mirror the RF_out signal spectrum about the carrier. This is equivalent to conjugating the complex RF envelope voltage. Depending on the configuration and number of mixers in an RF transmitter, the RF output signal from hardware RF generators can be inverted. If such an RF signal is desired, set this parameter to YES.
   - GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation are used to add certain impairments to the ideal output RF signal. Impairments are added in the order described here. The unimpaired RF I and Q envelope voltages have gain and phase imbalance applied. The RF is given by:

$$V_{RF}(t) = A\left(V_I(t)\cos(\omega_c t) - gV_Q(t)\sin\left(\omega_c t + \frac{\phi\pi}{180}\right)\right)$$

where *A* is a scaling factor based on the Power and ROut parameters specified by the designer, $V_I(t)$ is the in-phase RF envelope, $V_Q(t)$ is the quadrature phase RF envelope, *g* is the gain imbalance

$$g = 10^{\frac{GainImbalance}{20}}$$

and, φ (in degrees) is the phase imbalance.
Next, the signal $V_{RF}(t)$ is rotated by IQ_Rotation degrees. The I_OriginOffset

and Q_OriginOffset are then applied to the rotated signal. Note that the amounts specified are percentages with respect to the output rms voltage. The output rms voltage is given by sqrt(2 × ROut × Power).

- Bandwidth is used to determine the actual bandwidth of WLAN system and also is used to calculate the sampling rate and time step per sample. The default value is 20 MHz, which is defined in 802.11a/g specification. Bandwidth can be set to 40 MHz in order to double the rate for the 802.11a/g turbo mode.
- OversamplingOption sets the oversampling ratio of 802.11a/g RF signal source. Options from 0 to 5 result in oversampling ratio 2, 4, 8, 16, 32 where oversampling ratio = $2^{\text{OversamplingOption}}$. If OversamplingOption = 2, the oversampling ratio = $2^{2}$ = 4 and the simulation RF bandwidth is larger than the signal bandwidth by a factor of 4 (e.g. for Bandwidth=20 MHz, the simulation RF bandwidth = 20 MHz × 4 = 80 MHz).
- DataRate specifies the data rate: 6, 9, 12, 18, 24, 27, 36, 48 and 54 Mbps are available in this source. All data rates except 27 Mbps are defined in the 802.11a/g specification; 27 Mbps is from HIPERLAN/2 [2].
  The following table lists key parameters of 802.11a/g.

**Rate-Dependent Values**

| Data Rate (Mbps) | Modulation | Coding Rate (R) | Coded Bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

- IdleInterval specifies the idle interval between two consecutive bursts when generating a 802.11a signal source.
- For DataType:
  if PN9 is selected, a 511-bit pseudo-random test pattern is generated according to CCITT Recommendation O.153.
  if PN15 is selected, a 32767-bit pseudo-random test pattern is generated according to CCITT Recommendation O.151.
  if FIX4 is selected, a zero-stream is generated.
  if x_1_x_0 is selected (where x equals 4, 8, 16, 32, or 64) a periodic bit stream is generated, with the period being 2 x. In one period, the first x bits are 1s and the second x bits are 0s.
- DataLength is used to set the number of data bytes in a frame (or burst). There are 8 bits per byte.
- GuardInterval is used to set cyclic prefix in an OFDM symbol. The value range of GuardInterval is [0.0,1.0]. The cyclic prefix is a fractional ratio of the IFFT length. 802.11a/g defines GuardInterval=1/4 (0.8 μsec) and HIPERLAN/2 defines two GuardIntervals (1/8 and 1/4).

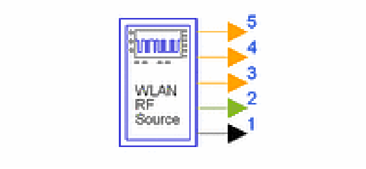
## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

   http://standards.ieee.org/getieee802/download/802.11a-1999.pdf
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November 2000.
3. IEEE P802.11G-2003, "Part II: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band," April 2003.

   http://standards.ieee.org/getieee802/download/802.11g-2003.pdf
4. CCITT, Recommendation O.151(10/92).
5. CCITT, Recommendation O.153(10/92).

# WLAN_80211a_RF



**Description** Signal source of IEEE 802.11a with RF modulation
**Library** WLAN, Signal Source
**Class** TSDFWLAN_80211a_RF
**Derived From** WLAN_SignalSourceBase

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| FCarrier | carrier frequency | 5200MHz | Hz | real | (0, ∞) |
| Power | modulator output power | 40mW | W | real | (0, ∞) |
| VRef | reference voltage for output power calibration | 0.1122V | V | real | (0, ∞) |
| Bandwidth | bandwidth | 20MHz | Hz | real | (0, ∞) |
| PhasePolarity | if set to Invert, Q channel signal is inverted: Normal, Invert | Normal | | enum | |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| I_OriginOffset | I origin offset in percent with respect to output rms voltage | 0.0 | | real | (-∞, ∞) |
| Q_OriginOffset | Q origin offset in percent with respect to output rms voltage | 0.0 | | real | (-∞, ∞) |
| IQ_Rotation | IQ rotation, in degrees | 0.0 | | real | (-∞, ∞) |
| NDensity | noise spectral density at output, in dBm/Hz | -173.975 | | real | (-∞, ∞) |
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| WindowType | type of window: Specification, CosRolloff | Specification | | enum | |
| TransitionTime | the transition time of window function | 100nsec | sec | real | (0, 800nsec] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{Order}]$ |

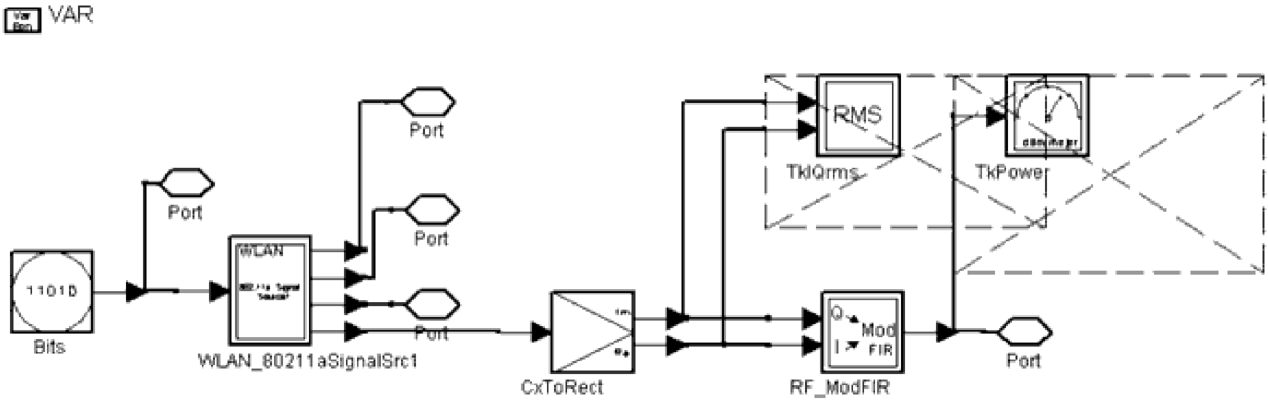
† for each array element: array size must be 7.

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |
| 2 | For_EVM | mapped SIGNAL and DATA | complex |
| 3 | EncodedBits | DATA before mapping | int |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

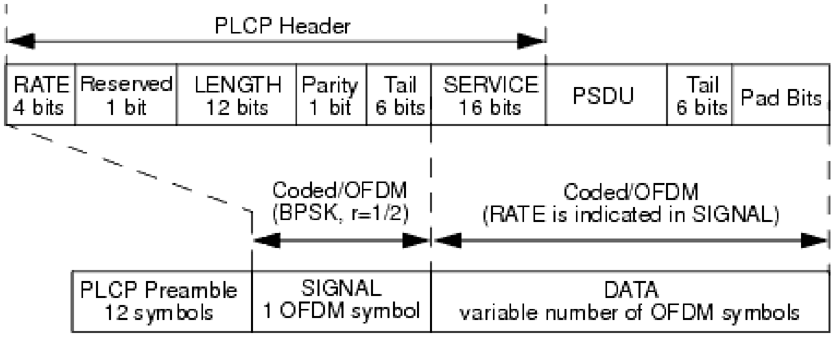
### Notes/Equations

1. This WLAN signal source generates an IEEE 802.11a and 802.11g OFDM RF signal. The generated signal can be configured in a top-level design using model parameters.
   The schematic for this subnetwork is shown in the following figure.

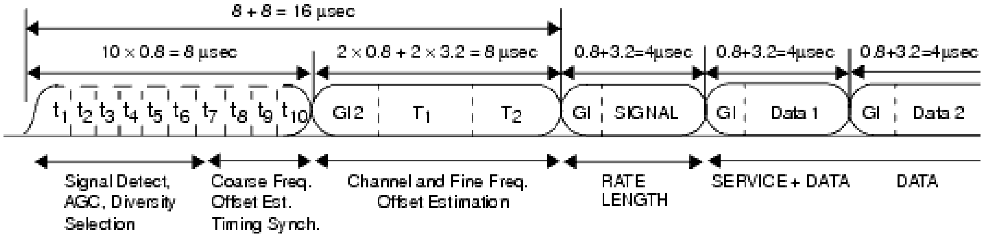


**WLAN_80211a_RF Schematic**

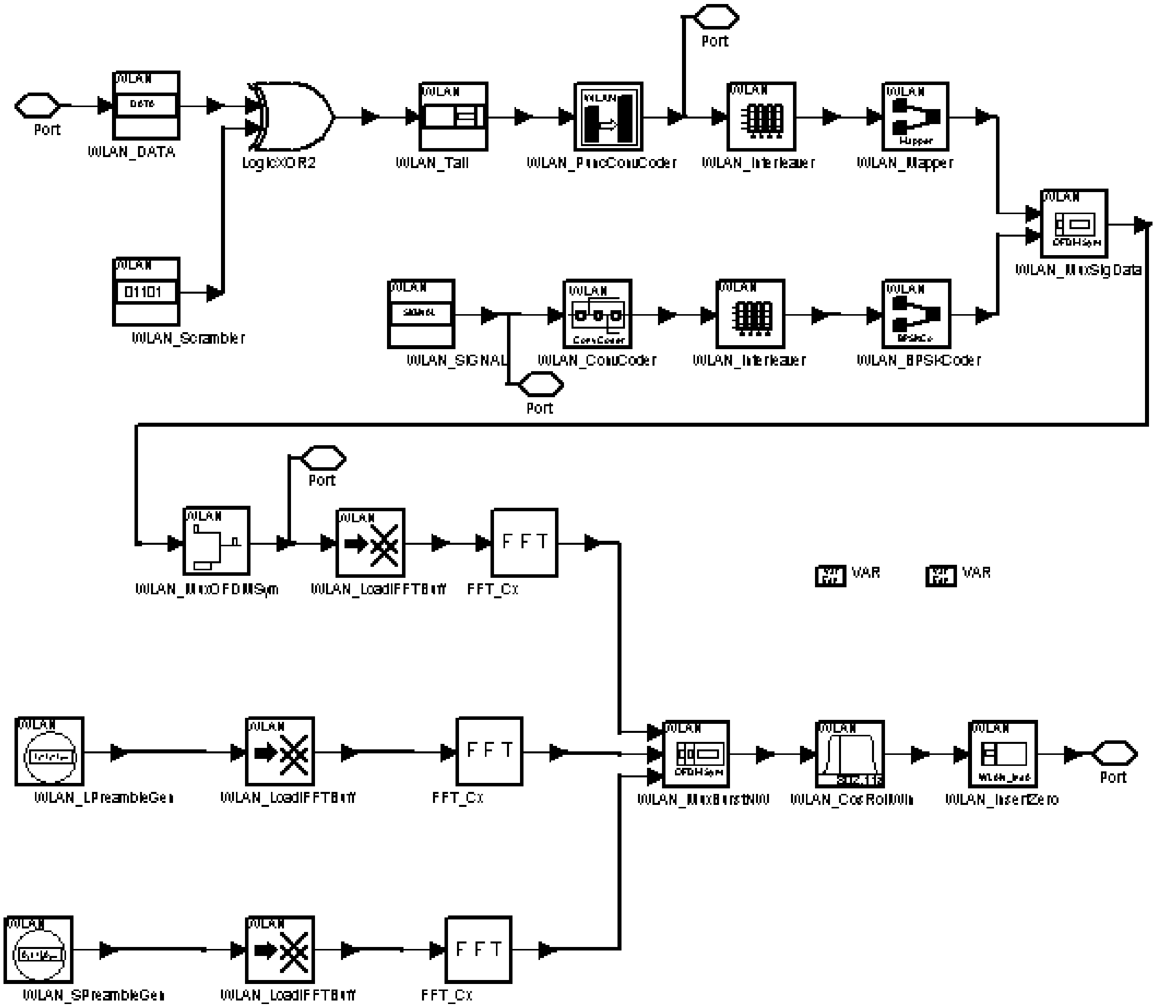
2. Model TkIQrms and TkPower are used to calibrate the output power. When these are activated and simulated, the result shown in "input IQ signal rms value" should be the VRef for WLAN_80211a_RF. If VRef is set correctly, the output power is the designated Power, as can be seen in the "Modulator output power in dBm".
3. Outputs include: RF_Signal which is the timed signal after RF modulation; For_EVM signal which is used for EVM measurement; EncodedBits which is encoded data bits before interleaving; PSDU which is the PSDU bits; and SIGNAL which is the SIGNAL bits.
4. Regarding the WLAN 802.11a/g signal burst structure, one burst consists of four parts. PLCP Preamble consists of 10 short preambles (8 usec) and 2 long preambles (8 usec). SIGNAL, includes 802.11a/g bursts of information (such as data rate, payload data, and length). DATA transmits payload data. Channel coding, interleaving, mapping and IFFT processes are also included in SIGNAL and DATA parts generation. The burst structure is illustrated in 802.11a/g Burst Format and OFDM Training Structure. The schematic of baseband 802.11a/g signal source is shown in WLAN_80211aSignalSrc1 Schematic.



**802.11a/g Burst Format**



**OFDM Training Structure**



**WLAN_80211aSignalSrc1 Schematic**

5. Parameter Details
   - The FCarrier parameter is the RF output signal frequency.
   - The Power parameter is the RF output signal power.
   - The PhasePolarity parameter is used to mirror the RF_Signal signal spectrum about the carrier. This is equivalent to conjugating the complex RF envelope voltage.
     Depending on the configuration and number of mixers in an RF transmitter, the RF output signal from hardware RF generators can be inverted. If such an RF signal is desired, set this parameter to Invert.

- The GainImbalance, PhaseImbalance, I_OriginOffset, Q_OriginOffset, and IQ_Rotation parameters are used to add certain impairments to the ideal output RF signal. Impairments are added in the order described here.
  The unimpaired RF I and Q envelope voltages have gain and phase imbalance applied. The RF is given by:
  $$V_{RF}(t) = A\left(V_I(t)\cos(\omega_c t) - gV_Q(t)\sin\left(\omega_c t + \frac{\phi\pi}{180}\right)\right)$$
  where *A* is a scaling factor based on the Power and R parameters specified by the designer, *V* I( *t* ) is the in-phase RF envelope, *V* Q( *t* ) is the quadrature phase RF envelope, *g* is the gain imbalance
  $$g = 10^{\frac{GainImbalance}{20}}$$
  and, φ (in degrees) is the phase imbalance.
  Next, the signal *V* RF( *t* ) is rotated by IQ_Rotation degrees. The I_OriginOffset and Q_OriginOffset are then applied to the rotated signal. Note that the amounts specified are percentages with respect to the output rms voltage. The output rms voltage is given by sqrt(2 × R × Power).
- Bandwidth is used to determine the actual bandwidth of WLAN system and also is used to calculate the sampling rate and time step per sample. The default value is 20 MHz, which is defined in 802.11a/g specification. Bandwidth can be set to 40 MHz in order to double the rate for the 802.11a/g turbo mode.
- Order is set to the FFT size of OFDM symbol. In fact this parameter controls the oversampling ratio of 802.11a/g RF signal source. Oversampling ratios is 1, 2, 4, 8, 16, and 32 when Order is set to 6, 7, 8, 9, 10 and 11 respectively.
- Rate specifies the data rate: 6, 9, 12, 18, 24, 27, 36, 48 and 54 Mbps are available in this source. All data rates except 27 Mbps are defined in the 802.11a/g specification; 27 Mbps is from HIPERLAN/2.
- The Idle parameter specifies padded number of zeros between two consecutive bursts when generating a 802.11a signal source. The duration of idle interval is Idle/Bandwidth.
- Length is used to set the number of data bytes in a frame (or burst).
- GuardType is used to set cyclic prefix mode in an OFDM symbol. Five modes are defined: T/2, T/8, T/16, T/32 and UserDefined. The number of cyclic prefix samples (GuardInterval parameter, set it as GI in equations) is defined as:
  if GuardType=T/32, $GI = 2^{Order-5}$
  if GuardType=T/16, $GI = 2^{Order-4}$
  if GuardType=T/8, $GI = 2^{Order-3}$
  if GuardType=T/4, $GI = 2^{Order-2}$
  if GuardType=T/2, $GI = 2^{Order-1}$
  if GuardType=UserDefined, the number of cyclic prefix samples (GI) is determined by GuardInterval.
- GuardInterval is used to set length of cyclic prefix in an OFDM symbol if GuardType=UserDefined.

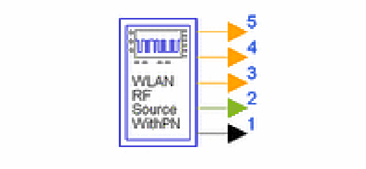
### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. IEEE P802.11g/D8.2, "Part 11: Wireless LAN Medium Access Control (MAC) and

Physical Layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band," April 2003.

3. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.

# WLAN_80211a_RF_WithPN



**Description** Signal source of IEEE 802.11a with RF modulation and phase noise
**Library** WLAN, Signal Source
**Class** TSDFWLAN_80211a_RF_WithPN
**Derived From** WLAN_SignalSourceBase

## Parameters

| **Name** | **Description** | **Default** | **Unit** | **Type** | **Range** |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp | Ohm | real | [-273.15, ∞) |
| Power | modulator output power | 40mW | W | real | (0, ∞) |
| VRef | reference voltage for output power calibration | 0.1122V | V | real | (0, ∞) |
| Bandwidth | bandwidth | 20MHz | Hz | real | (0, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 | | real | (-∞, ∞) |
| Frequency1 | first RF tone frequency | 5200MHz | Hz | real | (0, ∞) |
| Power1 | first RF tone carrier power | 0.01W | W | real | (0, ∞) |
| Phase1 | first RF tone carrier phase in degrees | 0.0 | W | real | (-∞, ∞) |
| AdditionalTones | list of additional RF tones defined with triple values for frequency in Hz, power in watts, phase in degrees | 0.0 | | real array | (0, ∞) |
| RandomPhase | set phase of RF tones to random uniformly distributed value between -PI and +PI: No, Yes: No, Yes | No | | enum | |
| PhaseNoiseData | phase noise specification defined with pairs of values for offset frequency in Hz, signal sideband pnase noise level in dBc | 0.0 | | real array | (-∞, ∞) |
| PN_Type | Phase noise model type with random or fixed offset freq spacing and amplitude: Random PN, Fixed freq offset, Fixed freq offset and amplitude: Random PN, Fixed freq offset, Fixed freq offset and amplitude | Random PN | | enum | |
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| WindowType | type of window: Specification, CosRolloff | Specification | | enum | |
| TransitionTime | the transition time of window function | 100nsec | sec | real | (0, 800nsec] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$] |

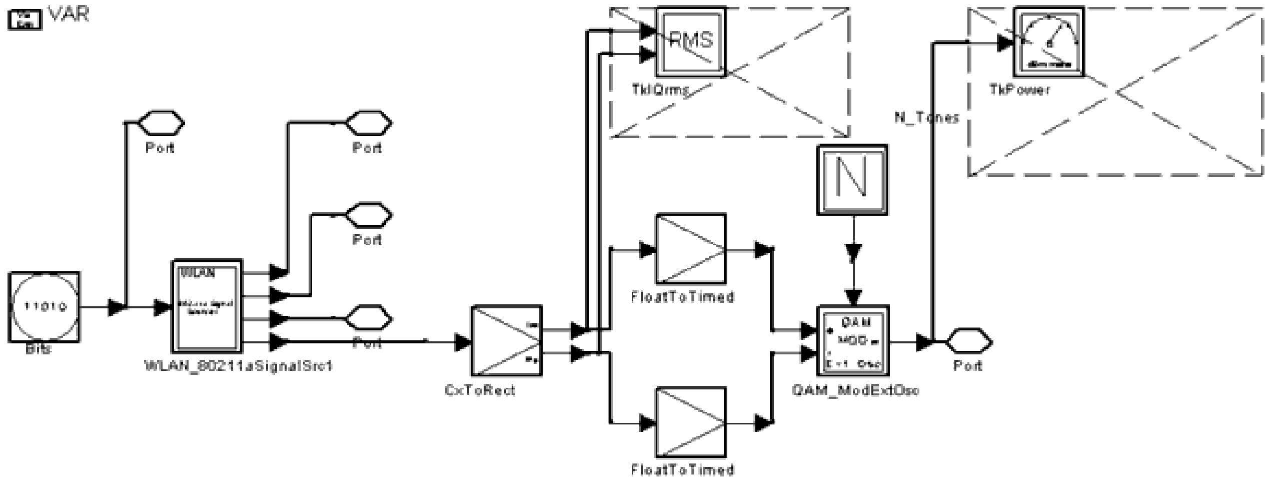
† for each array element: array size must be 7.

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | RF_Signal | RF signals | timed |
| 2 | For_EVM | mapped SIGNAL and DATA | complex |
| 3 | EncodedBits | DATA before mapping | int |
| 4 | PSDU | PSDU bits | int |
| 5 | SIGNAL | SIGNAL | int |

## Notes/Equations

1. WLAN_80211a_RF_WithPN generates a WLAN transmission signal with phase noise. The generated signal can be configured in a top-level design using model parameters.
   The schematic for this subnetwork is shown in the following figure.

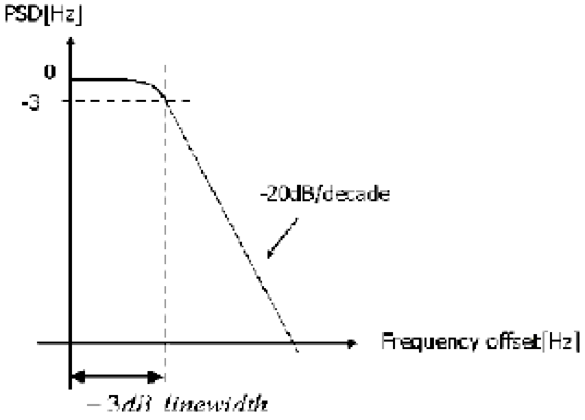


**WLAN_80211a_RF_WithPN Schematic**

2. The IEEE 802.11a baseband signal is fed into the RF modulator and phase noise is introduced by N_Tones.
3. Model TkIQrms and TkPower are used to calibrate the output power. When these are activated and simulated, the result shown in "input IQ signal rms value" should be the VRef for WLAN_80211a_RF_WithPN. If VRef is set correctly, the output power is the designated Power, as can be seen in the "Modulator output power in dBm".
4. The power density spectrum of an oscillator signal with phase noise is modeled by a Lorentzian spectrum. The single-sided spectrum S $_s$ (f) is given by

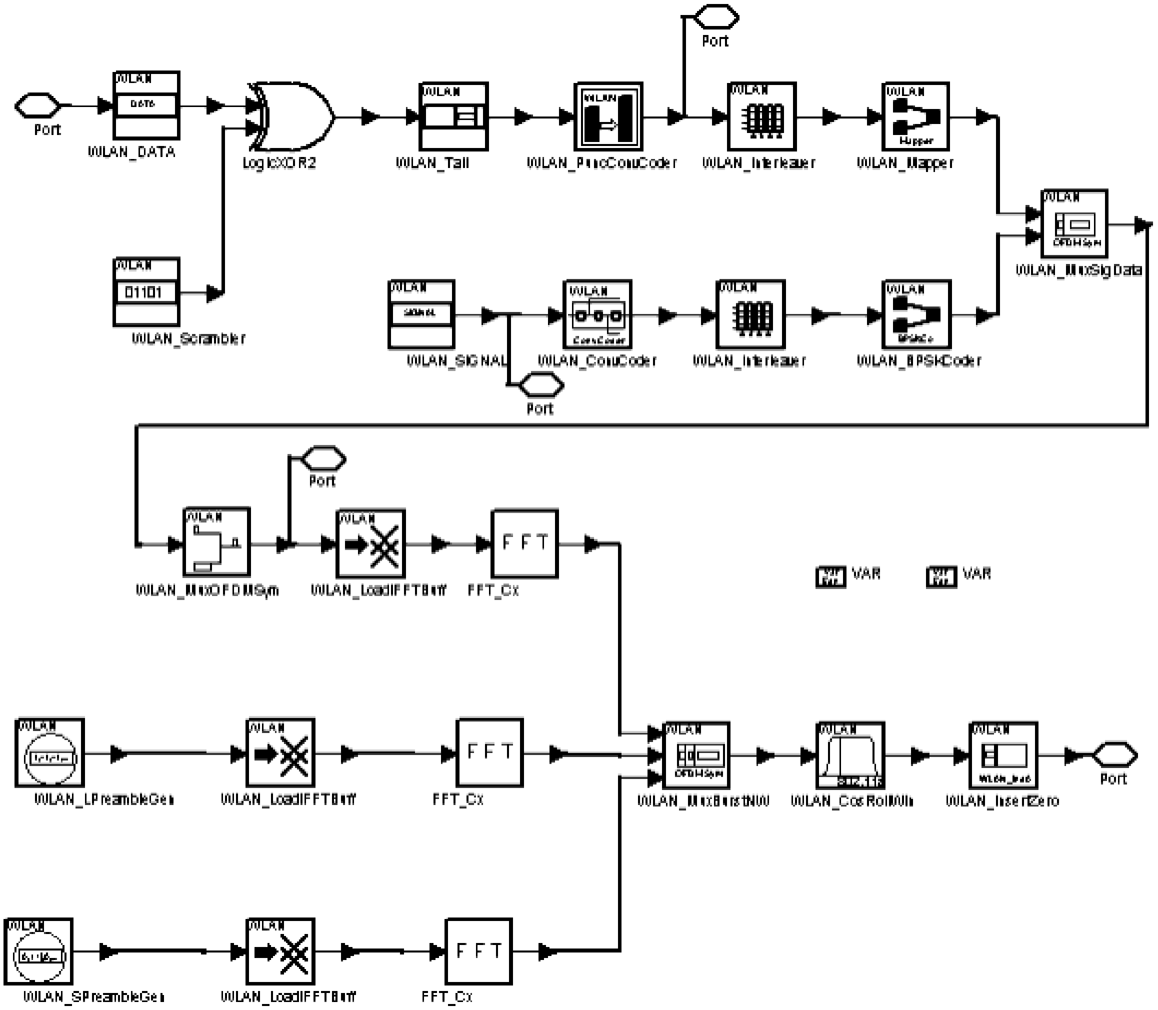
$$S_s(f) = \frac{2/(\pi f_l)}{1+f^2/f_l^2}$$

   The following figure illustrates a Lorentzian phase noise spectrum with a single-sided −3 dB line width of the oscillator signal. N_Tones models phase noise based on the Lorentzian spectrum.



**Phase Noise Power Spectral Density**

5. Outputs include: RF_Signal (timed signal after RF modulation); For_EVM signal (used for EVM measurement); EncodedBits (encoded data bits before interleaving), PSDU (PSDU bits); and SIGNAL (the SIGNAL bits).
6. WLAN_80211aSignalSrc1 implements the baseband signal source functions according to IEEE 802.11a Standard, including SIGNAL and DATA generation, scrambling, convolutional coding, interleaving, mapping, IFFT, multiplexing, adding a window function, and inserting idle. The schematic is shown in the following figure.



**WLAN_80211aSignalSrc1 Schematic**

7. Parameter Details
    - The Power parameter is the RF output signal power.
    - The GainImbalance, PhaseImbalance parameters are used to add certain impairments to the ideal output RF signal. Impairments are added in the order described here.
      The unimpaired RF I and Q envelope voltages have gain and phase imbalance applied. The RF is given by:
      $$V_{RF}(t) = A\left(V_I(t)\cos(\omega_c t) - gV_Q(t)\sin\left(\omega_c t + \frac{\phi\pi}{180}\right)\right)$$
      where *A* is a scaling factor based on the Power and R parameters specified by the designer, *V* I( *t* ) is the in-phase RF envelope, *V* Q( *t* ) is the quadrature phase RF envelope, *g* is the gain imbalance
      $$g = 10^{\frac{GainImbalance}{20}}$$
      and, φ (in degrees) is the phase imbalance.
      Note that the amounts specified are percentages with respect to the output rms voltage. The output rms voltage is given by sqrt(2 × R × Power).
    - Bandwidth is used to determine the actual bandwidth of WLAN system and also is used to calculate the sampling rate and time step per sample. The default value is 20MHz, which is defined in 802.11a/g specification. Bandwidth can be set to 40 MHz in order to double the rate for the 802.11a/g turbo mode.
    - Order is set to the FFT size of OFDM symbol. In fact this parameter controls the oversampling ratio of 802.11a/g RF signal source. Oversampling ratios is 1, 2, 4, 8, 16, and 32 when Order is set to 6, 7, 8, 9, 10 and 11 respectively.
    - Rate specifies the data rate: 6, 9, 12, 18, 24, 27, 36, 48 and 54 Mbps are available in this source. All data rates except 27 Mbps are defined in the 802.11a/g specification; 27 Mbps is from HIPERLAN/2.
    - The Idle parameter specifies padded number of zeros between two consecutive bursts when generating a 802.11a signal source. The duration of idle interval is Idle/Bandwidth.
    - Length is used to set the number of data bytes in a frame (or burst).
    - GuardType is used to set cyclic prefix mode in an OFDM symbol. Five modes are defined: T/2, T/8, T/16, T/32 and UserDefined. The number of cyclic prefix samples (GuardInterval parameter, set it as GI in equations) is defined as:
      if GuardType=T/32, $GI = 2^{Order-5}$
      if GuardType=T/16, $GI = 2^{Order-4}$
      if GuardType=T/8, $GI = 2^{Order-3}$
      if GuardType=T/4, $GI = 2^{Order-2}$
      if GuardType=T/2, $GI = 2^{Order-1}$
      if GuardType=UserDefined, the number of cyclic prefix samples (GI) is determined by GuardInterval.
    - GuardInterval is used to set length of cyclic prefix in an OFDM symbol if GuardType=UserDefined.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

2. IEEE P802.11g/D8.2, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band," April 2003.
3. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.

# WLAN_80211aSignalSrc

**Description** Signal source of IEEE 802.11a with idle
**Library** WLAN, Signal Source
**Class** SDFWLAN_80211aSignalSrc

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| WindowType | type of window: Specification, CosRolloff | Specification | | enum | |
| TransitionTime | the transition time of window function | 100nsec | sec | real | [0, 800nsec] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | $[0, 2^{\text{Order}}]$ |
| † for each array element: array size must be 7. | | | | | |

## Pin Inputs

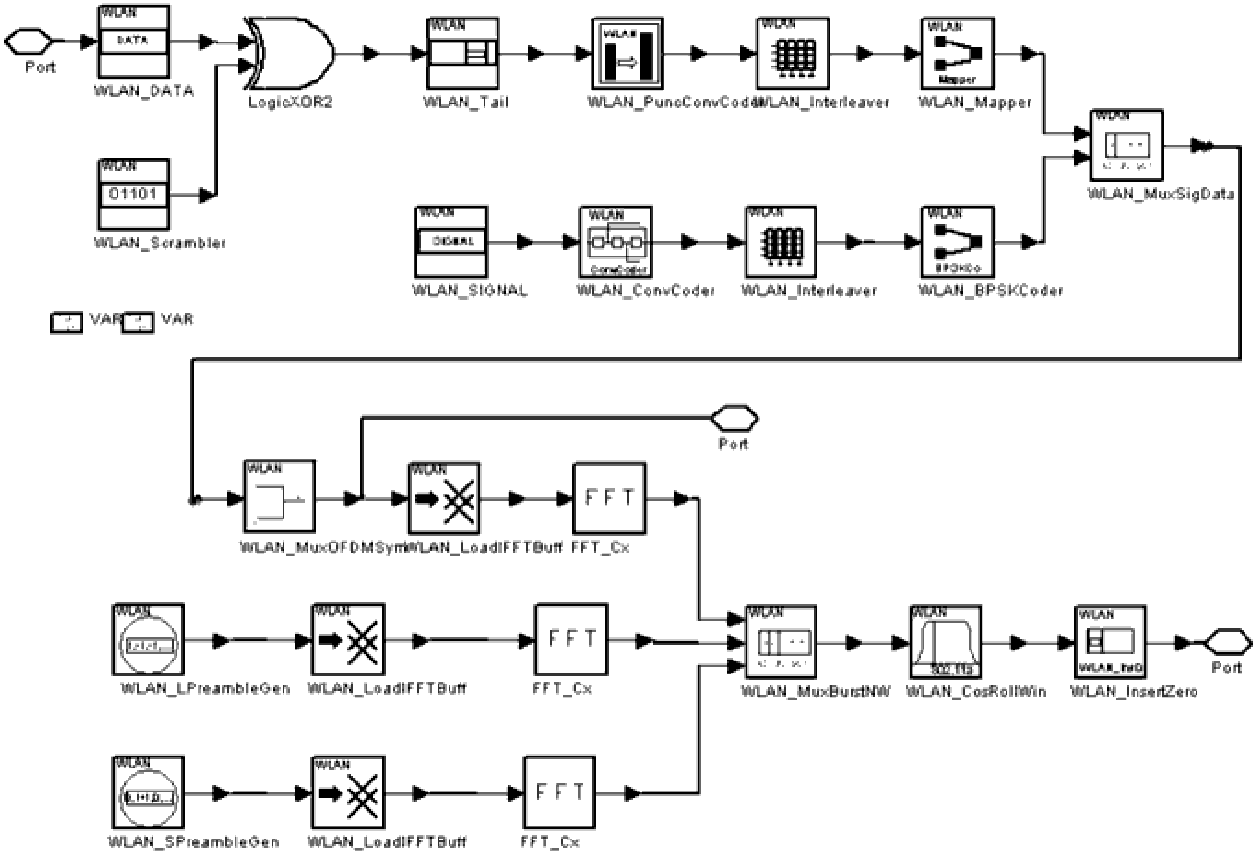
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE802.11a burst | complex |
| 3 | output | mapping signal before IFFT | complex |

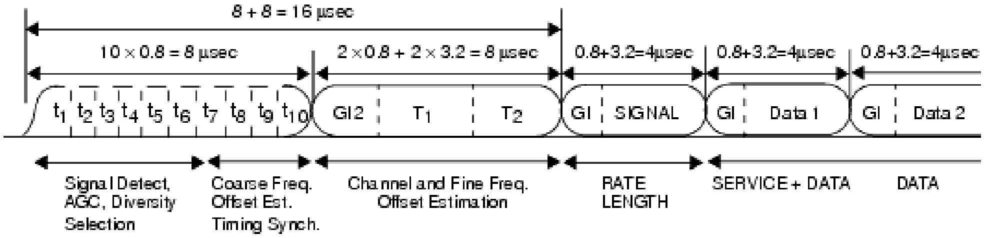
## Notes/Equations

1. This subnetwork performs IEEE 802.11a DATA convolutional coding, interleaving, mapping, IFFT, multiplexing, and adds a window. The schematic is shown in the following figure.



**WLAN_80211aSignalSrc Schematic**

2. As illustrated in the following figure, one PPDU frame includes PLCP Preamble (12 symbols: 10 short and 2 long preamble symbols), SIGNAL (one OFDM symbol) and DATA (variable number of OFDM symbols).
   Mapping modes are dependent on the Rate parameter in DATA and BPSK in SIGNAL. After mapping, DATA and SIGNAL are multiplexed, pilots are inserted, IFFT is performed, PLCP preambles are multiplexed into one PPDU frame (or burst), and the window function is added.

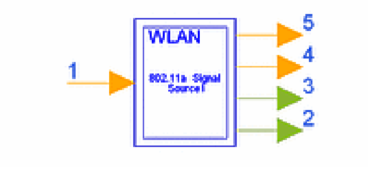


**PPDU Frame Structure**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211aSignalSrc1



**Description** Signal source of IEEE 802.11a with idle
**Library** WLAN, Signal Source
**Class** SDFWLAN_80211aSignalSrc1

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 | | int array | {0, 1}† |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| WindowType | type of window: Specification, CosRolloff | Specification | | enum | |
| TransitionTime | the transition time of window function | 100nsec | sec | real | [0, 800nsec] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{\text{Order}}$] |
| † for each array element: array size must be 7. | | | | | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

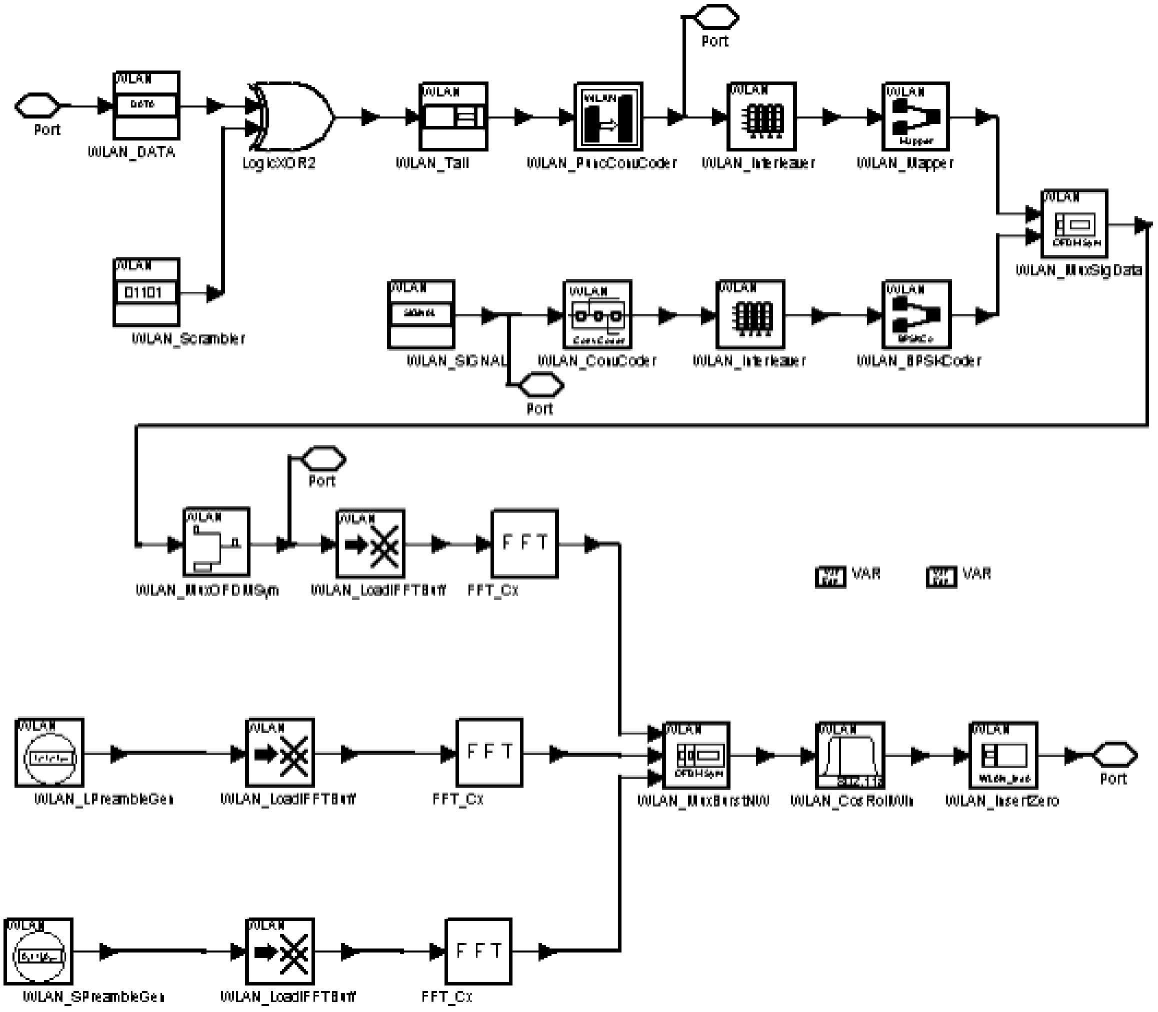
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | burst | IEEE802.11a burst | complex |
| 3 | For_EVM | mapping signal before IFFT | complex |
| 4 | EncodedBits | DATA bits before mapping | int |
| 5 | SIGNAL | SIGNAL bits | int |

## Notes/Equations

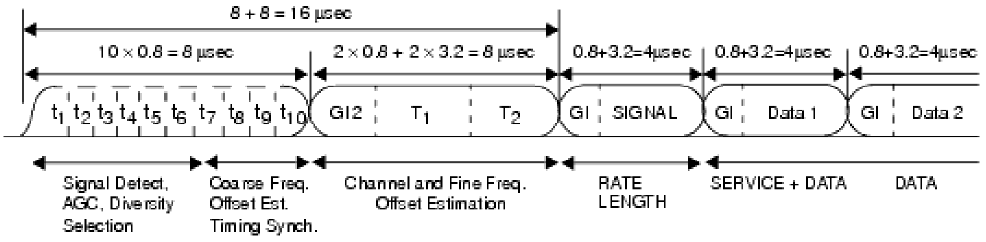
1. This subnetwork performs IEEE 802.11a DATA convolutional coding, interleaving, mapping, IFFT, and multiplexing, and adds a window function. The schematic is shown in the following figure.

Mapping modes are dependent on the Rate parameter in DATA and BPSK in SIGNAL. After mapping, DATA and SIGNAL are multiplexed, pilots are inserted, IFFT is performed, PLCP preambles are multiplexed into one PPDU frame (or burst), and a window function is added.



**WLAN_80211aSignalSrc1 Schematic**

2. One PPDU frame, as illustrated in the following figure, includes PLCP Preamble (10 short and 2 long preamble symbols), SIGNAL (one OFDM symbol) and DATA (variable number of OFDM symbols).

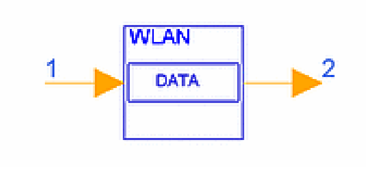


**PPDU Frame Structure**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DATA



**Description** DATA field of PPDU
**Library** WLAN, Signal Source
**Class** SDFWLAN_DATA

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

## Pin Inputs

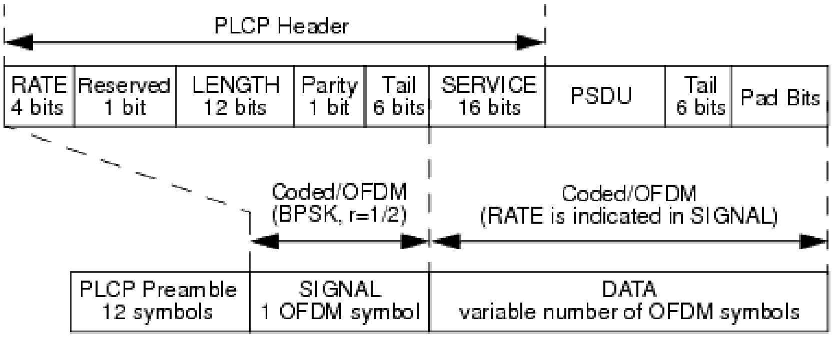
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | PSDU | PSDU bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | DATA bits output | int |

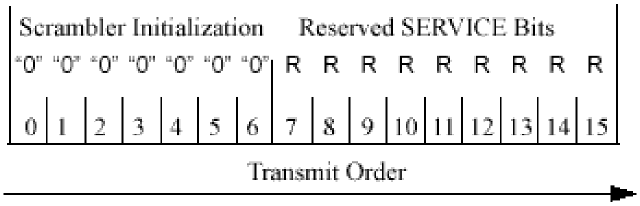
## Notes/Equations

1. This model is used to generate data field of PPDU frame. As illustrated in the following figure, the data field contains the service field, the PSDU, the tail bits, and the pad bits if needed. All bits in the data field are scrambled.



PPDU Frame Format

2. The service field (illustrated in the following figure) has 16 bits; bit 0 is transmitted first in time. Bits 0 to 6 are set to zero and used to synchronize the descrambler in the receiver. The remaining bits (7 to 15) reserved for future use are set to zero.



SERVICE Field Bit Assignments

3. The PPDU tail bit field is 6 bits of 0, which are required to return the convolutional encoder to the *zero state*. This improves the error probability of the convolutional decoder, which relies on future bits when decoding and which may be not be available past the end of the message. The PLCP tail bit field is produced by replacing 6 scrambled 0 bits following the end of message with 6 unscrambled 0 bits.
4. The number of bits in the data field is a multiple of $N_{CBPS}$, the number of coded bits in an OFDM symbol (48, 96, 192, or 288 bits). To achieve that, the length of the message is extended so that it becomes a multiple of $N_{DBPS}$, the number of data bits per OFDM symbol. At least 6 bits are appended to the message in order to accommodate the tail bits. The number of OFDM symbols $N_{SYM}$, the number of bits in the data field $N_{DATA}$, and the number of pad bits $N_{PAD}$ are calculated from the length of the PSDU (LENGTH) as follows:

$$N_{SYM} = \text{Ceiling}\,((16 + 8 \times \text{LENGTH} + 6)/ N_{DBPS})$$

$$N_{DATA} = N_{SYM} \times N_{DBPS}$$

$$N_{PAD} = N_{DATA} - (16 + 8 \times \text{LENGTH} + 6)$$

The function ceiling (.) is a function that returns the smallest integer value greater than or equal to its argument value. The appended bits ( *pad bits* ) are set to zeros and subsequently scrambled with the rest of the bits in the data field.
$N_{CBPS}$ and $N_{DBPS}$ are Rate dependent parameters listed in the following table.

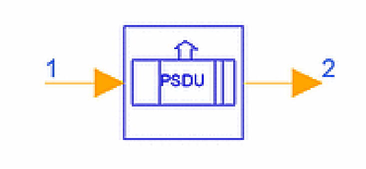
Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_ExtrPSDU



**Description** Extract PSDU from DATA
**Library** WLAN, Signal Source
**Class** SDFWLAN_ExtrPSDU

### Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

### Pin Inputs

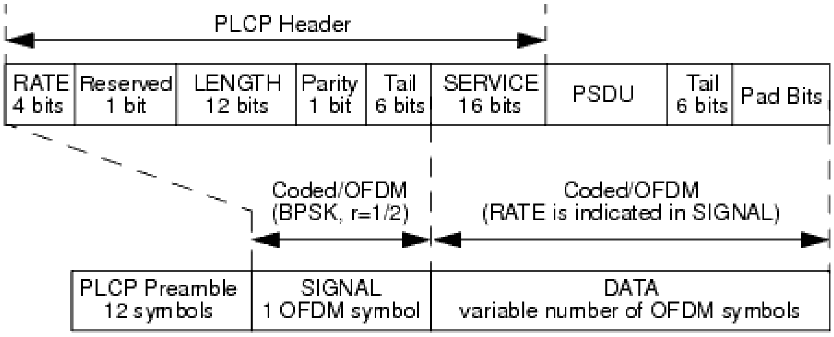
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | DATA | DATA bits | int |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | PSDU | PSDU bits | int |

### Notes/Equations

1. This model is used to extract PSDU field from data bits. Refer to the following figure.



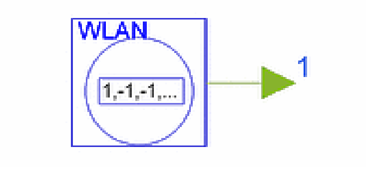
**PPDU Frame Format**

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz

Band," 1999.

# WLAN_LPreambleGen



**Description** Long training sequence generator
**Library** WLAN, Signal Source
**Class** SDFWLAN_LPreambleGen

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | 52 long training sequences | complex |

## Notes/Equations

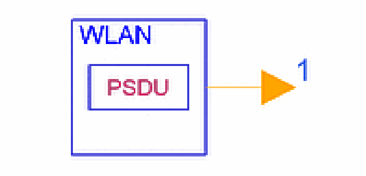
1. This model is used to generate the long training sequence in order to obtain the long OFDM training symbol. This symbol consists of 52 subcarriers, which are modulated by the elements of sequence L, given by

$$L_{0,51} = \{1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1\}$$

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_PSDU



**Description** Source of coder
**Library** WLAN, Signal Source
**Class** SDFWLAN_PSDU

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | source signal | int |

## Notes/Equations

1. This model is used to generate 100 octets of PSDU data according to the following table.

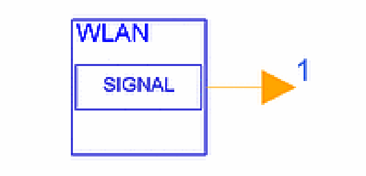
### PSDU

| ## | Value | Value | Value | Value | Value |
|---|---|---|---|---|---|
| 1...5 | 04 | 02 | 00 | 2e | 00 |
| 6...10 | 60 | 08 | cd | 37 | a6 |
| 11...15 | 00 | 20 | d6 | 01 | 3c |
| 16...20 | f1 | 00 | 60 | 08 | ad |
| 21...25 | 3b | af | 00 | 00 | 4a |
| 26...30 | 6f | 79 | 2c | 20 | 62 |
| 31...35 | 72 | 69 | 67 | 68 | 74 |
| 36...40 | 20 | 73 | 70 | 61 | 72 |
| 41...45 | 6b | 20 | 6f | 66 | 20 |
| 46...50 | 64 | 69 | 76 | 69 | 6e |
| 51...55 | 69 | 74 | 79 | 2c | 0a |
| 56...60 | 44 | 61 | 75 | 67 | 68 |
| 61...65 | 74 | 65 | 72 | 20 | 6f |
| 66...70 | 66 | 20 | 45 | 6c | 79 |
| 71...75 | 73 | 69 | 75 | 6d | 2c |
| 76...80 | 0a | 46 | 69 | 72 | 65 |
| 81...85 | 2d | 69 | 6e | 73 | 69 |
| 86...90 | 72 | 65 | 64 | 20 | 77 |
| 91...95 | 65 | 20 | 74 | 72 | 65 |
| 96...100 | 61 | da | 57 | 99 | ed |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_SIGNAL



**Description** SIGNAL field of PPDU
**Library** WLAN, Signal Source
**Class** SDFWLAN_SIGNAL

### Parameters

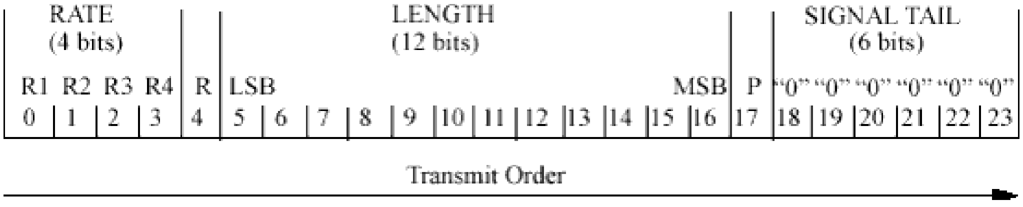
| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | SIGNAL bits output | int |

### Notes/Equations

1. The model is used to generate SIGNAL field of PPDU frame, which is composed of 24 bits, as illustrated in the following figure. Bits 0 to 3 encode the rate; bit 4 is reserved for future use; bits 5 to 16 encode the length field of the TXVECTOR, with the least significant bit (LSB) being transmitted first; bit 17 is the positive (even) parity bit for bits 0 to 16; bits 18 to 23 constitute the signal tail field and are all set to zero.



**Signal Field Bit Assignments**

2. The rate field conveys information about the type of modulation and the coding rate as used in the rest of the packet. Bits R1 to R4 are set dependent on Rate according to the values in the following table.

**Contents of Rate Field**

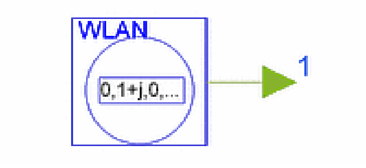
| Rate (Mbps) | R1 to R4 |
|---|---|
| 6 | 1101 |
| 9 | 1111 |
| 12 | 0101 |
| 18 | 0111 |
| 24 | 1001 |
| 27 | 1010 |
| 36 | 1011 |
| 48 | 0001 |
| 54 | 0011 |

3. The length field is an unsigned 12-bit integer that indicates the number of octets in the PSDU that the MAC is currently requesting the physical layer to transmit. The transmitted value will be in the 1 to 4095 range; the LSB will be transmitted first.
4. Encoding of the single SIGNAL OFDM symbol will be performed with BPSK modulation of the subcarriers and using convolutional coding at R = 1/2. The encoding procedure, which includes convolutional encoding, interleaving, modulation mapping processes, pilot insertion, and OFDM modulation, is as used for transmission of data at a 6 Mbps rate. Contents of the SIGNAL field are not scrambled.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_SPreambleGen



**Description** Short training sequence generator
**Library** WLAN, Signal Source
**Class** SDFWLAN_SPreambleGen

### Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| ShortType | type of short training sequence: A, B | B | enum |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | 52 short training sequences | complex |

### Notes/Equations

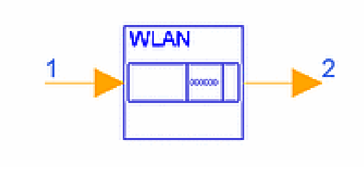
1. This model is used to generate the short training sequence in order to obtain the short OFDM training symbol; this symbol consists of 12 subcarriers that are modulated by the elements of sequence *S*.
    - If ShortType=B, the short training sequences used for IEEE 802.11a and HIPERLAN/2 standards are:
      $S_{0,51} = \sqrt{13/6}$
      · {0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0,0}
    - If ShortType=A, the short training sequences used only for HIPERLAN/2 standards are:
      $S_{0,51}$
      $= \sqrt{13/6}$ · {0, 0, 0,0,-1+j, 0, 0, 0, 1+j, 0, 0, 0, 1-j, 0, 0, 0, -1-j, 0, 0, 0, -1+j, 0, 0, 0, -1-j, 0, 0, -1+j, 0, 0, 0, -1-j, 0, 0, 0,-1+j, 0, 0, 0, -1-j, 0, 0, 0, 1-j, 0, 0, 0, 1+j, 0,0,0,0}
      The $\sqrt{13/6}$ multiplication factor normalizes the average power of the resulting OFDM symbol, which uses 12 out of 52 subcarriers.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_Tail



**Description** Attach tail bits
**Library** WLAN, Signal Source
**Class** SDFWLAN_Tail

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

## Pin Inputs

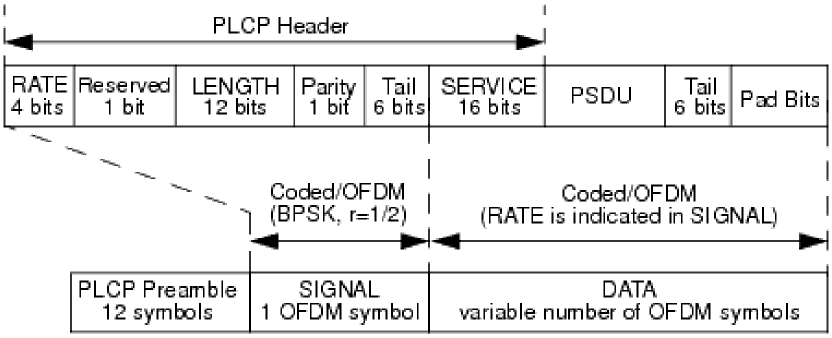
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | DATA without "zero" tail bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | DATA with six nonscrambled "zero"tail bits | int |

## Notes/Equations

1. This model is used to add six 0 tail bits to the scrambled DATA field of PPDU. The position of tail bits is illustrated in the following figure.



**PPDU Frame Format**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# About WLAN Design Library

The Agilent EEsof WLAN Design Library is for the 5 and 2.4 GHz wireless LAN market, IEEE 802.11a in the Americas, MMAC in Japan, BRAN HIPERLAN/2 in Europe, IEEE 802.11b and IEEE 802.11g. This design library focuses on the physical layer of WLAN systems and is intended to be a baseline system for designers to get an idea of what a nominal or ideal system performance would be. Evaluations can be made regarding degraded system performance due to system impairments that may include nonideal component performance.

# Agilent Instrument Compatibility

This WLAN design library is compatible with Agilent E443xB ESG-D Series Digital RF Signal Generator and Agilent E4438C ESG Vector Signal Generator.

Also, this WLAN design library is compatible with Agilent 89600 Series Vector Signal Analyzer.

The following table shows more information of instrument models, Firmware revisions, and options.

**Agilent Instrument Compatibility Information**

| WLAN Design Library | ESG Models | VSA Models |
|---|---|---|
| SpecVersion=1999 | E443xB, Firmware Revision B.03.75<br>Option 410 - "802.11a" Software Personality (Signal Studio)<br>E4438C, Firmware Revision C.02.20<br>Option 410 - "802.11a" Software Personality (Signal Studio) | 89600 Series, software version 3.01<br>Option B7R - "802.11a and HIPERLAN/2 OFDM Modulation Analysis" |

For more information about Agilent ESG Series of Digital and Analog RF Signal Generator and Options, please visit

*http://www.agilent.com/find/ESG*

For more information about Agilent 89600 Series Vector Signal Analyzer and Options, please visit

*http://www.agilent.com/find/89600*

# WLAN Systems

Three wireless LAN standards, IEEE 802.11, ETSI BRAN HIPERLAN/2, and MMAC HISWAN, are being developed. IEEE 802.11 was initiated in 1990, and several draft standards have been published for review including IEEE 802.11 and IEEE 802.11b for 2.4 GHz with 5.5 and 11 Mbps. The scope of the standard is to develop a MAC and physical layer specification for wireless connectivity for fixed, portable and moving stations within a local area.

In July 1998, the IEEE 802.11 standardization group selected OFDM as the basis for a new physical layer standard (IEEE 802.11a). This new physical layer standard has been finalized and targets 6 Mbps to 54 Mbps data rates in a 5 GHz band. A common MAC mechanism has been specified for IEEE 802.11, IEEE 802.11a, and IEEE 802.11b. The MAC mechanism provides CSMA/CA.

The HIPERLAN/2 standard is being developed in the ETSI/BRAN project. The system can operate globally in a 5 GHz band. Core specifications of the HIPERLAN/2 standard were finalized at the end of 1999. HIPERLAN/2 provides high-speed 6 Mbps to 54 Mbps wireless multimedia communications between mobile terminals and various broadband core networks.

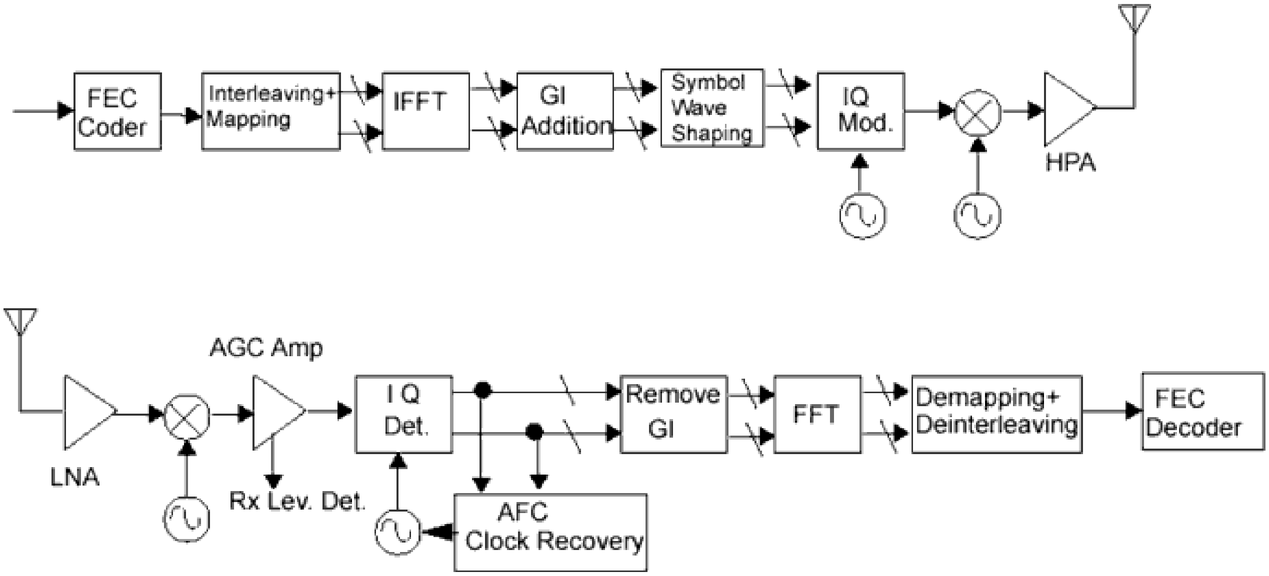
The physical layer of HIPERLAN/2 was harmonized with IEEE 802.11a. Orthogonal frequency division multiplexing (OFDM) was selected as the modulation scheme; the coding/modulation scheme for the subcarriers of OFDM symbol is the same as that in IEEE 802.11a. In support of QoS, HIPERLAN/2 adopts a centralized and scheduled MAC mechanism.

The HISWAN standard is being developed by ARIB, the Japanese Multimedia Mobile Access Communication group. The physical layer of HISWAN is the same as IEEE 802.11a.

All three 5 GHz WLAN standards have physical layers based on OFDM. OFDM transmits data simultaneously over multiple, parallel frequency sub-bands and offers robust performance under severe radio channel conditions. OFDM also offers a convenient method for mitigating delay spread effects. A cyclic extension of the transmitted OFDM symbol can be used to achieve a guard interval between symbols. Provided that this guard interval exceeds the excess delay spread of the radio channel, the effect of the delay spread is constrained to frequency selective fading of the individual sub-bands. This fading can be canceled by means of a channel compensator, which takes the form of a single tap equalizer on each sub-band.

The IEEE 802.11a transmitter and receiver OFDM physical layer block diagram is shown in the following figure.

Major specifications for the IEEE 802.11a OFDM physical layer are listed in the following table.



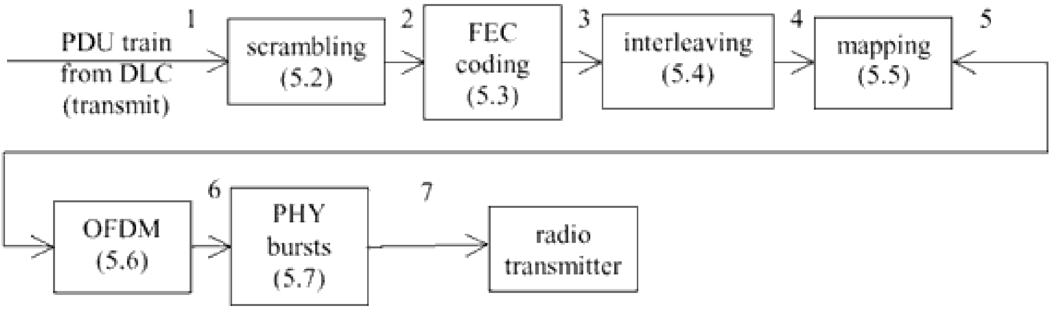
**IEEE 802.11a Transmitter and Receiver for OFDM Physical Layer Block Diagram**

**IEEE 802.11a OFDM Physical Layer Major Specifications**

| **Specification** | **Settings** |
|---|---|
| Information data rate | 6, 9, 12, 18, 24, 36, 48 and 54 Mbps |
| Modulation | BPSK OFDM, QPSK OFDM, 16-QAM OFDM, 64-QAM OFDM |
| Error correcting code | K = 7 (64 states) convolutional code |
| Coding rate | 1/2, 2/3, 3/4 |
| Number of subcarriers | 52 |
| OFDM symbol duration | 4.0 μs |
| Guard interval | 0.8 μs ( $T_{GI}$ ) |
| Occupied bandwidth | 16.6 MHz |

The HIPERLAN/2 transmitter is shown in the following figure.

Major specifications for the HIPERLAN/2 OFDM physical layer are listed in the following table.



**HIPERLAN/2 Transmitter for OFDM Physical Layer Block Diagram**

**HIPERLAN/2 Transmitter for OFDM Physical Layer Block Diagram**

**HIPERLAN/2 OFDM Physical Layer Major Specifications**

| **Specification** | **Settings** |
|---|---|
| Information data rate | 6, 9, 12, 18, 24, 27, 36, and 54 Mbps |
| Modulation | BPSK OFDM, QPSK OFDM, 16-QAM OFDM, and 64-QAM OFDM |
| Error correcting code | K = 7 (64 states) convolutional code |
| Coding rate | 1/2, 3/4, 9/16 |
| Number of subcarriers | 52 |
| Sampling rate $f_s$ = $1/T$ | 20 MHz |
| Useful symbol part duration $T_U$ | $64 \times T$ 3.2 us |
| Cyclic prefix duration $T_{CP}$ | $16 \times T$ 0.8 us (mandatory)<br>$8 \times T$ 0.4 us (mandatory) |
| Symbol interval $T_S$ | $80 \times T$ 4.0 us ($T_U + T_{CP}$ )<br>$72 \times T$ 3.6 us ($T_U + T_{CP}$ ) |
| Sub-carrier spacing $\Delta_f$ | 0.3125 MHz ($1/T_U$ ) |
| Spacing between the two outmost sub-carriers | 16.25 MHz |

# Component Libraries

The WLAN Design Library is organized by library according to the types of behavioral models and subnetworks.

## 11b Receivers

This library provides models for use with IEEE 802.11b receivers.

- WLAN_11bBurstRec: 11b burst receiver
- WLAN_11bBurstSync: 11b burst synchronizer
- WLAN_11bCIREstimator: channel estimator for 802.11b
- WLAN_11bDFE: decision feedback equalizer for 11b
- WLAN_11bDemuxBurst: 11b burst demultiplexer and frequency compensator
- WLAN_11bDescrambler: 11b descrambler
- WLAN_11bFreqEstimator: 11b frequency offset estimator
- WLAN_11bPreamble: signal source of IEEE 802.11b preamble
- WLAN_11bRake: rake combiner for 802.11b
- WLAN_11b_Equalizer: 802.11b receiver with equalizer
- WLAN_11b_Rake: 802.11b Rake receiver
- WLAN_CCKDemod: 11b CCK demodulator
- WLAN_CCK_RF_Rx_DFE: 802.11b CCK receiver with equalizer
- WLAN_CCK_RF_Rx_Rake: 802.11b CCK Rake receiver
- WLAN_CCK_Rx_DFE: 802.11b CCK receiver with equalizer
- WLAN_CCK_Rx_Rake: 802.11b CCK Rake receiver
- WLAN_Despreader: barker despreader for 11b
- WLAN_FcCompensator: carrier frequency compensation for 802.11b
- WLAN_HeaderDemap: header demapper
- WLAN_PhaseRotator: phase rotator after decision feedback equalizer for 11b
- WLAN_PrmblDemap: preamble demapper
- WLAN_RecFilter: receiver matched filter

## 11b Signal Sources

This library provides IEEE 802.11b signal source generator. All models can only be used with IEEE 802.11b.

- WLAN_11SignalSrc: signal source of IEEE 802.11 with idle
- WLAN_11bCCKSignalSrc: signal source of IEEE 802.11b with idle and CCK modulation
- WLAN_11bCCKSignalSrc1: signal source of IEEE 802.11b with idle and CCK modulation
- WLAN_11bCCK_RF: RF Signal source of IEEE 802.11b with idle and CCK modulation
- WLAN_11bMuxBurst: IEEE 802.11b burst multiplexer
- WLAN_11bPBCCSignalSrc: signal source of IEEE 802.11b with idle and PBCC modulation
- WLAN_11bScrambler: IEEE 802.11b scrambler
- WLAN_Barker: barker spreader

- WLAN_CCKMod: CCK modulator
- WLAN_CRC: CRC calculation
- WLAN_HeaderMap: header mapper
- WLAN_IdlePadding: idle padding
- WLAN_MuxPLCP: PLCP multiplexer
- WLAN_PBCCConvCoder: PBCC convolutional encoder
- WLAN_PBCCMod: PBCC modulator
- WLAN_PLCPHeader: IEEE 802.11b PLCP header without CRC
- WLAN_PLCPPreamble: IEEE 802.11b PLCP preamble
- WLAN_PSDUMap: PSDU mapper
- WLAN_PreambleMap: preamble mapper
- WLAN_TransFilter: pulse-shaping filter

## Channel Components

This library provides the WLAN channel model.

- WLAN_ChannelModel: WLAN channel model

## Channel Coding Components

This library provides models for channel coding, scrambling and interleaving in the transmitter end, and channel decoding and deinterleaving in the receiving end. All models can be used with IEEE 802.11a and HIPERLAN/2 systems.

- WLAN_11aConvDecoder: 11a viterbi decoder
- WLAN_ConvCoder: convolutional coding of input bits
- WLAN_ConvDecoder: bit-by-bit viterbi decoder for WLAN convolutional code (for IEEE 802.11a, HIPERLAN/2, and MMAC systems)
- WLAN_ConvDecoder1_2: convolutional decoder for 1/2 rate
- WLAN_Deinterleaver: deinterleaving of input bits
- WLAN_Interleaver: interleave input bits
- WLAN_PuncCoder: puncture coder
- WLAN_PuncCoderP1: puncture coder pattern P1 for HIPERLAN/2 systems
- WLAN_PuncConvCoder: punctured convolutional encoder
- WLAN_PuncConvDecoder: punctured convolutional decoder
- WLAN_PuncDecoder: puncture decoder
- WLAN_PuncDecoder1: punctured convolutional decoder (for IEEE 802.11a, HIPERLAN/2 and MMAC systems)
- WLAN_PuncDecoderP1: puncture decoder pattern P1 for HIPERLAN/2 systems
- WLAN_Scrambler: scramble the input bits

## Measurements

This library provides models for BER/PER, EVM and constellation measurements for IEEE 802.11a systems.

- WLAN_80211a_BERPER: Bit and packet error rate measurements sink
- WLAN_80211a_Constellation: Constellation measurement sink
- WLAN_80211a_EVM: 802.11a EVM measurement
- WLAN_80211a_RF_EVM: EVM model for WLAN EVM Measurement
- WLAN_DSSS_CCK_PBCC_EVM: EVM measurement for DSSS/CCK/PBCC WLAN signals (802.11b and non-OFDM 802.11g)

## Modulation Components

OFDM subcarriers are modulated using BPSK, QPSK, 16-QAM or 64-QAM modulation. This library provides models for BPSK, QPSK, 16-QAM or 64-QAM modulation and demodulation for IEEE 802.11a, HIPERLAN/2, and MMAC systems.

- WLAN_BPSKCoder: BPSK mapping
- WLAN_BPSKDecoder: BPSK demapping
- WLAN_Demapper: BPSK, QPSK, 16-QAM or 64-QAM demapping according to data rate
- WLAN_Mapper: BPSK, QPSK, 16-QAM or 64-QAM mapping according to data rate
- WLAN_QAM16Coder: 16-QAM mapping
- WLAN_QAM16Decoder: 16-QAM demapping
- WLAN_QAM64Coder: 64-QAM mapping
- WLAN_QAM64Decoder: 64-QAM demapping
- WLAN_QPSKCoder: QPSK mapping
- WLAN_QPSKDecoder: QPSK demapping
- WLAN_SoftDemapper: 11a soft demapper

## Multiplex Components

This library provides models for IEEE 802.11a and HIPERLAN/2 systems.

- WLAN_BurstOut: real burst output
- WLAN_BurstReceiver: burst receiver
- WLAN_CommCtrl2: 2-input commutator with input particle number control
- WLAN_CommCtrl3: 3-input commutator with input particle number control
- WLAN_CosRollWin: cosine-rolloff window function
- WLAN_DemuxBurst: burst demultiplexer with frequency offset compensator and guard interval remover
- WLAN_DemuxBurstNF: burst demultiplexer with guard interval remover, without frequency offset compensator
- WLAN_DemuxOFDMSym: OFDM signal demultiplexer
- WLAN_DemuxSigData: signal and data signal demultiplexer
- WLAN_DistCtrl2: 2-output distributor with output particle number control
- WLAN_DistCtrl3: 3-output distributor with output particle number control
- WLAN_H2CosRollWin: adds cosine-rolloff windows to burst signals for HIPERLAN/2
- WLAN_H2MuxOFDMSym: OFDM symbol multiplexer for HIPERLAN/2
- WLAN_InsertZero: insert zeros before data with input particle number control
- WLAN_LoadIFFTBuff: data stream loader into IFFT buffer
- WLAN_MuxBrdBurst: broadcast burst multiplexer for HIPERLAN/2
- WLAN_MuxBurst: burst multiplexer
- WLAN_MuxBurstNW: burst multiplexer without window function

- WLAN_MuxDLBurst: downlink burst multiplexer for HIPERLAN/2
- WLAN_MuxDataChEst: data and estimated channel impulse response multiplexer
- WLAN_MuxDiBurst: direct link burst multiplexer for HIPERLAN/2
- WLAN_MuxOFDMSym: OFDM symbol multiplexer
- WLAN_MuxSigData: signal and data multiplexer
- WLAN_MuxULBurstL: uplink burst with long preamble multiplexer for HIPERLAN/2
- WLAN_MuxULBurstS: uplink burst with short preamble multiplexer for HIPERLAN/2

## Receivers

This library provides models for use with IEEE 802.11a receivers.

- WLAN_80211aRxFSync: IEEE 802.11a receiver with full frequency synchronization function
- WLAN_80211aRxFSync1: IEEE 802.11a receiver with full frequency synchronization function
- WLAN_80211aRxNoFSync: IEEE 802.11a receiver without full frequency synchronization function
- WLAN_80211aRxNoFSync1: IEEE 802.11a receiver without full frequency synchronization function
- WLAN_80211aRx_Soft: IEEE 802.11a receiver with full frequency synchronization
- WLAN_80211a_RF_RxFSync: IEEE 802.11a receiver with full frequency synchronization
- WLAN_80211a_RF_RxNoFSync: IEEE 802.11a receiver without frequency synchronization
- WLAN_80211a_RF_Rx_Soft: IEEE 802.11a receiver with full frequency synchronization
- WLAN_BurstSync: burst synchronizer
- WLAN_ChEstimator: channel estimator
- WLAN_FineFreqSync: fine carrier frequency synchronizer
- WLAN_FreqSync: carrier frequency synchronizer
- WLAN_OFDMEqualizer: OFDM equalizer by the channel estimation
- WLAN_PhaseEst: phase estimator
- WLAN_PhaseTrack: phase tracker in OFDM demodulation
- WLAN_RmvNullCarrier: null sub-carrier remover in OFDM

## Signal Sources

This library provides short and long training sequence generators and signal and data bits generators. All models can be used with IEEE 802.11a and HIPERLAN/2 systems.

- WLAN_80211aSignalSrc: IEEE 802.11a signal source
- WLAN_80211aSignalSrc1: IEEE 802.11a signal source with idle
- WLAN_80211a_RF: IEEE 802.11a signal source with RF modulation
- WLAN_80211a_RF_WithPN: IEEE 802.11a signal source with RF modulation and phase noise
- WLAN_DATA: data part of PPDU
- WLAN_ExtrPSDU: extract PSDU from data
- WLAN_LPreambleGen: long training sequence generator

- WLAN_PSDU: source of coder
- WLAN_SIGNAL: signal part of PPDU
- WLAN_SPreambleGen: short training sequence generator
- WLAN_Tail: attach tail bits

## Test Components

This library provides auxiliary models for basic measurements.

- WLAN_BERPER: bit and packet error rate measurements
- WLAN_EVM: error vector magnitude
- WLAN_RF_CCDF: RF signal complementary cumulative distribution function
- WLAN_RF_PowMeas: power level measurement

# Glossary of Terms

| | |
|---|---|
| ACPR | adjacent channel power ratio |
| ARIB | Association of Radio Industries and Business |
| BPSK | binary phase shift keying |
| BRAN | broadband radio access network |
| CIR | channel impulse response |
| CSMA/CA | carrier sense multiple access/collision avoidance |
| ETSI | European Telecommunication Standard Institute |
| EVM | error vector magnitude |
| FEC | forward error correction |
| FFT | fast fourier transform |
| GI | guard interval |
| HIPERLAN | high performance local area network |
| HISAW | high-speed wireless area network |
| IEEE | Institute of Electrical and Electronic Engineering |
| IFFT | inverse fast fourier transform |
| MAC | medium access control |
| MMAC | multimedia mobile access communication |
| OFDM | orthogonal frequency division multiplexing |
| PA | power amplifier |
| PHY | physical layer |
| PHY-SAP | physical layer service access point |
| PPDU | PLCP protocol data unit |
| PSDU | PLCP service data unit |
| QAM | quadrature amplitude modulation |
| QPSK | quadrature phase shift keying |
| SDU | service data unit |
| WLAN | wireless local area network |

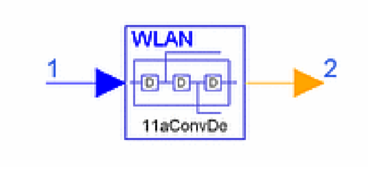
# References

1. IEEE Std 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-speed Physical Layer in the 5GHZ Band."
2. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) Layer."
3. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions."
4. H. Sudo, K. Ishikawa and G-i, Ohta, "OFDM Transmission Diversity Scheme For MMAC Systems," Proceedings of VTC Spring 2000, Vol.1, pp.410-414.
5. Richard Van Nee and Ramjee Prasad, "OFDM For Wireless Multimedia Communications," Artech House Publishers, Boston & London, 1999.
6. IEEE Std 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band."
7. IEEE Std 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications."

# Channel and Channel Coding Components

- *WLAN 11aConvDecoder* (wlan)
- *WLAN ConvCoder* (wlan)
- *WLAN ConvDecoder* (wlan)
- *WLAN Deinterleaver* (wlan)
- *WLAN Interleaver* (wlan)
- *WLAN PuncCoder* (wlan)
- *WLAN PuncCoderP1* (wlan)
- *WLAN PuncConvCoder* (wlan)
- *WLAN PuncDecoder* (wlan)
- *WLAN PuncDecoder1* (wlan)
- *WLAN PuncDecoderP1* (wlan)
- *WLAN Scrambler* (wlan)

# WLAN_11aConvDecoder



**Description 11a viterbi decoder**
**Library WLAN, Channel Coding**
**Class SDFWLAN_11aConvDecoder**
**Derived From WLAN_ViterbiDecoder1**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| TrunLen | path memory truncation length | 60 | int | [20, 200) |
| InputFrameLen | input bits | 288 | int | [2, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | code words to be viterbi-decoded. | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | decoded bits. | int |

## Notes/Equations

1. This component is used to viterbi-decode the input code words burst by burst and the initial state of the decoder is all zero. InputFrameLen/2 output tokens are produced when InputFrameLen input tokens are consumed.
2. **Viterbi Decoding Algorithm**
   CC(2,1,7) is used as an example in the following algorithm. The generator functions of the code are: g0 which equals 133 (octal) and g1 which equals 171 (octal). Because the constraint length is 7, there are 64 possible states in the encoder. In the Viterbi decoder all states are represented by a single column of nodes in the trellis at every symbol instant. At each node in the trellis, there are 2 merging paths; the path with the shortest distance is selected as the survivor.
   In WLAN systems, the encoded packets are very long; it is impractical to store the entire length of the surviving sequences before determining the information sequence when decoding delay and memory is concerned. Instead, only the most recent L information bits in each surviving sequence are stored. Once the path with the shortest distance is identified, the symbol associated with the path L periods ago is conveyed to the output as a decoded information symbol. Generally, parameter L (normally L ≥ 5K) is sufficiently large for the present symbol of the surviving sequences to have a minimum effect on decoding of the *L* th previous symbol. In WLAN systems, L=TrunLen.
   The following is the Viterbi algorithm for decoding a CC(n,k,K) code, where K is the constraint length of convolutional code. In our components, the convolutional code is

processed with k=1.

**Branch Metric Calculation**

Branch metric $m_j^{(\alpha)}$ , at the *J* th instant of the α path through the trellis is defined as the logarithm of the joint probability of the received n-bit symbol

$$r_{j1} r_{j2} \dots r_{jn}$$

conditioned on the estimated transmitted n-bit symbol

$$c_{j1}^{(\alpha)} c_{j2}^{(\alpha)} \dots c_{jn}^{(\alpha)}$$

for the α path. That is,

$$m_j^{(\alpha)} = \ln\left(\prod_{i=1}^{n} P(r_{ji} | c_{ji}^{(\alpha)})\right)$$

$$= \sum^{n} \ln P(r_{ji} | c_{ji}^{(\alpha)}).$$

If receiver is regarded as a part of the channel, for the Viterbi decoder the channel can be considered as an AWGN channel. Therefore,

$$m_j^{(\alpha)} = \sum_{i=1}^{n} r_{ji} c_{ji}$$

**Path Metric Calculation**

The path metric $M^{(\alpha)}$ for the α path at the *J* th instant is the sum of the branch metrics belonging to the α path from the first instant to the *J* th instant. Therefore,

$$M^{(\alpha)} = \sum_{i=1}^{J} m_j^{(\alpha)}$$

**Information Sequence Update**

There are 2 $^k$ merging paths at each node in the trellis and the decoder selects from paths $\alpha_1, \alpha_2, \dots, \alpha_{2^k}$ the one having the largest metric, namely,

$$max(M^{(\alpha_1)}, M^{(\alpha_2)}, \dots, M^{(\alpha_{2^k})})$$

and this path is known as the survivor.

**Decoder Output**

When the survivor has been determined at the *J* th instant, the decoder outputs the ( *J-L* )th information symbol from its memory of the survivor with the largest metric.

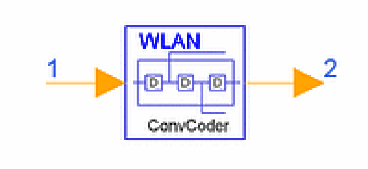
### References

1. IEEE Std 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and

Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

2. S. Lin and D. J. Costello, Jr., *Error Control Coding Fundamentals and Applications* , Prentice Hall, Englewood Cliffs NJ, 1983.

# WLAN_ConvCoder



**Description Convolutional coding the input bits**
**Library WLAN, Channel Coding**
**Class SDFWLAN_ConvCoder**

### Pin Inputs

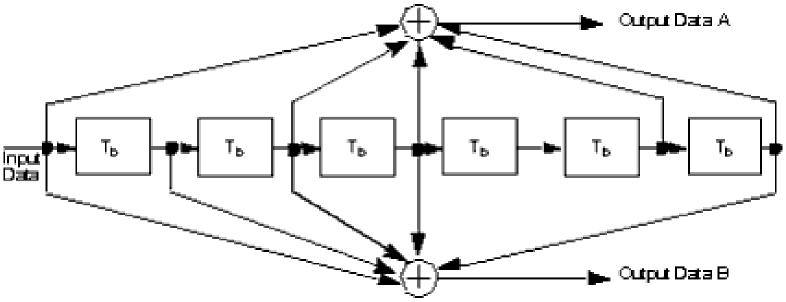
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | bits to be coded | int |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | coded bits | int |

### Notes/Equations

1. This model is used to perform normal convolutional encoding of data rate 1/2 over the input signal.
   Each firing, 1 token is consumed and 2 tokens are produced.
2. Referring to the following figure, the generator polynomial is $G1 = 133_{oct}$ for output A and $G2 = 171_{oct}$ for output B.



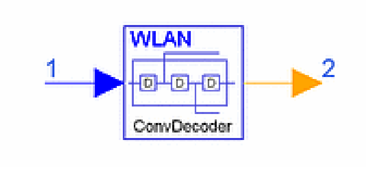
Convolutional Code of Rate 1/2 (Constraint Length=7)

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz

Band," 1999.

# WLAN_ConvDecoder



**Description Bit by bit viterbi decoder for 11a convolutional code**
**Library WLAN, Channel Coding**
**Class SDFWLAN_ConvDecoder**
**Derived From WLAN_ViterbiDecoder**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| SymbolLen | path memory truncation length | 10 | int | (0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | The code words to be viterbi-decoded. | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | the decoded bits. | int |

## Notes/Equations

1. This component is used to viterbi-decode the input code words. CC(2,1,7) and *g0 133 g1 171* is decoded *.* There is a delay, the length of which is equal to the memory length of convolutional code. Padding bits detect when the code words end.
   One output token is produced when 2 input tokens are consumed.
2. The Viterbi decoding algorithm is described, using CC(2,1,7) as an example.
   Generator functions of the code are g0 which equals 133 (octal), and g1 which equals 171 (octal).
   Because the constraint length is 7, there are 64 possible states in the encoder. In the Viterbi decoder all states are represented by a single column of nodes in the trellis at every symbol instant. At each node in the trellis, there are 2 merging paths; the path with the shortest distance is selected as the survivor.
   In WLAN systems, the encoded packets are very long; it is impractical to store the entire length of the surviving sequences before determining the information sequence when decoding delay and memory is concerned. Instead, only the most recent L information bits in each surviving sequence are stored. Once the path with the shortest distance is identified the symbol associated with the path L periods ago is conveyed to the output as a decoded information symbol. Generally, parameter L is sufficiently large, normally L ≥ 5K, for the present symbol of the surviving sequences to have a minimum effect on decoding of the *L* th previous symbol. In WLAN systems, L=8×SymbolLen.
   The following is the Viterbi algorithm for decoding a CC(n,k,K) code, where K is the constraint length of convolutional code. In our components, the convolutional code is

processed with k=1.

Branch Metric Calculation

Branch metric $m_j^{(\alpha)}$ , at the *J* th instant of the α path through the trellis is defined as the logarithm of the joint probability of the received n-bit symbol $r_{j1}r_{j2}...r_{jn}$ conditioned on the estimated transmitted n-bit symbol $c_{j1}^{(\alpha)}\ c_{j2}^{(\alpha)}\ ...\ c_{jn}^{(\alpha)}$ for the α path. That is,

$$m_j^{(\alpha)} = \ln\left(\prod_{i=1}^{n} P(r_{ji}|c_{ji}^{(\alpha)})\right)$$

$$= \sum^{n} \ln P(r_{ji}|c_{ji}^{(\alpha)}).$$

If Rake receiver is regarded as a part of the channel, for the Viterbi decoder the channel can be considered as an AWGN channel. Therefore,

$$m_j^{(\alpha)} = \sum_{i=1}^{n} r_{ji}c_{ji}$$

### Path Metric Calculation

The path metric $M^{(\alpha)}$ for the α path at the *J* th instant is the sum of the branch metrics belonging to the α path from the first instant to the *J* th instant. Therefore,

$$M^{(\alpha)} = \sum_{i=1}^{J} m_j^{(\alpha)}$$

### Information Sequence Update

There are 2 $^k$ merging paths at each node in the trellis; from paths $\alpha_1, \alpha_2, ... ,\alpha_{2^k}$ the decoder selects the one having the largest metric, namely,

$$max(M^{(\alpha_1)}, M^{(\alpha_2)}, ... , M^{(\alpha_{2^k})})\ ;$$

this path is known as the survivor.

### Decoder Output

When the two survivors have been determined at the *J* th instant, the decoder outputs the ( *J-L* )th information symbol from its memory of the survivor with the largest metric.

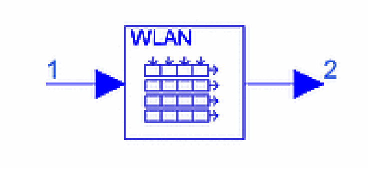
## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz

Band," 1999.

2. S. Lin and D. J. Costello, Jr., *Error Control Coding Fundamentals and Applications* , Prentice Hall, Englewood Cliffs NJ, 1983.

# WLAN_Deinterleaver



**Description Deinterleave the input bits**
**Library WLAN, Channel Coding**
**Class SDFWLAN_Deinterleaver**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input bits to be deinterleaved | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | deinterleaved bits | real |

## Notes/Equations

1. This model is used as a deinterleaver for HIPERLAN/2 and IEEE 802.11a. It performs the inverse relation of an interleaver and is defined by two permutations.
   j will be used to denote the index of the original received bit before the first permutation; i denotes the index after the first and before the second permutation; k denotes the index after the second permutation, just prior to delivering the coded bits to the convolutional (Viterbi) decoder.
   The first permutation is defined by

   $$i = s \times \mathrm{floor}(j/s) + (j + \mathrm{floor}(16 \times j/ N_{CBPS})) \bmod s \quad j = 0,1, \ldots N_{CBPS} - 1$$

   The value of $s$ is determined by the number of coded bits per subcarrier $N_{DBPS}$ according to

   $$s = \max( N_{DBPS} /2, 1)$$

   The second permutation is defined by

   $$k = 16 \times i - ( N_{CBPS} - 1)\mathrm{floor}(16 \times i/ N_{CBPS} ) \quad i = 0, 1, \ldots N_{CBPS} - 1$$

   where $N_{DBPS}$ and $N_{CBPS}$ are determined by data rates listed in the following table.

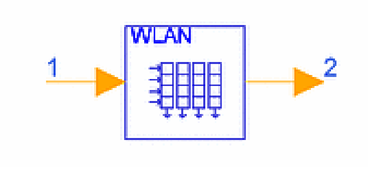
### Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 (IEEE 802.11a) | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 (HIPERLAN/2) | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 (IEEE 802.11a) | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_Interleaver



**Description Interleave the input bits**
**Library WLAN, Channel Coding**
**Class SDFWLAN_Interleaver**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input bits to be interleaved | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | interleaved bits | int |

## Notes/Equations

1. This model is used for HIPERLAN/2 and IEEE 802.11a. Encoded data bits are interleaved by a block interleaver with a block size corresponding to the number of bits in a single OFDM symbol, $N_{CBPS}$ .
2. The interleaver is defined by a two-step permutation. The first permutation ensures that adjacent coded bits are mapped onto nonadjacent subcarriers. The second permutation ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation, thereby avoiding long runs of low reliability bits.
   k will be used to denote the index of the coded bit before the first permutation; i will denote the index after the first and before the second permutation; j will denote the index after the second permutation, just prior to modulation mapping.
   The first permutation is defined by

   $$i = (N_{CBPS}/16)\ (k \bmod 16) + \mathrm{floor}(k/16) \quad k = 0, 1, \ldots, N_{CBPS} - 1$$

   The function floor (.) denotes the largest integer not exceeding the parameter.
   The second permutation is defined by

   $$j = s \times \mathrm{floor}(i/s) + (i + N_{CBPS} - \mathrm{floor}(16 \times i/N_{CBPS})) \bmod s \quad i = 0, 1, \ldots N_{CBPS} - 1$$

$_{CBPS}$

where

$s$ is determined by the number of coded bits per subcarrier $N_{DBPS}$

according to

$s = \max(N_{DBPS}/2, 1)$

where $N_{DBPS}$ and $N_{CBPS}$ are determined by data rates listed in the following table.

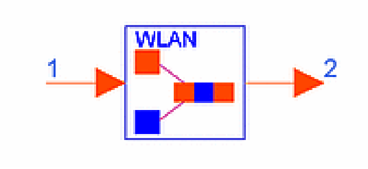
**Rate-Dependent Values**

| Data Rate (Mbps) | Modulation | Coding Rate (R) | Coded Bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 (IEEE 802.11a) | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 (HIPERLAN/2) | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 (IEEE 802.11a) | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_PuncCoder



**Description Puncture coder**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncCoder**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal to be perforated | anytype |

## Pin Outputs

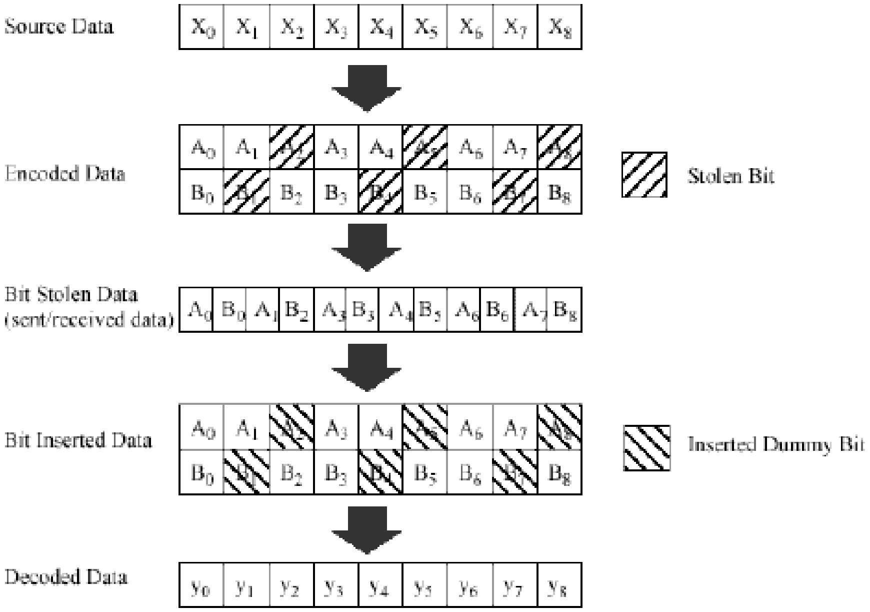
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signal after perforated | anytype |

## Notes/Equations

1. This model is used to perforate the input convolutional code to produce a punctured convolutional code. Each firing, K tokens are consumed and N tokens are produced; K and N are determined by Rate according to the following table.

| Rate | K | N |
|---|---|---|
| Mbps_6 | 2 | 2 |
| Mbps_9 | 6 | 4 |
| Mbps_12 | 2 | 2 |
| Mbps_18 | 6 | 4 |
| Mbps_24 (IEEE 802.11a) | 2 | 2 |
| Mbps_27 (HIPERLAN/2) | 18 | 16 |
| Mbps_36 | 6 | 4 |
| Mbps_48 (IEEE 802.11a) | 4 | 3 |
| Mbps_54 | 6 | 4 |

2. Typically, punctured convolutional code is generated by perforating a mother convolutional code according to a certain pattern to achieve a different data rate. This model determines the perforation pattern according to the Rate selected; the input convolutional coded bits are read to determine whether to output the input bit or simply discard it according to the pattern in the following figure.

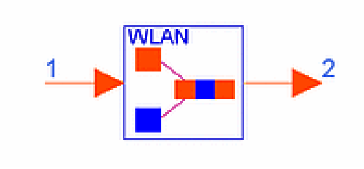


**Puncturing and Transmit Sequence**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_PuncCoderP1



**Description Puncture coder pattern P1**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncCoderP1**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal to be perforated | anytype |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signal after perforated | anytype |

## Notes/Equations

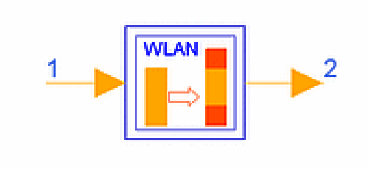
1. This model is used to perforate the input convolutional code to produce a punctured convolutional code. 26 tokens are consumed at input port and 24 tokens are produced after the star is fired.
2. Punctured convolutional code is usually generated by perforating a mother convolutional code according to a certain pattern. This model is rate independent. It reads the input convolutional coded bits and determines either to output the input bit or simply discard it according to the pattern given in the following table.

| PDU-Wise Bit Numbering | Puncture Pattern | Transmit Sequence (after pallel-to-serial conversion) |
|---|---|---|
| 0-155 | X: 1111110111111<br>Y: 1111111111110 | $X_1 Y_1 X_2 Y_2 X_3 Y_3 X_4 Y_4 X_5 Y_5 X_6 Y_6$<br>$X_8 Y_7 X_9 Y_8 X_{10} Y_9 X_{11} Y_{10} X_{12} Y_{11} X_{13} Y_{12}$ |

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_PuncConvCoder



**Description Punctured convolutional encoder**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncConvCoder**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | rate determining punctured convolutional code type: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Length | octet number of PSDU | 100 | int | [1, 4095] |

## Pin Inputs

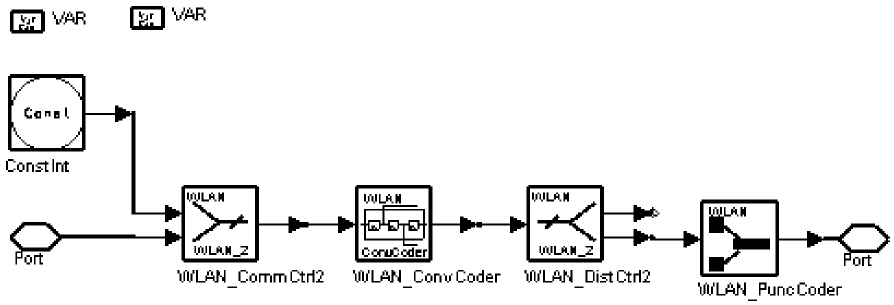
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be encoded | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | encoded signal | int |

## Notes/Equations

1. This subnetwork is used to perform punctured convolutional encoding over the input signal. The schematic for this subnetwork is shown in the following figure.



**WLAN_PuncConvCoder Schematic**

2. A convolutional coding model is used to encode into mother convolutional code of data rate 1/2. A puncture encoder model is used to generate punctured convolutional code. The Rate parameter determines the type of WLAN punctured convolutional code.
Before convolutional coding, 8 zero bits are inserted after NDATA bits. Before punctured coding, 16 zero bits are discarded. The number of OFDM symbols (DATA

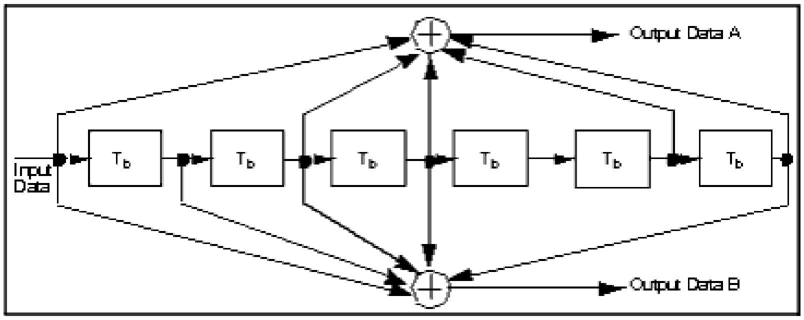
part) $N_{SYM}$ is:

$$N_{SYM} = Ceiling\ ((16 + 8 \times\ Length + 6) / N_{DBPS})$$

where $N_{DBPS}$ is determined by data rate according to the following table.
NDATA is NDBPS × $N_{SYM}$.

| *Data Rate (Mbps)* | Modulation | *Coding Rate (R)* | Coded Bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

3. Referring to the following figure, the generator polynomial $G\,1 = 133_{oct}$ for A output is and $G\,2 = 171_{oct}$ for B output.

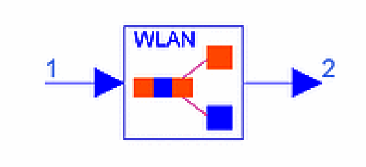


**Mother Convolutional Code of Rate 1/2 (Constraint Length=7)**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_PuncDecoder



**Description Puncture decoder**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncDecoder**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal to be refilled | real |

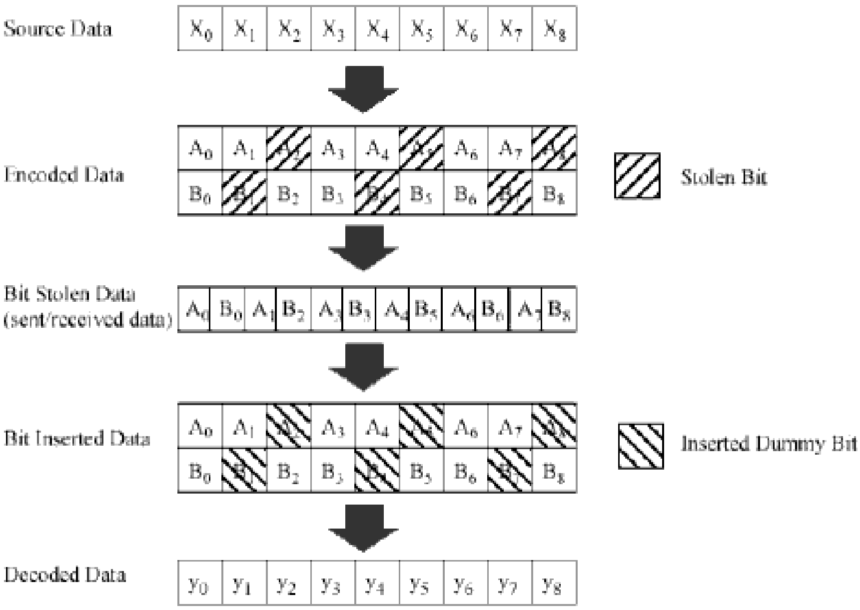
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signal after refilled | real |

## Notes/Equations

1. This model is used to refill the coding that was perforated during the puncture encoding process. It interpolates a zero value to the punctured data stream to form a full-length data stream. The perforation pattern is determined based on Rate, it then interpolates zero into the input bits to form the output.
   Each firing, K tokens are consumed and N tokens are produced. K and N are determined according to the following table.

| Rate | K | N |
|---|---|---|
| Mbps_6 | 2 | 2 |
| Mbps_9 | 4 | 6 |
| Mbps_12 | 2 | 2 |
| Mbps_18 | 4 | 6 |
| Mbps_24 (IEEE 802.11a) | 2 | 2 |
| Mbps_27 (HIPERLAN/2) | 16 | 18 |
| Mbps_36 | 4 | 6 |
| Mbps_48 (IEEE 802.11a) | 3 | 4 |
| Mbps_54 | 4 | 6 |

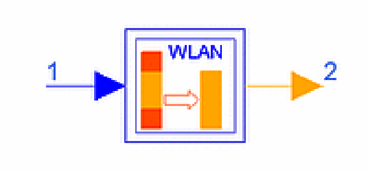


**Puncture and Transmit Sequence**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_PuncDecoder1



**Description Punctured convolutional decoder**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncDecoder1**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Rate | rate determining punctured convolutional code type: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| SymbolLen | path memory truncation length | 10 | int | (0, ∞) |

## Pin Inputs

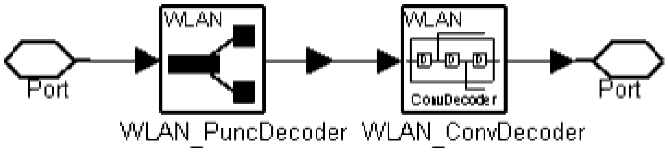
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be decoded | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | decoded signal | real |

## Notes/Equations

1. This subnetwork is used to perform punctured convolutional decoding of data rate 1/2 over the input signal. The schematic is shown in the following figure.



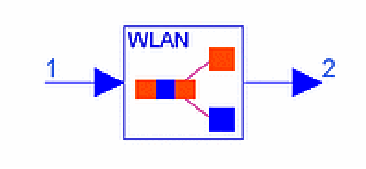
**WLAN_PuncDecoder1 Schematic**

Punctured convolutional encoded input is decoded to normal convolutional coded data. A general Viterbi convolutional decoder is used for further decoding.
The data rate of the mother convolutional code is 1/2; generator polynomials for X output is $G_1 = 171_{oct}$ and $G_2 = 133_{oct}$ for Y output.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. S. Lin and D. J. Costello, Jr., Error Control Coding Fundamentals and Applications, Prentice Hall, Englewood Cliffs NJ, 1983.

# WLAN_PuncDecoderP1



**Description Puncture decoder pattern P1**
**Library WLAN, Channel Coding**
**Class SDFWLAN_PuncDecoderP1**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal to be refilled | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output signal after refilled | real |

## Notes/Equations

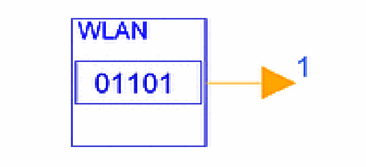
1. This model depunctures data that was perforated during the puncture encoding process.
   Each firing, 24 tokens are consumed at input port and 26 tokens are produced.
2. This model is rate independent. It interpolates zero value to the punctured data stream to form a full-length data stream according to the pattern shown in the following table.

| PDU-Wise Bit Numbering | Puncture Pattern | Transmit Sequence (after pallel-to-serial conversion) |
|---|---|---|
| 0-155 | X: 1111110111111<br>Y: 1111111111110 | $X_1 Y_1 X_2 Y_2 X_3 Y_3 X_4 Y_4 X_5 Y_5 X_6 Y_6$<br>$X_8 Y_7 X_9 Y_8 X_{10} Y_9 X_{11} Y_{10} X_{12} Y_{11} X_{13} Y_{12}$ |

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_Scrambler



**Description Scramble the input bits**
**Library WLAN, Channel Coding**
**Class SDFWLAN_Scrambler**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| InitState | initial state of scrambler | 1 0 1 1 1 0 1 | int array | 0 or 1 array size is 7 |
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

## Pin Outputs

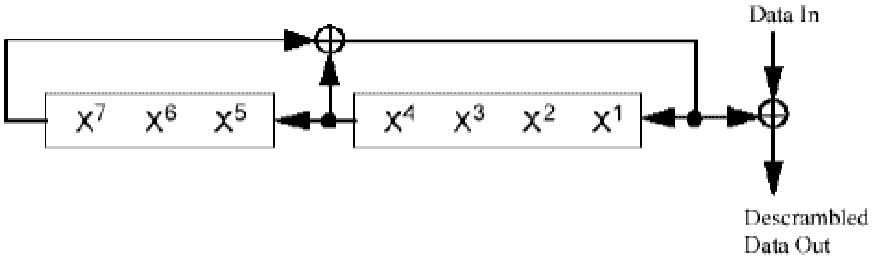
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | output | scramble sequence | int |

## Notes/Equations

1. This model is used for HIPERLAN/2 and IEEE 802.11a to generate scramble sequence used for scrambling and descrambling.
   The length-127 frame synchronous scrambler (the following figure) uses the generator polynomial S(x) as follows. When the *all ones* initial state is used, the 127-bit sequence generated repeatedly by the scrambler (left-most used first) is:

   00001110 11110010 11001001 00000010 00100110 00101110 10110110
   00001100 11010100 11100111 10110100 00101010 11111010 01010001
   10111000 1111111

   The same scrambler is used to scramble transmitted data and descramble received data.



2. According to IEEE 802.11a, the initial state of the scrambler is set to a pseudo random non-zero state. The seven LSBs of the SERVICE field will be set to all zeros prior to scrambling to enable estimation of the initial state of the scrambler in the receiver.
3. According to HIPERLAN/2, all PDU trains belonging to a MAC frame are transmitted by using the same initial state for scrambling. Initialization is performed as follows:

    Broadcast PDU train in case AP uses one sector: scrambler initialized at the 5th bit of BCH, at the 1st bit of FCH, at the 1st bit of ACH without priority, and at the 1st bit of ACH with priority.
    Broadcast PDU train in case AP uses multiple sectors: scrambler initialized at the 5th bit of BCH.
    FCH and ACH PDU train transmitted only in the case of a multiple sector AP: scrambler initialized at the 1st bit of FCH, at the 1st bit of ACH without priority, and at the 1st bit of ACH with priority.
    Downlink PDU train, uplink PDU train with short preamble, uplink PDU train with long preamble, and direct link PDU train: scrambler initialized at the 1st bit of the PDU train.

    The initial state is set to a pseudo random non-zero state determined by the Frame counter field in the BCH at the beginning of the corresponding MAC frame. The Frame counter field consists of the first four bits of BCH, represented by $(n_4 n_3 n_2 n_1)_2$ and is transmitted unscrambled. $n_4$ is transmitted first. The initial state is derived by appending $(n_4 n_3 n_2 n_1)_2$ to the fixed binary number $(111)_2$ in the form $(111 n_4 n_3 n_2 n_1)_2$. For example, if the Frame counter is given as $(0100)_2$, the initial state of the scrambler will be $(1110100)_2$. The transport channel content starting with $(10011101000)_2$ will be scrambled to $(00111110011)_2$.

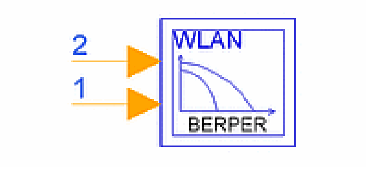
## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# Measurements for WLAN Design Library

- *WLAN 80211a BERPER* (wlan)
- *WLAN 80211a Constellation* (wlan)
- *WLAN 80211a EVM* (wlan)
- *WLAN 80211a RF EVM* (wlan)
- *WLAN DSSS CCK PBCC EVM* (wlan)

# WLAN_80211a_BERPER



**Description Bit and packet error rate measurements sink**
**Library WLAN, Measurements**
**Class SDFWLAN_80211a_BERPER**

## Parameters

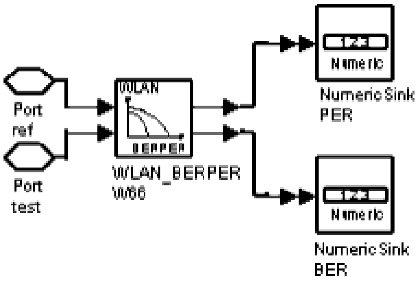
| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Delay | delay number of PSDUs | 2 | int | [0, ∞) |
| Start | start frame of PSDUs | 100 | int | [0, ∞) |
| Stop | stop frame of PSDUs | 100 | int | [Start, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | ref | reference input PSDU | int |
| 2 | test | received PSDU | int |

## Notes/Equations

1. This subnetwork measures bit and packet error rates. The schematic for this subnetwork is shown in the following figure.
   The reference signal of IEEE 802.11a and the received signal inputs are fed into this subnetwork; bit and packet error rates are saved and can be displayed in a Data Display window.



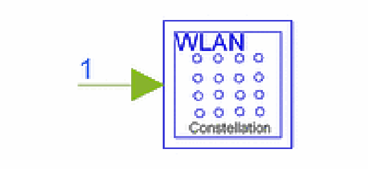
**WLAN_80211a_BERPER Schematic**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211a_Constellation



**Description Constellation measurement sink**
**Library WLAN, Measurements**
**Class SDFWLAN_80211a_Constellation**

## Parameters

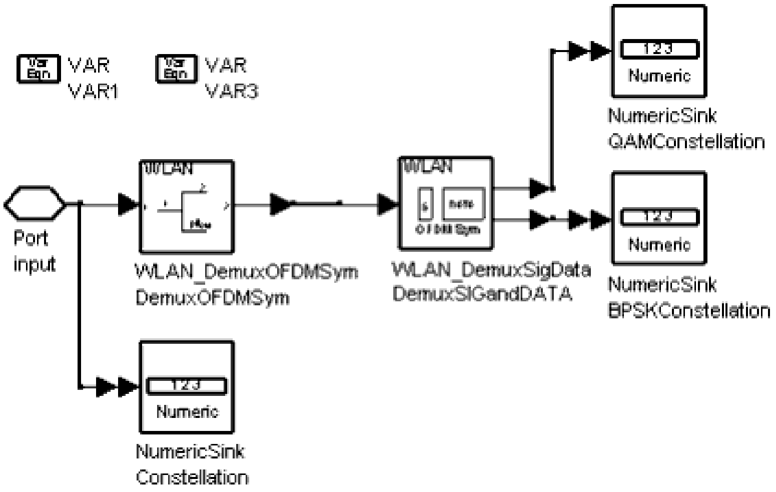
| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Nf | number of frames for the measurement | 1 | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signal to be tested | complex |

## Notes/Equations

1. This subnetwork integrates symbol demultiplexer WLAN_DemuxOFDMSym and SIGNAL and DATA signals demultiplexer WLAN_DemuxSigData, and sinks.
The schematic for this subnetwork is shown in the following figure. Sinks named Constellation, QAMConstellation, and BPSKConstellation show OFDM symbol, DATA, and SIGNAL constellations, respectively. Results are saved and can be displayed in a Data Display window.



**WLAN_80211a_Constellation Schematic**

2. QAMConstellation shows constellations based on Rate values given in the following table.

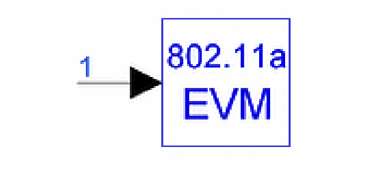
**Rate-Dependent Values**

| Data Rate (Mbps) | Modulation |
|---|---|
| 6 | BPSK |
| 9 | BPSK |
| 12 | QPSK |
| 18 | QPSK |
| 24 | 16-QAM |
| 27 | 16-QAM |
| 36 | 16-QAM |
| 48 | 64-QAM |
| 54 | 64-QAM |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_80211a_EVM



**Description 802.11a EVM measurement**
**Library WLAN, Measurements**
**Class TSDF_WLAN_80211a_EVM**

## Parameters

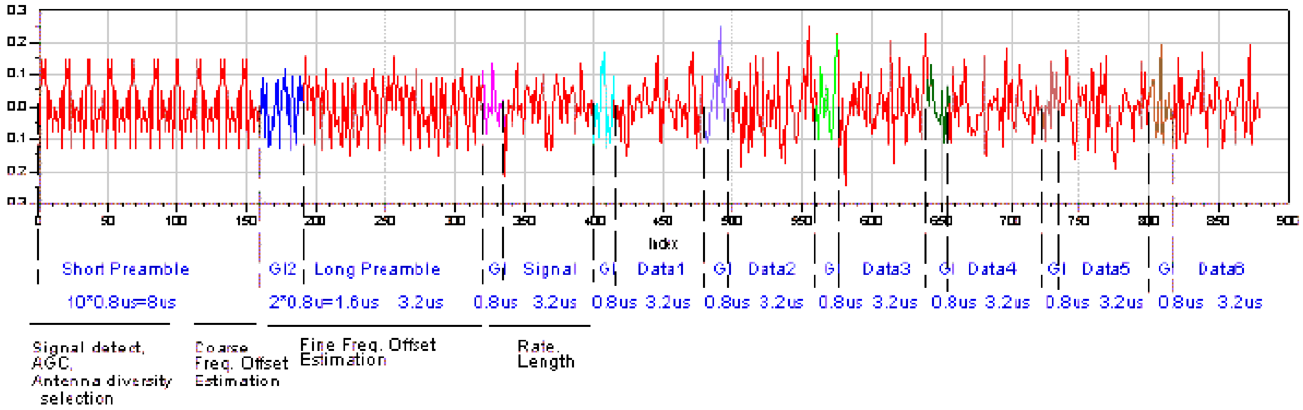
| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RLoad | load resistance. DefaultRLoad will inherit from the DF controller. | DefaultRLoad | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C, of load resistance. DefaultRTemp will inherit from the DF controller. | DefaultRTemp | Celsius | real | [-273.15, ∞) |
| FCarrier | carrier frequency | 5.2e9 | Hz | real | (0, ∞) |
| Start | start time for data recording. DefaultTimeStart will inherit from the DF Controller. | DefaultTimeStart | sec | real | [0, ∞) |
| AverageType | average type: Off, RMS (Video) | RMS (Video) | | enum | |
| FramesToAverage | number of frames that will be averaged if AverageType is RMS (Video) | 20 | | int | [1, ∞) |
| DataSubcarrierModulation | modulation format of the data subcarriers: Auto Detect, BPSK, QPSK, QAM 16, QAM 64 | Auto Detect | | enum | |
| GuardInterval | guard interval time, expressed as a fraction of the FFT time length | 0.25 | | real | [0, 1] |
| SearchLength | search length | 1.0e-3 | sec | real | (0, ∞) |
| ResultLengthType | Auto Select automatically decides the ResultLength value, whereas Manual Override sets it to the specified value: Auto Select, Manual Override | Auto Select | | enum | |
| ResultLength | result length when ResultLengthType is set to Manual Override. If ResultLengthType is set to Auto Select then this value is used as the maximum ResultLength. | 60 | | int | [1, 1367] |
| MeasurementOffset | measurement offset | 0 | | int | [0, ∞) |
| MeasurementInterval | measurement interval | 11 | | int | [1, ∞) |
| SubcarrierSpacing | spacing between subcarriers in Hz | 312.5e3 | Hz | real | (0, ∞) |
| SymbolTimingAdjust | amount of time (expressed as a percent of the FFT time length) to back away from the end of the symbol time when deciding the part of the symbol that the FFT will be performed on | -3.125 | | real | [-100*GuardInterval, 0] |
| Sync | determines whether synchronization will be based on a short or long preamble symbol sequence: Short Training Seq, Channel Estimation Seq | Short Training Seq | | enum | |

**Pin Inputs**

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal | timed |

## Notes/Equations

1. This component performs an EVM measurement for an 802.11a WLAN signal. The input signal must be a timed RF (complex envelope) signal or the component will error out. This measurement provides results for EVMrms_percent, EVM_dB, PilotEVM_dB, CPErms_percent, IQ_Offset_dB, and SyncCorrelation. To use these results in an ael expression or in the *Goal* expression in an optimization setup, you must prefix them with the instance name of the component followed by a dot, for example *W1.EVM_dB* .
   Following is a brief description of the algorithm used (the algorithm used is the same as the one used in the Agilent 89600 VSA) and a detailed description of the parameter usage. The following figure shows the structure of an OFDM burst. Many of the terms mentioned later in these notes such as the preamble, SIGNAL symbol, DATA symbols, guard intervals (GI) are shown in this figure.



**Structure of an OFDM Burst.**

2. Starting at the time instant specified by the Start parameter, a signal segment of length *SearchLength* is acquired. This signal segment is searched in order for a complete burst to be detected. The burst search algorithm looks for both a *burst on* and a *burst off* transition. In order for the burst search algorithm to detect a burst, an idle part must exist between consecutive bursts and the bursts must be at least 15 dB above the noise floor. The recommended minimum duration for idle part is 2 µsec.
   If the acquired signal segment does not contain a complete burst, the algorithm will not detect any burst and the analysis that follows will most likely produce incorrect results. Therefore, SearchLength must be long enough to acquire at least one complete burst. Because the time instant specified by the Start parameter can be soon after the beginning of a burst, it is recommended that SearchLength be set to a value approximately equal to 2 × burstLength + 3 × idle, where burstLength is the duration of a burst in seconds and idle is the duration of the idle part in seconds. If it is known that Start is close to the beginning of a burst then SearchLength can be set to burstLength + 2 × idle. If the duration of the burst or the idle part is unknown, then a TimedSink component can be used to record the signal and the signal can be plotted in the data display. By observing the magnitude of the signal's envelope versus time one can determine the duration of the burst and the idle interval.
   After a burst is detected, synchronization is performed based on the value of the

Sync parameter. The burst is then demodulated (the FCarrier parameter sets the frequency of the internal local oscillator signal). The burst is then analyzed to get the EVM measurement results.

3. If *AverageType* is set to *Off* , only one burst is detected, demodulated, and analyzed. If *AverageType* is set to *RMS (Video)* , after the first burst is analyzed the signal segment corresponding to it is discarded and new signal samples are collected from the input to fill in the signal buffer of length SearchLength. When the buffer is full again a new burst search is performed and when a burst is detected it is demodulated and analyzed. These steps repeat until *FramesToAverage* bursts are processed.
   If for any reason a burst is misdetected the results from its analysis are discarded. The EVM results obtained from all the successfully detected, demodulated, and analyzed bursts are averaged to give the final result.
4. With the DataSubcarrierModulation parameter the designer can specify the data subcarrier modulation format. If DataSubcarrierModulation is set to *Auto Detect* , the algorithm will use the information detected within the OFDM burst (SIGNAL symbol - RATE data field) to automatically determine the data subcarrier modulation format. Otherwise, the format determined from the OFDM burst will be ignored and the format specified by the DataSubcarrierModulation parameter will be used in the demodulation for all data subcarriers. This parameter has no effect on the demodulation of the pilot subcarriers and the SIGNAL symbol, whose format is always BPSK.
5. The GuardInterval parameter specifies the guard interval (also called cyclic extension) length for each symbol time, as a fraction of the FFT time period. The value must match the guard interval length actually used in the input signal in order for the demodulation to work properly.
6. The ResultLengthType and ResultLength parameters control how much data is acquired and demodulated.

   > When ResultLengthType is set to *Auto Select,* the measurement result length is automatically determined from the information in the decoded SIGNAL symbol (LENGTH data field). In this case, the parameter ResultLength defines a maximum result length for the burst in symbol times; that is, if the measurement result length that is automatically detected is bigger than ResultLength it will be truncated to ResultLength.
   > When ResultLengthType is set to *Manual Override* , the measurement result length is set to ResultLength regardless of what is detected from the SIGNAL symbol of the burst. The value specified in ResultLength includes the SIGNAL symbol but does not include any part of the burst preamble.

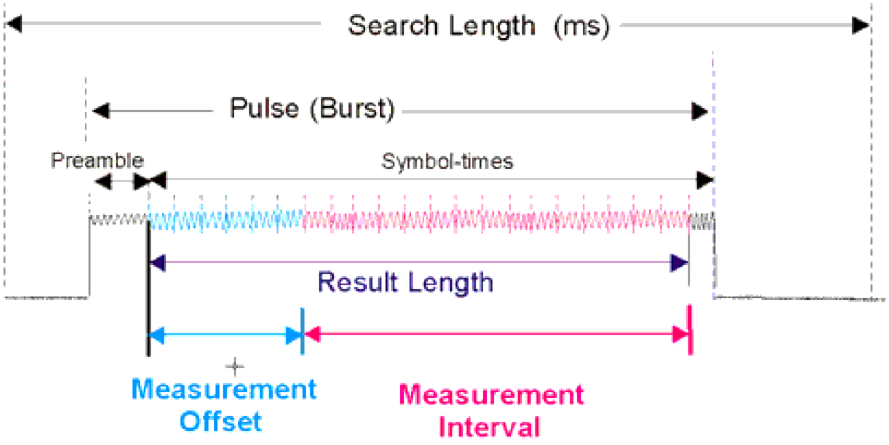
   The following table summarizes the differences between how *Auto Select* and *Manual Override* modes determine the measurement result length. The table lists the measurement result lengths actually used for *Auto Select* and *Manual Override* modes for three different values of the ResultLength parameter (30, 26 and 20 symbol-times). It is assumed that the input burst is 26 symbol-times long.

   **ResultLength Parameter Settings**

| ResultLength Type | ResultLength | Measurement Result Length Actually Used |
|---|---|---|
| Auto Select | 20 | 20 |
| Auto Select | 26 | 26 |
| Auto Select | 30 | 26 |
| Manual Override | 20 | 20 |
| Manual Override | 26 | 26 |
| Manual Override | 30 | 30 |

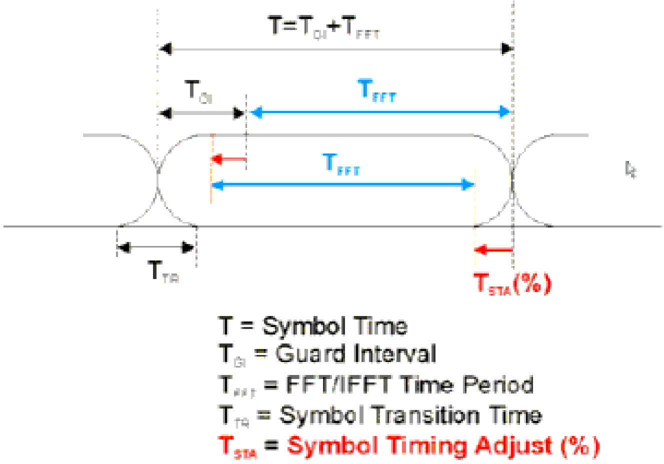
Note that when ResultLengthType is set to *Manual Override* and ResultLength=30 (greater than the actual burst size) the algorithm will demodulate the full 30 symbol-times even though this is 4 symbol-times beyond the burst width.

7. With the MeasurementInterval and MeasurementOffset parameters the designer can isolate a specific segment of the ResultLength for analysis. Only the segment specified by these two parameters will be analyzed in order to get the EVM results. The following figure shows the interrelationship between the SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.



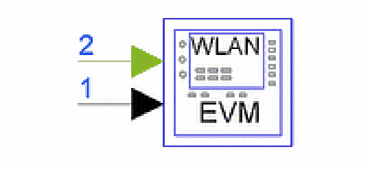
**Interrelationship between SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.**

8. With SubcarrierSpacing parameter the designer can specify the subcarrier spacing of the OFDM signal. The subcarrier spacing must match the actual subcarrier spacing in the input signal in order for the demodulation and analysis to be successful.
9. Normally, when demodulating an OFDM symbol, the guard interval is skipped and an FFT is performed on the last portion of the symbol time. However, this means that the FFT will include the transition region between this symbol and the following symbol. To avoid this, it is generally beneficial to back away from the end of the symbol time and use part of the guard interval. The SymbolTimingAdjust parameter controls how far the FFT part of the symbol is adjusted away from the end of the symbol time. The value is in terms of percent of the used (FFT) part of the symbol time. Note that this parameter value is negative, because the FFT start time is moved back by this parameter. The following figure explains this concept. When setting this parameter, be careful to not back away from the end of the symbol time too much because this may make the FFT include corrupt data from the transition region at the beginning of the symbol time.



**SymbolTimingAdjust Definition**

# WLAN_80211a_RF_EVM



**Description 802.11a EVM Measurement**
**Library WLAN, Measurements**
**Class TSDFWLAN_80211a_RF_EVM**
**Derived From WLAN_ReceiverBase**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RIn | input resistance | DefaultRIn | Ohm | real | (0, ∞) |
| ROut | output resistance | DefaultROut | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C | DefaultRTemp |  | real | [-273.15, ∞) |
| GainImbalance | gain imbalance in dB, Q channel relative to I channel | 0.0 |  | real | (-∞, ∞) |
| PhaseImbalance | phase imbalance in degrees, Q channel relative to I channel | 0.0 |  | real | (-∞, ∞) |
| RefFreq | internal reference frequency | 5200MHz | Hz | real | (0, ∞) |
| Sensitivity | voltage output sensitivity, Vout/Vin | 1 |  | real | (-∞, ∞) |
| Phase | reference phase in degrees | 0.0 | deg | real | (-∞, ∞) |
| Length | octet number of PSDU | 256 |  | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 |  | enum |  |
| Order | FFT points=2^Order | 6 |  | int | [6, 11] |
| ScramblerInit | initial state of scrambler | 1 0 1 1 1 0 1 |  | int array | {0, 1}† |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 |  | enum |  |
| GuardInterval | guard interval defined by user | 16 |  | int | [0, $2^{Order}$] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | (0, ∞) |
| Idle | padded number of zeros between two bursts | 0 |  | int | [0, ∞) |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |
| Nf | number of frames for the measurement | 30 |  | int | [1, ∞) see Note 3 |
| Start | sample number to start collecting numeric data | DefaultNumericStart |  | int | [0, ∞) |
| Stop | sample number to stop collecting numeric data | DefaultNumericStop |  | int | [Start, ∞) |

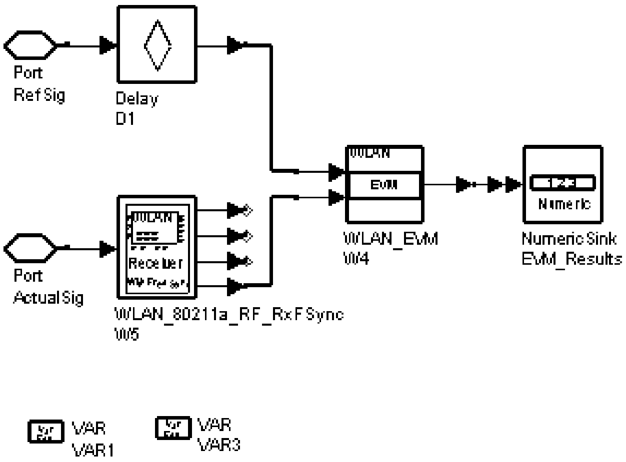
† for each array element: array size must be 7.

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | ActualSig | RF signal | timed |
| 2 | RefSig | reference signal | complex |

## Notes/Equations

1. This subnetwork measures the error vector magnitude of IEEE 802.11a.
   The schematic for this subnetwork is shown in the following figure. IEEE 802.11a timed RF signal and baseband signal (that serves as reference signal for EVM) are fed into this subnetwork. EVM results can be displayed in a Data Display window.

   **WLAN_80211a_RF_EVM Schematic**
2. Nf is the number of frames used to generate an averaged EVM_Results value. Start and Stop define the number (or frame) to start collecting EVM_Results and to stop collecting EVM_Results, respectively. If Start < Nf, then EVM_Results values for indexes < Nf are the sum of the EVM values of all previous frames divided by Nf. For example, if Nf is 5 and Start is 1, then the first EVM_Results value is EVM1 / 5 (where EVM1 is the EVM value of the first frame). The second EVM_Results value is (EVM1 + EVM2) / Nf. Once the index reaches Nf, then the EVM_Results value will be the average EVM of the first Nf frames. For values of index > Nf, the EVM_Results value is the average EVM of the last (most recent) Nf frames. So, it is best to set Start ≥ Nf in the EVM simulation schematic. The first Nf-Start EVM_Results are not correct if Start < Nf.
3. The baseband WLAN_EVM model determines the error vector magnitude. The observed signal is tested in a manner similar to an actual receiver.
   - Start of frame is detected.
   - Transition from short to channel estimation sequences is detected and fine timing (with one sample resolution) is established.
   - Coarse and fine frequency offsets are estimated.
   - The packet is derotated according to estimated frequency offset.
   - The complex channel response coefficients are estimated for each subcarrier.
   - Each data OFDM symbol is transformed into subcarrier received values; the phase from the pilot subcarriers is estimated; subcarrier values are rotated according to the estimated phase; and, each subcarrier value is divided by a complex estimated channel response coefficient.
   - For each data-carrying subcarrier, the closest constellation point is determined and the Euclidean distance from it is calculated.

- The RMS average of all errors in a packet is calculated using the formula

  $$Error_{RMS} = \frac{\sum_{i=1}^{N_f} \sqrt{\frac{\sum_{j=1}^{N_{SYM}} \left[ \sum_{k=1}^{Carriers} \{ (I(i,j,k) - I_0(i,j,k))^2 + (Q(i,j,k) - Q_0(i,j,k))^2 \} \right]}{Carriers \times N_{SYM} \times P_0}}}{N_f}$$

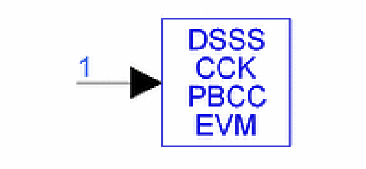
  where
  *Carriers* is the number of subcarriers (48 or 52) in one OFDM symbol
  $N_{SYM}$ is the length of the packet
  $N_f$ is the number of frames for the measurement
  $(I_0(i,j,k), Q_0(i,j,k))$ denotes the ideal symbol point of the $i^{th}$ frame,
  $j^{th}$ OFDM symbol of the frame,
  $k^{th}$ subcarrier of the OFDM symbol in the complex plane
  $(I(i,j,k), Q(i,j,k))$ denotes the observed point of the $i^{th}$ frame,
  $j^{th}$ OFDM symbol of the frame,
  $k^{th}$ subcarrier of the OFDM symbol in the complex plane
  $P_0$ is the average power of the constellation

4. The test must be performed over at least 20 frames $N_f$, and the RMS average taken. Packets under test must be at least 16 OFDM symbols long.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DSSS_CCK_PBCC_EVM



**Description EVM measurement for DSSS/CCK/PBCC WLAN signals (802.11b and non-OFDM 802.11g)**
**Library WLAN, Measurements**
**Class TSDF_WLAN_DSSS_CCK_PBCC_EVM**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| RLoad | load resistance. DefaultRLoad will inherit from the DF controller. | DefaultRLoad | Ohm | real | (0, ∞) |
| RTemp | physical temperature, in degrees C, of load resistance. DefaultRTemp will inherit from the DF controller. | DefaultRTemp | Celsius | real | [-273.15, ∞) |
| FCarrier | carrier frequency in Hz | 2.4e9 | Hz | real | (0, ∞) |
| Start | start time for data recording. DefaultTimeStart will inherit from the DF Controller. | DefaultTimeStart | sec | real | [0, ∞) |
| AverageType | average type: OFF, RMS (Video) | RMS (Video) | | enum | |
| FramesToAverage | number of frames that will be averaged if AverageType is RMS (Video) | 20 | | int | [1, ∞) |
| DataModulationFormat | modulation format: Auto Detect, Barker 1, Barker 2, CCK 5.5, CCK 11, PBCC 5.5, PBCC 11, PBCC 22, PBCC 33 | Auto Detect | | enum | |
| SearchLength | search length in sec | 1.0e-3 | sec | real | (0, ∞) |
| ResultLengthType | setting of ResultLength (see description of ResultLength parameter): Auto Select, Manual Override | Auto Select | | enum | |
| ResultLength | result length (maximum result length) in chips when ResultLengthType = Manual Override (Auto Select) | 2816 | | int | [1, 108344] |
| MeasurementOffset | measurement offset in number of chips | 22 | | int | [0, ∞) |
| MeasurementInterval | measurement interval in number of chips | 2794 | | int | [1, ∞) |
| MirrorFrequencySpectrum | mirror the frequency spectrum: NO, YES | NO | | enum | |
| ChipRate | chip rate in Hz | 11e6 | Hz | real | (0, ∞) |
| ClockAdjust | clock adjustment in chips | 0.0 | | real | [-0.5, 0.5] |
| EqualizationFilter | turn off/on the equalization filter: OFF, ON | OFF | | enum | |
| FilterLength | equalization filter length | 21 | | int | [3, ∞)† |
| DescrambleMode | descramble mode: Off, Preamble Only, Preamble & Header Only, On | On | | enum | |
| ReferenceFilter | reference filter: Rectangular, Gaussian | Rectangular | | enum | |
| ReferenceFilterBT | reference filter BT (used for Gaussian filter) | 0.5 | | real | [0.05, 100] |

† FilterLength must be an odd number.

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signal | timed |

### Notes

1. This component performs an EVM measurement for a CCK or PBCC WLAN burst. This includes all WLAN 802.11b and 802.11g signals with non-OFDM bursts. The input signal must be a timed RF (complex envelope) signal or the component will error out. This measurement provides results for:

Avg_WLAN_80211b_1000_chip_Pk_EVM_pct: average EVM in percentage as specified by the standard (section 18.4.7.8 *Transmit modulation accuracy* in 802.11b specification; pages 55-57) except that the EVM value is normalized
WLAN_80211b_1000_chip_Pk_EVM_pct: EVM in percentage as specified by the standard (section 18.4.7.8 *Transmit modulation accuracy* in 802.11b specification; pages 55-57) with the exception that the EVM value is normalized versus frame
Avg_EVMrms_pct: average EVM rms in percentage as defined in the Agilent 89600 VSA
EVMrms_pct: EVM rms in percentage as defined in the Agilent 89600 VSA versus frame
EVM_Pk_pct: peak EVM in percentage versus frame
EVM_Pk_chip_idx: peak EVM chip index versus frame
Avg_MagErr_rms_pct: average magnitude error rms in percentage
MagErr_rms_pct: magnitude error rms in percentage versus frame
MagErr_Pk_pct: peak magnitude error in percentage versus frame
MagErr_Pk_chip_idx: peak magnitude error chip index versus frame
Avg_PhaseErr_deg: average phase error in degrees
PhaseErr_deg: phase error in degrees versus frame
PhaseErr_Pk_deg: peak phase error in degrees versus frame
PhaseErr_Pk_chip_idx: peak phase error chip index versus frame
Avg_FreqError_Hz: average frequency error in Hz
FreqError_Hz: frequency error in Hz versus frame
Avg_IQ_Offset_dB: average IQ offset in dB
IQ_Offset_dB: IQ offset in dB versus frame
Avg_SyncCorrelation: average sync correlation
SyncCorrelation: sync correlation versus frame

Results named *Avg_* are averaged over the number of frames specified by the designer (if AverageType is set to *RMS* ( *Video* )). Results that are not named *Avg_* are results versus frame. To use any of the results in an ael expression or in the *Goal* expression in an optimization setup, you must prefix them with the instance name of the component followed by a dot, for example *W1.Avg_EVMrms_pct* .
Following is a brief description of the algorithm used (the algorithm used is the same as the one used in the Agilent 89600 VSA) and a detailed description of the parameter usage.

2. Starting at the time instant specified by the Start parameter, a signal segment of length *SearchLength* is acquired. This signal segment is searched in order for a complete burst to be detected. The burst search algorithm looks for both a *burst on* and a *burst off* transition. In order for the burst search algorithm to detect a burst, an idle part must exist between consecutive bursts and the bursts must be at least 15 dB above the noise floor. The recommended minimum duration for idle part is 2 μsec.
If the acquired signal segment does not contain a complete burst, the algorithm will not detect any burst and the analysis that follows will most likely produce incorrect results. Therefore, SearchLength must be long enough to acquire at least one complete burst. Since the time instant specified by the Start parameter can be a little after the beginning of a burst, it is recommended that SearchLength is set to a value approximately equal to 2 × burstLength + 3 × idle, where burstLength is the duration of a burst in seconds and idle is the duration of the idle part in seconds. If it is known that Start is close to the beginning of a burst then SearchLength can be set to burstLength + 2 × idle. If the duration of the burst or the idle part is unknown, then a TimedSink component can be used to record the signal and the signal can be

plotted in the data display. By observing the magnitude of the signal's envelope versus time one can determine the duration of the burst and the idle interval.
After a burst is detected, the I and Q envelopes of the input signal are extracted. The FCarrier parameter sets the frequency of the internal local oscillator signal for the I and Q envelope extraction. Then synchronization is performed based on the preamble. Finally, the burst is demodulated and analyzed to get the EVM measurement results.

3. If AverageType is set to *OFF* , only one burst is detected, demodulated, and analyzed.
   If AverageType is set to *RMS* ( *Video* ), after the first burst is analyzed the signal segment corresponding to it is discarded and new signal samples are collected from the input to fill in the signal buffer of length SearchLength. When the buffer is full again a new burst search is performed; when a burst is detected it is demodulated and analyzed. These steps are repeated until FramesToAverage bursts are processed. If a burst is misdetected for any reason the results from its analysis are discarded. The EVM results obtained from all successfully detected, demodulated, and analyzed bursts are averaged to give the final averaged results. The EVM results from each successfully analyzed burst are also recorded (in the variables that are not named *Avg_* ).
4. With the DataModulationFormat parameter the designer can specify the modulation format used in the PSDU part of the frame. If DataModulationFormat is set to Auto Detect, the algorithm will use the information detected in the PLCP header part of the frame to automatically determine the modulation format. Otherwise, the modulation format determined from the PLCP header is ignored and the modulation format specified by the DataModulationFormat parameter is used in the demodulation of the PSDU part of the frame.
5. The ResultLengthType and ResultLength parameters control how much data is acquired and demodulated.

   > When ResultLengthType is set to *Auto Select* , the measurement result length is automatically determined from the information in the PLCP header part of the frame. In this case, the parameter ResultLength defines a maximum result length for the burst in chips; that is, if the measurement result length that is automatically detected is bigger than ResultLength it will be truncated to ResultLength. The maximum result length specified by the ResultLength parameter includes the PLCP preamble and PLCP header.
   > When ResultLengthType is set to *Manual Override* , the measurement result length is set to ResultLength regardless of what is detected in the PLCP header part of the frame. The result length specified by the ResultLength parameter includes the PLCP preamble and PLCP header.

   The following table summarizes how *Auto Select* and *Manual Override* modes determine the measurement result length. The table lists the measurement result lengths actually used for Auto Select and Manual Override modes for three different values of the ResultLength parameter (3300, 2816 and 2200 chips). It is assumed that the input burst is 2816 chips long.

   **ResultLength Parameter Settings**

| ResultLengthType | ResultLength | Measurement Result Length Actually Used |
|---|---|---|
| Auto Select | 2200 | 2200 |
| Auto Select | 2816 | 2816 |
| Auto Select | 3300 | 2816 |
| Manual Override | 2200 | 2200 |
| Manual Override | 2816 | 2816 |
| Manual Override | 3300 | 3300 |

Note that when ResultLengthType is set to *Manual Override* and ResultLength=3300 (greater than the actual burst size) the algorithm will demodulate the full 3300 chips even though this is 484 chips beyond the burst width.

6. With the MeasurementInterval and MeasurementOffset parameters the designer can isolate a specific segment of the ResultLength for analysis. Only the segment specified by these two parameters will be analyzed in order to get the EVM results. The values of MeasurementInterval and MeasurementOffset are in number of chips and are relative to the ideal starting point of the PLCP preamble portion of the burst. For a signal that uses the long PLCP format, the ideal starting point of the PLCP preamble is exactly 128 symbol times (128 × 11 chips) before the start of the SFD sync pattern. For a signal that uses the short PLCP format, the ideal starting point of the PLCP preamble is exactly 56 symbol times (56 × 11 chips) before the start of the SFD sync pattern.
7. The MirrorFrequencySpectrum parameter can be used to conjugate the input signal (when MirrorFrequencySpectrum is set to *YES* ) before any other processing is done. Conjugating the input signal is necessary if the configuration of the mixers in your system has resulted in a conjugated signal compared to the one at the input of the up-converter and if the preamble and header are *short* format. In this case, if MirrorFrequencySpectrum is not set to *YES* the header bits (which carry the modulation format and length information) will not be recovered correctly so the demodulation of the PSDU part of the frame will most likely fail.
8. The ChipRate parameter specifies the fundamental chip rate of the signal to be analyzed. The default is 11 MHz, which matches the chip rate of 802.11b and 802.11g; however, this parameter can be used when experimenting with signals that do not follow the standard specifications. A special case is the optional 802.11g 33 Mbit PBCC mode, where the chip rate of the transmitted signal starts at 11 MHz, but changes to 16.5 MHz in the middle of the burst. In this case ChipRate should still be set to 11 MHz (the algorithm will automatically switch to 16.5 MHz at the appropriate place in the burst).
9. Although the algorithm synchronizes to the chip timing of the signal, it is possible for the synchronization to be slightly off. The ClockAdjust parameter allows the designer to specify a timing offset which is added to the chip timing detected by the algorithm. This parameter should only be used when trying to debug unusual signals.
10. The EqualizationFilter and FilterLength parameters define whether an equalization filter will be used or not and what the filter length (in number of chips) should be. Using an equalization filter can improve dramatically the EVM results because the equalizer can compensate for ISI caused by the transmit filter. However, an equalization filter can also compensate for distortion introduced by the DUT. If the filter used in the transmitter is Gaussian, then turning the equalizer off and selecting a Gaussian reference filter might be a better option.
11. The DescrambleMode parameter specifies what type of descrambling is done.

    *Off* means no descrambling is done.
    *Preamble Only* means the PLCP preamble is descrambled.
    *Preamble & Header Only* means that the PLCP preamble and PLCP header are descrambled.

*On* means that all parts of the burst are descrambled.

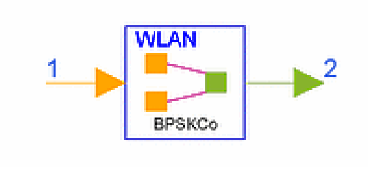
Normally, 802.11b or 802.11g signals have all bits scrambled before transmission, so this parameter should normally be set to *On* . However, when debugging an 802.11b or 802.11g transmitter, it is sometimes helpful to disable scrambling in the transmitter, in which case you should disable descrambling in this component.
If the input signal's preamble is scrambled but you disable descrambling of the preamble (or vice versa), then the algorithm will not be able to synchronize to the signal properly. Similarly, if the input signal's header is scrambled but you disable descrambling of the header (or vice versa) then the algorithm will not be able to correctly identify the burst modulation type and burst length from the header.

12. The ReferenceFilter parameter can be used to select a reference filter for EVM analysis. If a Gaussian reference filter is selected, then the ReferenceFilterBT parameter sets its BT (bandwidth time product).
While the IEEE 802.11b/g standards do not specify either a transmit filter or a receive filter, these do have a spectral mask requirement, and a transmitter must use some sort of transmit filter to meet the spectral mask. On the other hand, the description of the EVM measurement in the standard does not use any receive or measurement filter.
The absence of the need to use any transmit or receive filter is partly because the standard has a very loose limit for EVM (35% peak on 1000 chips of data). If the standard definition is followed when calculating EVM, no measurement or reference filter should be used (ReferenceFilter must be set to Rectangular). However, this means that even a completely distortion-free input signal will still give non-zero EVM unless the input signal has a zero-ISI transmit filter.
If a non-zero-ISI transmit filter is used and additional distortion is added to the signal due to the DUT, then the EVM will measure the overall error due to both the transmit filter ISI and the DUT distortion. Turning on the equalizer will remove most of the transmit filter ISI, but it can also remove some of the distortion introduced by the DUT. To get a better idea of the EVM due to DUT distortion a reference filter that matches the transmit filter can be used.
Currently, only Rectangular and Gaussian filters are available as reference filters.

# Modulation Components

- *WLAN BPSKCoder* (wlan)
- *WLAN BPSKDecoder* (wlan)
- *WLAN Demapper* (wlan)
- *WLAN Mapper* (wlan)
- *WLAN QAM16Coder* (wlan)
- *WLAN QAM16Decoder* (wlan)
- *WLAN QAM64Coder* (wlan)
- *WLAN QAM64Decoder* (wlan)
- *WLAN QPSKCoder* (wlan)
- *WLAN QPSKDecoder* (wlan)
- *WLAN SoftDemapper* (wlan)

# WLAN_BPSKCoder



**Description BPSK mapping**
**Library WLAN, Modulation**
**Class SDFWLAN_BPSKCoder**

## Pin Inputs

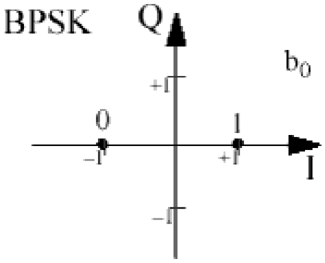
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input data bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after constellation mapping | complex |

## Notes/Equations

1. This model is used to perform BPSK constellation mapping and modulation.
   This model consumes one input token and produces one complex output token.
   Mapping is illustrated in the following figure.

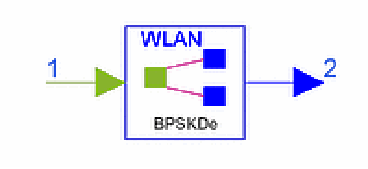


**BPSK Constellation Mapping**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_BPSKDecoder



**Description BPSK demapping**
**Library WLAN, Modulation**
**Class SDFWLAN_BPSKDecoder**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after demodulation | real |

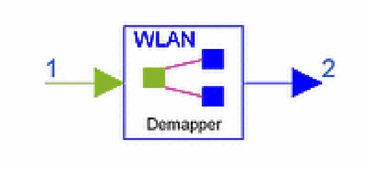
## Notes/Equations

1. This model is used to perform BPSK decoding, which is the reverse process used by WLAN_BPSKCoder.
   This model decodes the complex BPSK signal to float-pointing data to be decoded by a Viterbi convolutional decoder.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_Demapper



**Description BPSK, QPSK 16-QAM or 64-QAM demapping**
**Library WLAN, Modulation**
**Class SDFWLAN_Demapper**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after demodulation | real |

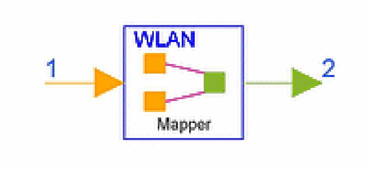
## Notes/Equations

1. This model demaps BPSK, QPSK, 16-QAM or 64-QAM data.
   When Rate is set to 6 or 9 Mbps, the BPSK input signal data will be mapped to floating-point data for Viterbi convolutional decoding according to the BPSK mapping constellation.
   When Rate is set to 12 or 18 Mbps, the QPSK input signal data will be demapped to floating-point data for Viterbi convolutional decoding according to the QPSK mapping constellation.
   When Rate is set to 24, 27, or 36 Mbps, the 16-QAM input signal data will be demapped to floating-point data for Viterbi convolutional decoding according to the 16-QAM mapping constellation.
   When Rate is set to 48 or 54Mbps, the QAM input signal data will be mapped to floating-point data for Viterbi convolutional decoding according to the 64-QAM mapping constellation.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_Mapper



**Description Mapping of BPSK, QPSK 16-QAM or 64-QAM**
**Library WLAN, Modulation**
**Class SDFWLAN_Mapper**

## Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |

## Pin Inputs

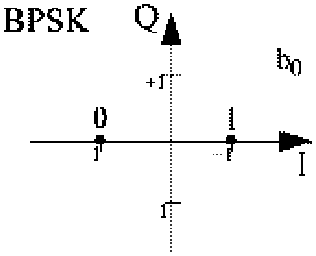
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input data bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after constellation mapping | complex |

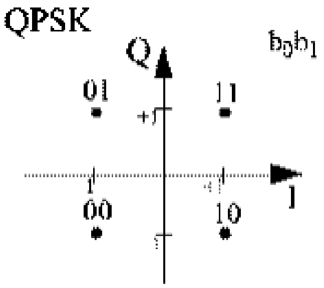
## Notes/Equations

1. This model maps BPSK, QPSK, 16-QAM or 64-QAM data.
   When Rate is 6 or 9 Mbps, BPSK mapping will consume one input bit to produce complex output data, as illustrated in the following figure.

   BPSK Constellation Mapping

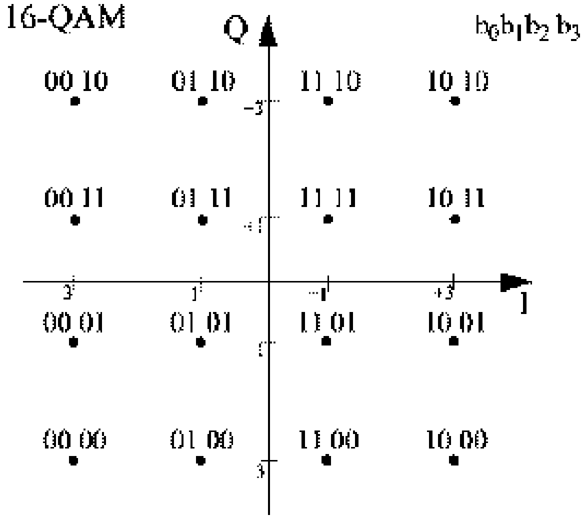
   When Rate is 12 or 18 Mbps, input data bits are formed in 2-bit groups and mapped to complex data as illustrated in the following figure. After mapping, the output signal is normalized by normalization factor *a* , where
   $a = 1/(\sqrt{2})$



**QPSK Mapping and Corresponding Bit Patterns**

When Rate is 24, 27, or 36 Mbps, input data bits will be formed into 4-bit groups and mapped to complex data as illustrated in the following figure. After mapping, the output signal is normalized by normalization factor *a* , where

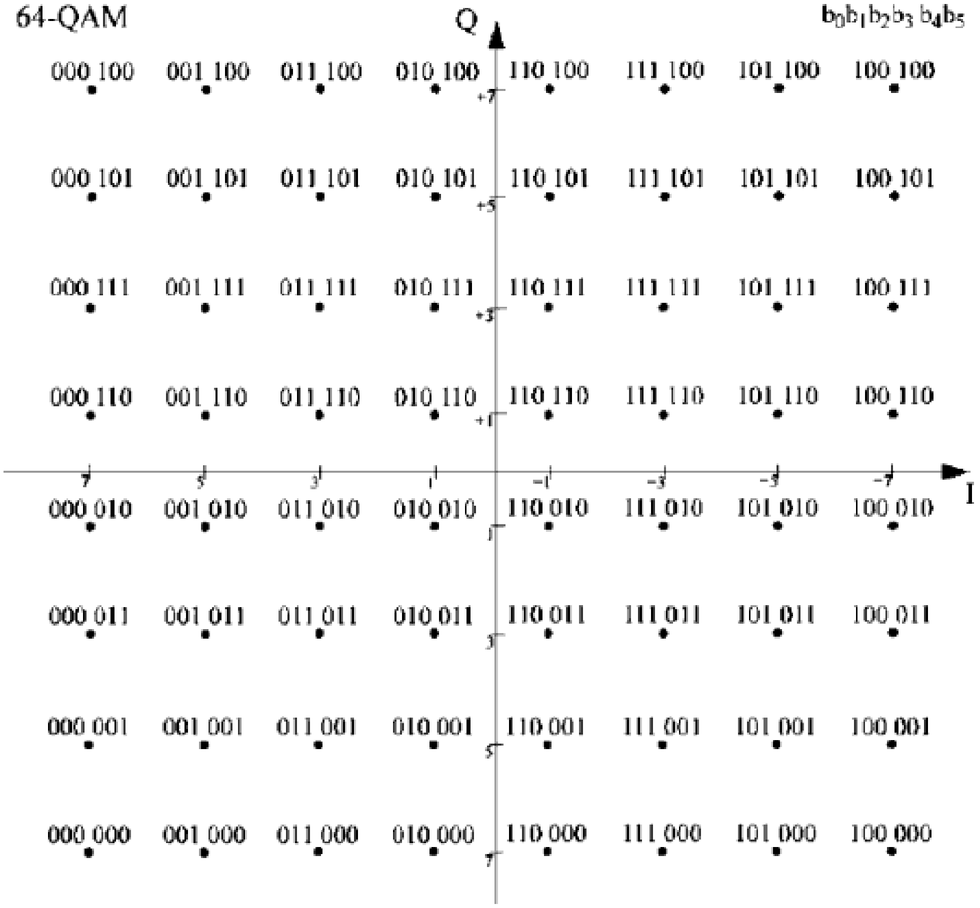
$a = 1/(\sqrt{10})$



**16-QAM Mapping and Corresponding Bit Patterns**

When Rate is 48 or 54Mbps, input data bits are formed in 6-bit groups and mapped to as illustrated in the following figure. After mapping, the output signal is normalized by normalization factor *a,* where

$a = 1/(\sqrt{42})$

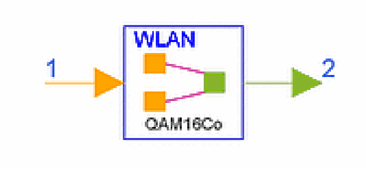


64-QAM Mapping and Corresponding Bit Patterns

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_QAM16Coder



**Description 16-QAM mapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QAM16Coder**

### Pin Inputs

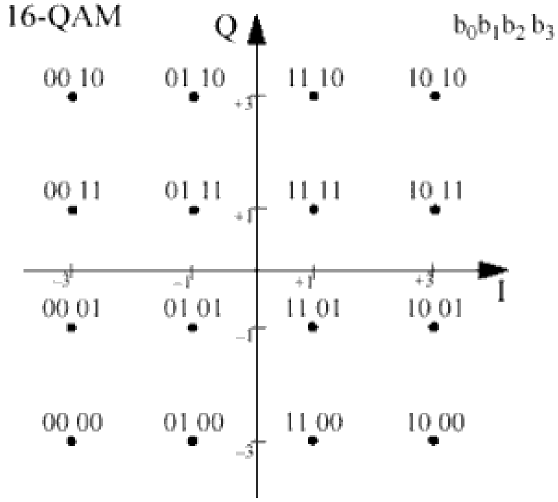
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input data bits | int |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after constellation mapping | complex |

### Notes/Equations

1. This model is used to perform 16-QAM mapping.
   This model groups the input data bits into 4-bit groups and maps them to complex signal from the 16-QAM constellation illustrated in the following figure. After mapping, the output signal is normalized by normalization factor *a*, where
   $a = 1/(\sqrt{10})$

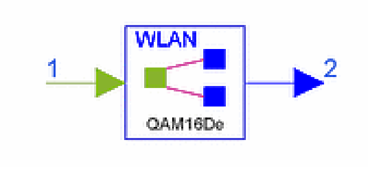


**16-QAM Mapping and Corresponding Bit Patterns**

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_QAM16Decoder



**Description 16-QAM demapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QAM16Decoder**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after demodulation | real |

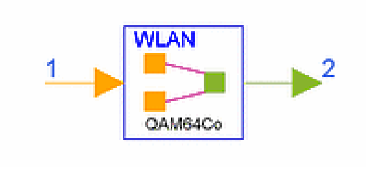
## Notes/Equations

1. This model is used to perform 16-QAM demapping, the reverse of the process performed by WLAN_QAM16Coder.
   Complex QAM input signal data is demapped to floating-point data for Viterbi convolutional decoding according to 16-QAM.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_QAM64Coder



**Description 64-QAM mapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QAM64Coder**

## Pin Inputs

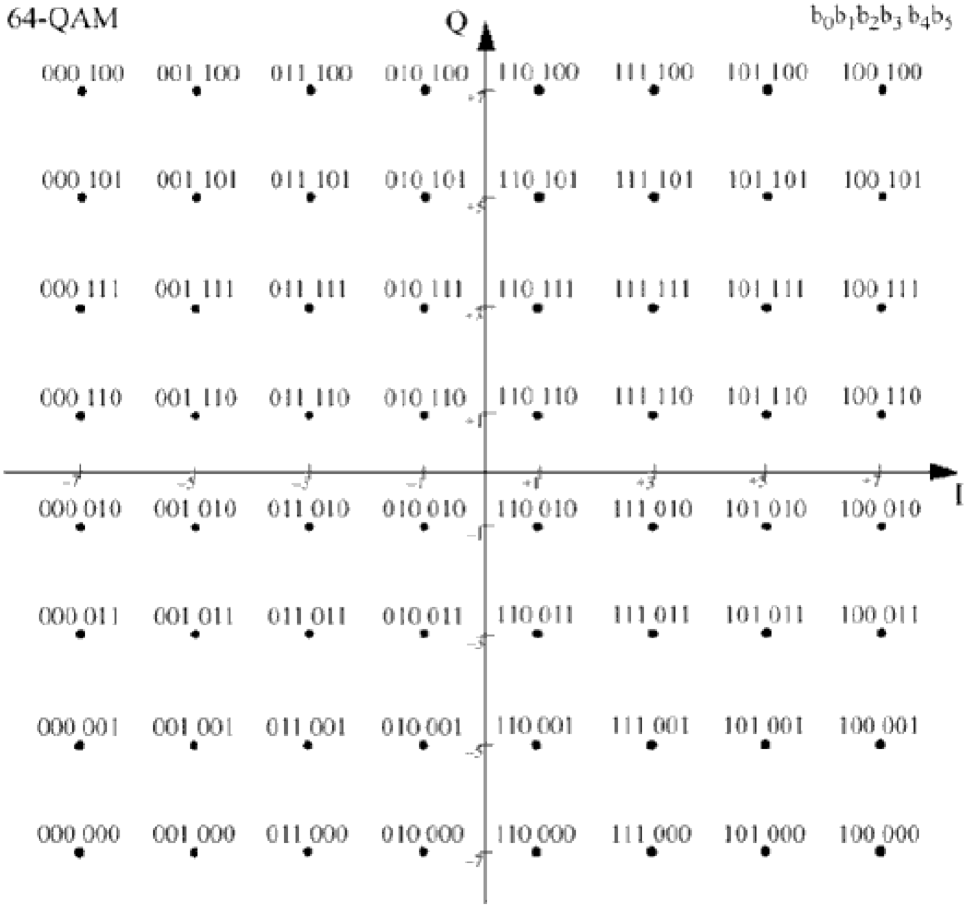
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input data bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after constellation mapping | complex |

## Notes/Equations

1. This model is used to perform 64-QAM mapping.
   This model groups the input data bits to 6-bit groups and maps them to a complex signal from the 64-QAM constellation illustrated in the following figure. After mapping, the output signal is normalized by normalization factor $a$ , where
   $a = 1/(\sqrt{42})$

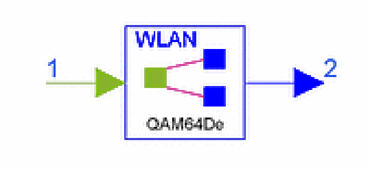


**64-QAM Mapping and Corresponding Bit Patterns**

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_QAM64Decoder



**Description 64-QAM demapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QAM64Decoder**

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after demodulation | real |

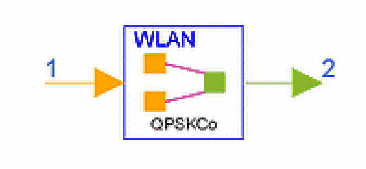
## Notes/Equations

1. This model is used to perform 64-QAM demapping, which is the reverse process of WLAN_QAM64Coder.
   This model demaps the input complex QAM signal data to floating-point data for Viterbi convolutional decoding according to the 64-QAM mapping constellation.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_QPSKCoder



**Description QPSK mapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QPSKCoder**

## Pin Inputs

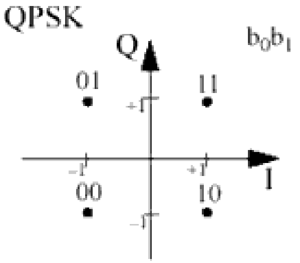
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input data bits | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after constellation mapping | complex |

## Notes/Equations

1. This model is used to perform QPSK constellation mapping and modulation. This model groups the input data bits into 2-bit groups and maps them to complex data according to the QPSK constellation illustrated in the following figure. After mapping, the output signal is normalized by normalization factor *a*, where $a = 1/(\sqrt{2})$.

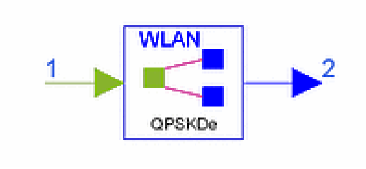


QPSK Mapping and Corresponding Bit Patterns

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_QPSKDecoder



**Description QPSK demapping**
**Library WLAN, Modulation**
**Class SDFWLAN_QPSKDecoder**

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signal after demodulation | real |

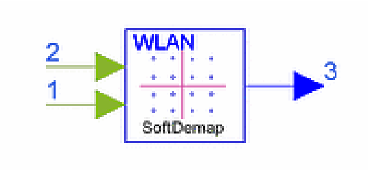
### Notes/Equations

1. This model is used to perform QPSK demodulation, which is the reverse process of WLAN_QPSKCoder.
   This model maps the input complex QPSK signal data to floating-point data for Viterbi convolutional decoding according to the QPSK mapping constellation.

### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_SoftDemappe



**Description 11a soft demapper**
**Library WLAN, Modulation**
**Class SDFWLAN_SoftDemapper**

### Parameters

| Name | Description | Default | Type |
|---|---|---|---|
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum |
| DecoderType | demapping type: Hard, Soft, CSI | CSI | enum |

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be demodulated | complex |
| 2 | CSIBits | channel state information | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | decision bits | real |

### Notes/Equations

1. This model demaps BPSK, QPSK, 16-QAM or 64-QAM data.
    - When Rate is set to 6 or 9 Mbps, BPSK input signal data will be demapped to floating-point data according to the BPSK mapping constellation.
    - When Rate is set to 12 or 18 Mbps, QPSK input signal data will be demapped to floating-point data according to the QPSK mapping constellation.
    - When Rate is set to 24, 27, or 36 Mbps, 16-QAM input signal data will be demapped to floating-point data according to the 16-QAM mapping constellation.
    - When Rate is set to 48 or 54Mbps, the 64-QAM input signal data will be demapped to floating-point data for Viterbi convolutional decoding according to the 64-QAM mapping constellation.
    - If input is multiplied by sqrt(42) and I is the real part of product and Q is the imaginary part, the decision equations for 64-QAM are: $b0 = I;\ b1 = 4 - |I|;\ b2 = 2 - |b1|;\ b3 = Q;\ b4 = 4 - |Q|;\ b5 = 2 - |b4|$.
    - If input is multiplied by sqrt(10) and I is the real part of product and Q is the imaginary part, the decision equations for 16-QAM are: $b0 = I;\ b1 = 2 - |b0|;\ b2 = Q;\ b3 = 2 - |b2|$.
    - If input is multiplied by sqrt(2) and I is the real part of product and Q is the imaginary part, the decision equations for QPSK are: $b0 = I;\ b1 = Q$.

- The decision equation for BPSK is: b0 = I.
  Based on the above calculations, let any one of decision bits equal b:
- when DecoderType is set to Hard, if b < 0, -1.0 is output, otherwise 1.0 is output.
- when DecoderType is set to Soft, if b < -1.0, -1.0 is output; if b > 1.0, 1.0 is output.
- when DecoderType is set to CSI, b is multiplied by CSI (= $|H(i)|^2$ ) and output. Different bits which form one mapping symbol have the same CSI.

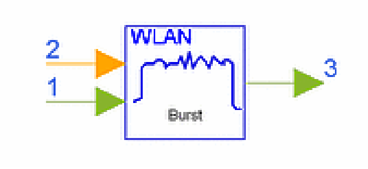
### References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. M. R. G. Butler, S. Armour, P.N. Fletcher, A.R. Nix, D.R. Bull, "Viterbi Decoding Strategies for 5 GHz Wireless LAN Systems," 2001 IEEE.

# Multiplex Components for WLAN Design Library

- *WLAN BurstOut* (wlan)
- *WLAN BurstReceiver* (wlan)
- *WLAN CommCtrl2* (wlan)
- *WLAN CommCtrl3* (wlan)
- *WLAN CosRollWin* (wlan)
- *WLAN DemuxBurst* (wlan)
- *WLAN DemuxBurstNF* (wlan)
- *WLAN DemuxOFDMSym* (wlan)
- *WLAN DemuxSigData* (wlan)
- *WLAN DistCtrl2* (wlan)
- *WLAN DistCtrl3* (wlan)
- *WLAN H2CosRollWin* (wlan)
- *WLAN H2MuxOFDMSym* (wlan)
- *WLAN InsertZero* (wlan)
- *WLAN LoadIFFTBuff* (wlan)
- *WLAN MuxBrdBurst* (wlan)
- *WLAN MuxBurst* (wlan)
- *WLAN MuxBurstNW* (wlan)
- *WLAN MuxDataChEst* (wlan)
- *WLAN MuxDiBurst* (wlan)
- *WLAN MuxDLBurst* (wlan)
- *WLAN MuxOFDMSym* (wlan)
- *WLAN MuxSigData* (wlan)
- *WLAN MuxULBurstL* (wlan)
- *WLAN MuxULBurstS* (wlan)

# WLAN_BurstOut



**Description Output a real burst**
**Library WLAN, Multiplex**
**Class SDFWLAN_BurstOut**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received burst signals | complex |
| 2 | index | synchronization index | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | outputed burst signals | complex |

## Notes/Equations

1. This model is used to output a real burst signal after burst synchronization.
2. Length and Rate parameters determine the number of complex signals in one burst. The number of OFDM symbols, $N_{SYM}$ can be calculated as:

$$N_{SYM} = Ceiling((16 + 8 \times Length + 6)/N_{DBPS})$$

where $N_{DBPS}$ is determined by data rate, shown in the following table.

### Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

After determining $N_{SYM}$ , the number of input tokens $N_{total}$ can be calculated as follows

$$N_{total} = (2^{Order} + 2^{Order-2}) \times (N_{SYM} + 5)$$

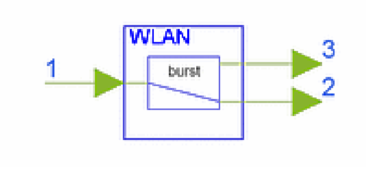
The buffer length for input pin 1 is 2 × $N_{total}$; $N_{total}$ tokens are fired each operation.

Based on the input signal at index pin 2, this model determines the starting point of 10 short preambles, then 2 long preambles, one SIGNAL OFDM symbol and $N_{SYM}$ DATA OFDM symbols, which consist of one burst.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_BurstReceiver



**Description Burst receiver**
**Library WLAN, Multiplex**
**Class SDFWLAN_BurstReceiver**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | int | [0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received signals | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output received signals | complex |
| 3 | sync | output signal for OFDM symbol synchronization | complex |

## Notes/Equations

1. This model is used to output signals for OFDM symbol synchronization, which includes the 10 short preambles.
   Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols, $N_{SYM}$ is:

   $$N_{SYM} = Ceiling((16 + 8 \times Length + 6) / N_{DBPS})$$

   where $N_{DBPS}$ is determined by data rate according to the following table.

   **Rate-Dependent Values**

| Data Rate (Mbps) | Modulation | Coding Rate(R) | Coded Bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

After determining $N_{SYM}$ , the number of input tokens $N_{total}$ can be calculated:

$$N_{total} = (2^{Order} + 2^{Order - 2}) \times 4 + (2^{Order} + GI) \times (N_{SYM} + 1) + Idle$$

where *Idle* is Idle parameter; and GI (GuardInterval parameter) is defined as:

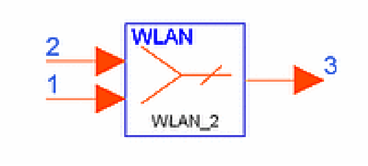
if GuardType=T/32, $GI = 2^{Order -5}$
if GuardType=T/16, $GI = 2^{Order -4}$
if GuardType=T/8, $GI = 2^{Order -3}$
if GuardType=T/4, $GI = 2^{Order -2}$
if GuardType=T/2, $GI = 2^{Order -1}$
if GuardType=UserDefined, GI is determined by GuardInterval.

All input data is output at pins 2 and 3.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_CommCtrl2



**Description 2-input commutator with input particle number control**
**Library WLAN, Multiplex**
**Class SDFWLAN_CommCtrl2**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NumInput1 | number of particles from input 1 | 1 | int | [1, ∞) |
| NumInput2 | number of particles from input 2 | 1 | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | in1 | input 1 | anytype |
| 2 | in2 | input 2 | anytype |

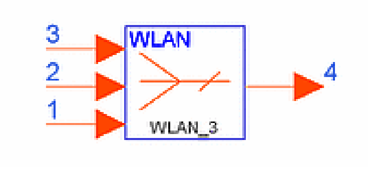
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | output comprised of two inputs | anytype |

## Notes/Equations

1. This model is used to combine two input signals into one.
   NumInput1 from input 1 and NumInput2 from input 2 data particles are combined and output.

# WLAN_CommCtrl3



**Description 3-input commutator with input particle number control**
**Library WLAN, Multiplex**
**Class SDFWLAN_CommCtrl3**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NumInput1 | number of particles from input 1 | 1 | int | [1, ∞) |
| NumInput2 | number of particles from input 2 | 1 | int | [1, ∞) |
| NumInput3 | number of particles from input 3 | 1 | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | in1 | input 1 | anytype |
| 2 | in2 | input 2 | anytype |
| 3 | in3 | input 3 | anytype |

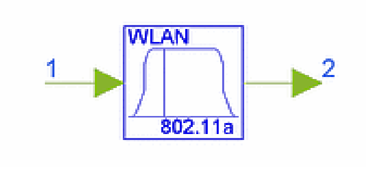
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | output comprised of three inputs | anytype |

## Notes/Equations

1. This model combines three input signals into one. NumInput1 from input 1, NumInput2 from input 2 and NumInput3 from in3 data particles are combined and output.

# WLAN_CosRollWin



**Description Add Cosine-Rolloff windows to Burst signals**
**Library WLAN, Multiplex**
**Class SDFWLAN_CosRollWin**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 100 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| WindowType | type of window: Specification, CosRolloff | Specification | | enum | |
| TransitionTime | the transition time of window function | 100nsec | sec | real | (0, 800nsec] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$ ] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signals | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signals after adding window function | complex |

## Notes/Equations

1. This model is used to add a window function to burst signals.
2. Two types of window functions are provided in this model:
    - *Specification*, according to the 802.11a specification, can be expressed as:

$$W_T(t) = \begin{cases} Sin^2\left(\frac{\pi}{2}(0.5 + t/T_{TR})\right) & (-T_{TR}/2 < t < T_{TR}/2) \\ 1 & (T_{TR}/2 < t < T - (T_{TR}/2)) \\ Sin^2\left(\frac{\pi}{2}(0.5 - (t - T)/T_{TR})\right) & ((T - T_{TR}/2) \le t < T + (T_{TR}/2)) \end{cases}$$

$T_{TR}$ is TransitionTime, which is usually set to 100 nsec.

$W_T(t)$ represents the time-windowing function, depending on the value of the duration parameter T, may extend over more than one period $T_{FFT}$ .

The following figure illustrates extending the windowing function over more than one period and shows smoothed transitions by applying a windowing function.

- *CosRolloff* can be expressed as:

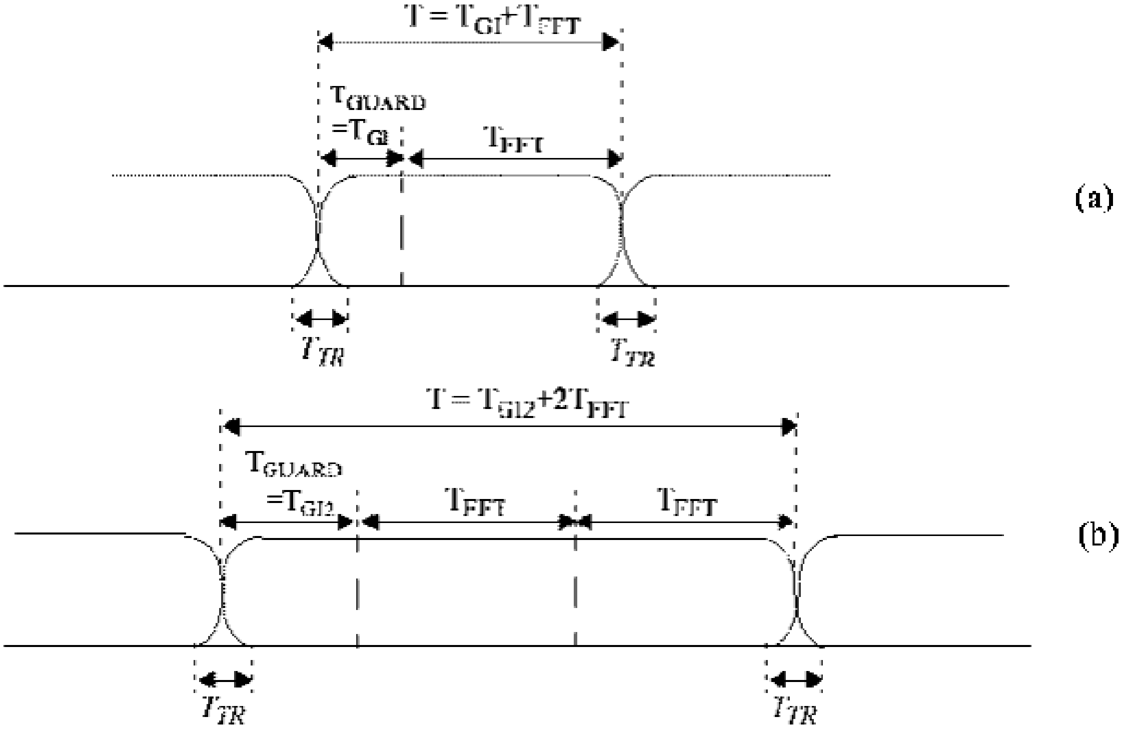
$$windowLength = 2 \times int(T_{TR}/(2 \times (50 \times 10^{-9}/2^{(Order-6)})))-1$$

$$SymbolInterval = \pi/(windowLength)$$

$$W_T[i] = 0.5 - 0.5 \times \cos((i+0.5) \times symbolInterval) \qquad 0 \le i < windowLength$$

$T_{TR}$ is TransitionTime, which is usually set to 100 nsec; Order specifies the FFT size. $W_T[i]$ represents the $i$ th coefficient of discrete time-windowing function.

The following figure illustrates OFDM windowing and cyclic extension.

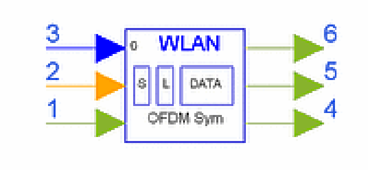


**OFDM frame with Windowing and Cyclic Extension; a=single reception, b=two receptions of FFT period**

## References

1. IEEE Standard 802.11a-1999, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band, 1999.

# WLAN_DemuxBurst



**Description Burst de-multiplexer with frequency offset compensator and guard interval remover**
**Library WLAN, Multiplex**
**Class SDFWLAN_DemuxBurst**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 256 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | | enum | |
| GuardInterval | guard interval defined by user | 16 | | int | [0, $2^{Order}$] |
| TSYM | one OFDM symbol interval | 4e-6 | sec | real | [0, ∞) |
| Idle | padded number of zeros between two bursts | 0 | | int | [0, ∞) |
| FreqOffset | actual frequency offset | 0.0 | Hz | real | (-∞, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received burst signals | complex |
| 2 | index | synchronization index | int |
| 3 | DeltaF | carrier frequency offset | real |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | LPrmbl1 | output first long preamble OFDM signals | complex |
| 5 | LPrmbl2 | output second long preamble OFDM signals | complex |
| 6 | output | output SIGNAL and DATA OFDM signals | complex |

## Notes/Equations

1. This model is used to demultiplex the received burst signals into two long preambles, SIGNAL and DATA OFDM signals, removing the guard interval and the carrier frequency offset.
2. The transmitter transmits burst-by-burst in ADS. The burst sequence is a continuous stream. (The 802.11 burst is transmitted burst-by-burst.) This model includes frequency compensation. The transmitted consecutive bursts are independent.
   The *DeltaF* pin 3 inputs the estimated frequency offset ($\Delta f_i$) of each received burst.

This estimated frequency offset must not effect the next bursts in the frequency compensator. The FreqOffset parameter is set as the actual frequency offset between the transmitter and the receiver; when the ith burst is processed, the actual phase of previous i-1 bursts is calculated and removed. The ith estimated frequency offset ($\Delta f_i$) compensates for the phase in the current burst only.

3. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols, $N_{SYM}$ is:

$$N_{SYM} = Ceiling\ ((16 + 8 \times\ Length + 6) / N_{DBPS})$$

where $N_{DBPS}$ is determined by data rate listed in the following table.

**Rate-Dependent Values**

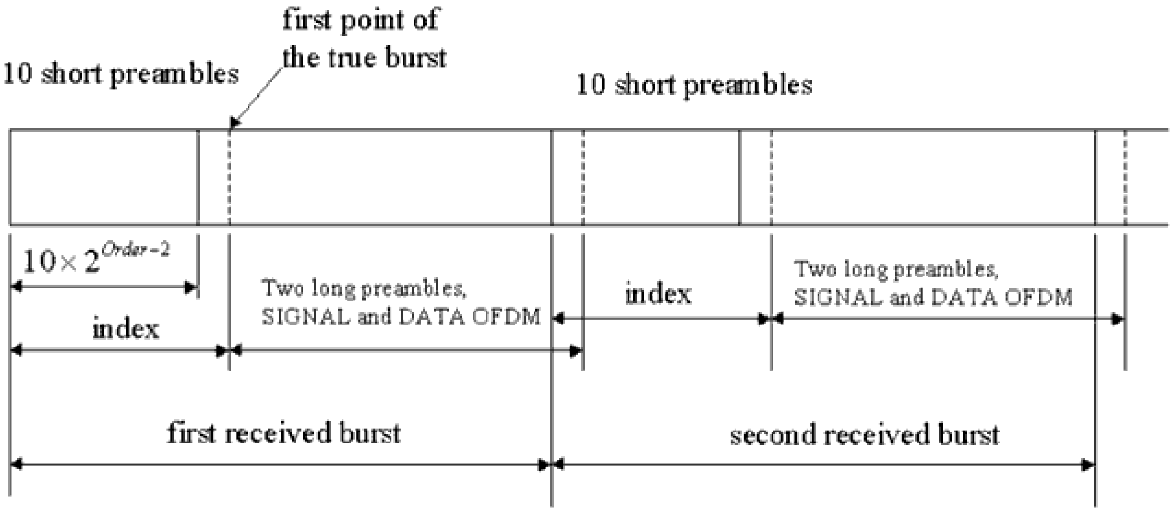
| Data Rate (Mbps) | Modulation | Coding Rate (R) | Coded bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

After determining $N_{SYM}$ , the number of input tokens $N_{total}$ can be calculated:

$$N_{total} = (2^{Order} + 2^{Order-E}) \times (N_{SYM} + 5)$$

The buffer length for input pin is $2 \times N_{total}$ ; $N_{total}$ tokens are fired each operation.

This model determines the first point of the received burst according to the input signal at index pin 2. The following figure illustrates the selection of one burst signal.



### Determining the True Burst

Referring to the previous figure, the *first point of the true burst* includes two long preambles, one SIGNAL OFDM symbol, and $N_{SYM}$ DATA OFDM symbols that are output at the *LPrmbl1* , *LPrmbl2* , and *output* pins, respectively. The frequency offset and the guard interval will be removed after the true burst is determined.
$x_0, x_1, \ldots, x_{N-1}$ are the true burst signals from the first point of the true burst in the previous figure; , $y_0, y_1, \ldots, y_{N-1}$ the phase caused by frequency offset, are removed
where

$$N = (2^{Order} + 2^{Order - 2}) \times (N_{SYM} + 3)$$

Then, the equation is

$$y_k = x_k \times e^{-j2\pi\Delta f(k+L)T}$$

where

$\Delta f_i$ is the frequency offset which is the input at *DeltaF* pin 3,

$L = 10 \times 2^{Order - 2}$

$$T = \frac{3.2 \times 10^{-6}}{2^{Order}}$$

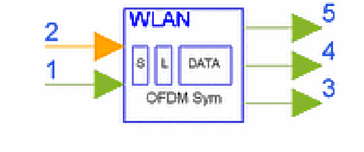
if Order=6, T=50 nsec; if Order=7, T=25 nsec.

After removing the phase caused by frequency offset, the long preambles, SIGNAL and DATA OFDM symbols will be output.
The first long preamble ($2^{Order}$ complex signals) is output at *LPrmbl1* pin; the second long preamble ($2^{Order}$ complex signals) is output at *LPrmbl2* pin; $N_{SYM} + 1$ OFDM symbols (SIGNAL and DATA parts) is output at *output* pin.
This model causes one burst delay.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DemuxBurstNF



**Description Burst de-multiplexer w/guard interval remover, wo/frequency offset compensator**
**Library WLAN, Multiplex**
**Class SDFWLAN_DemuxBurstNF**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |
| Idle | padded number of zeros between two bursts | 0 | int | [0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | received burst signals | complex |
| 2 | index | synchronization index | int |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | LPrmbl1 | output first long preamble OFDM signals | complex |
| 4 | LPrmbl2 | output second long preamble OFDM signals | complex |
| 5 | output | output SIGNAL and DATA OFDM signals | complex |

## Notes/Equations

1. This model is used to demultiplex the received burst signals into long preamble OFDM symbols and SIGNAL and DATA OFDM symbols, and for removing the guard interval. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols, $N_{SYM}$ is:

$$N_{SYM} = Ceiling\ ((16 + 8 \times\ Length + 6)\ /\ N_{DBPS}\ )$$

where $N_{DBPS}$ is determined by data rate, shown in the following table.

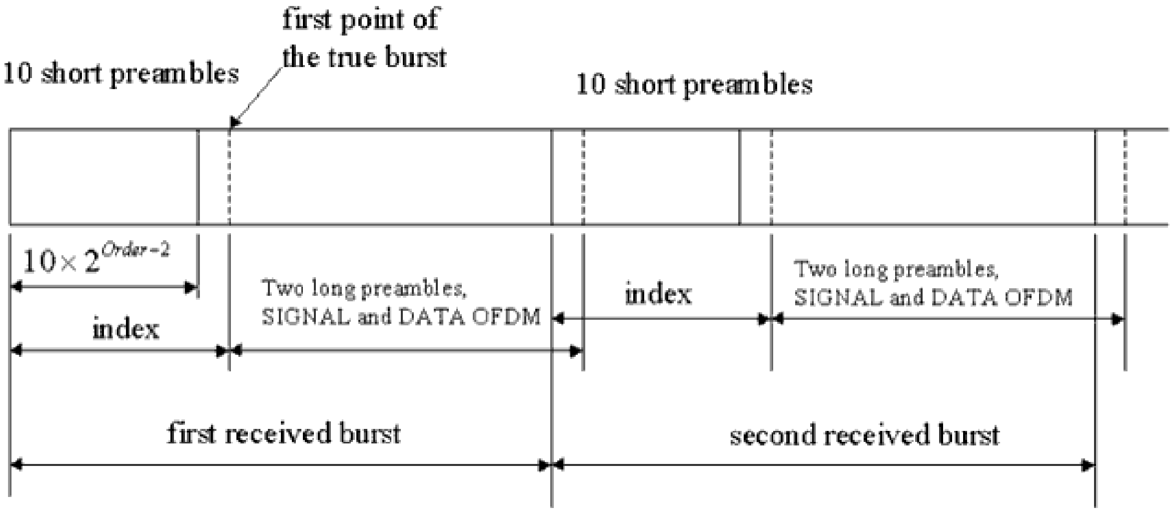
**Rate-Dependent Values**

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

After determining $N_{SYM}$, the number of input tokens $N_{total}$ can be calculated:

$$N_{total} = (2^{Order} + 2^{Order-2}) \times 4 + (2^{Order} + GI) \times (N_{SYM} + 1) + Idle$$

The length of buffer for input pin is 2 × $N_{total}$; $N_{total}$ tokens are fired each operation. According to input signal at the *index* pin, this model can determine the first point of the received burst. The following figure illustrates the selection of one burst signal.



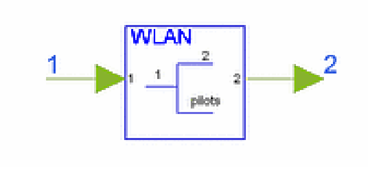
### Determining the True Burst

Referring to the previous figure, the *first point of the true burst* includes two long preambles, one SIGNAL OFDM symbol and $N_{SYM}$ DATA OFDM symbols that are output at *LPrmbl1*, *LPrmbl2* and *output* pins, respectively. The first long preamble ($2^{Order}$ complex signals) is output at *LPrmbl1*; the second long preamble ($2^{Order}$ complex signals) is output at *LPrmbl2*; $N_{SYM}$ + 1 OFDM symbols (SIGNAL and DATA parts) are output at output pin.
This model causes one burst delay.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DemuxOFDMSym



**Description OFDM symbol demultiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_DemuxOFDMSym**

### Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Data | number of input data in one OFDM symbol | 48 | int | {48} |

### Pin Inputs

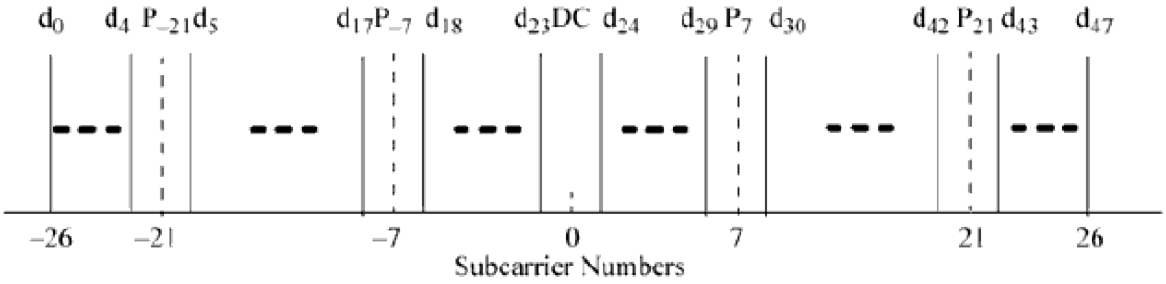
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | equalized signals before de-multiplexer | complex |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | data | OFDM demodulation data | complex |

### Notes/Equations

1. The model is used to demultiplex IEEE 802.11a OFDM symbol (such as QPSK, 16-QAM, and 64-QAM modulation) into data and pilots.
2. Subcarrier frequency allocation is illustrated in the following figure. The 52 complex inputs are composed of 48 complex data and 4 pilot signals that are demultiplexed into 48 complex data and 4 pilots according to the following figure; the pilots are not output.



**Subcarrier Frequency Allocation**

Output data are *y* 0, *y* 1, ... , *y* 47; input signals are *x* 0, *x* 1, ... , *x* 51. The equations are:

$y_i = x_{i\,\_}\ \_{\mathrm{i}} = 0, 1, 2, 3, 4$

$y_i = x_{i+1\,\_}\ \_{\mathrm{i}} = 5, \ldots , 17$

$y_i = x_{i+2\,\_}\ \_{\mathrm{i}} = 18, \ldots , 29$

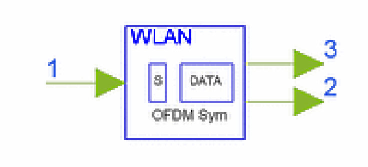
$y_i = x_{i+3\,\_}\ \_{\mathrm{i}} = 30, \ldots , 42$

$y_i = x_{i+4\,\_}\ \_{\mathrm{i}} = 43, 44, 45, 46, 47$

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DemuxSigData



**Description SIGNAL and DATA signals demultiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_DemuxSigData**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | equalized signals | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | SIGNAL | output SIGNAL signal | complex |
| 3 | DATA | output DATA signal | complex |

## Notes/Equations

1. This model is used to demultiplex the received equalized input signal into one SIGNAL OFDM symbol and $N_{SYM}$ DATA OFDM symbols.

   The Length and Rate parameters are used to determine the number of complex signals in one burst. The number of DATA OFDM symbols $N_{SYM}$ is:

   $$N_{SYM} = Ceiling\left((16 + 8 \times \ Length + 6) / N_{DBPS}\right)$$

   where $N_{DBPS}$ is determined by data rate given in the following table.

   The SIGNAL OFDM symbol is output at SIGNAL pin 2, DATA OFDM symbols are output at DATA pin 3.

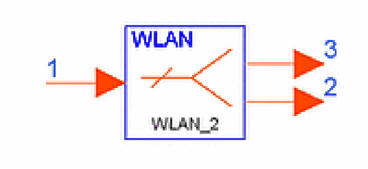
   ### Rate-Dependent Values

| Data Rate (Mbps) | Modulation | Coding Rate (R) | Coded Bits per Subcarrier (NBPSC) | Coded Bits per OFDM Symbol (NCBPS) | Data Bits per OFDM Symbol (NDBPS) |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_DistCtrl2



**Description 2-output distributor with output particle number control**
**Library WLAN, Multiplex**
**Class SDFWLAN_DistCtrl2**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NumOutput1 | number of particles directed to output 1 | 1 | int | [1, ∞) |
| NumOutput2 | number of particles directed to output 2 | 1 | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input to be distributed over the two outputs | anytype |

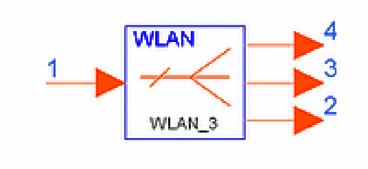
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | out1 | output 1 | anytype |
| 3 | out2 | output 2 | anytype |

## Notes/Equations

1. This model is used to distribute one data stream to two outputs.
NumOutput1 and NumOutput2 data particles are distributed to output 1 and output 2, respectively.

# WLAN_DistCtrl3



**Description 3-output distributor with output particle number control**
**Library WLAN, Multiplex**
**Class SDFWLAN_DistCtrl3**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NumOutput1 | number of particles directed to output 1 | 1 | int | [1, ∞) |
| NumOutput2 | number of particles directed to output 2 | 1 | int | [1, ∞) |
| NumOutput3 | number of particles directed to output 3 | 1 | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input to be distributed over the three outputs | anytype |

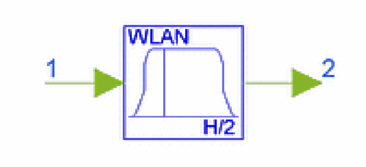
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | out1 | output 1 | anytype |
| 3 | out2 | output 2 | anytype |
| 4 | out3 | output 3 | anytype |

## Notes/Equations

1. This model is used to distribute one data stream to three outputs. NumOutput1, NumOutput2, and NumOutput3 data particles are distributed to output 1, output 2, and output 3, respectively.

# WLAN_H2CosRollWin



**Description Add Cosine-Rolloff windows to Burst signals**
**Library WLAN, Multiplex**
**Class SDFWLAN_H2CosRollWin**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| BurstType | type of burst type: Broadcast, Downlink, UplinkS, UplinkL, Directlink | Broadcast | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signals | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | signals after adding Cosine-Rolloff windows | complex |

## Notes/Equations

1. This model is used to add cosine-rolloff windowing to HIPERLAN/2 PHY burst signals.
2. The cosine-rolloff windowing function can be expressed as follows.
   symbolInterval=π/windowLength;
   for (i=0; i<windowLength; i++)
   $W_T[i] = 0.5 - 0.5 \times \cos((i + 0.5) \times symbolInterval)$

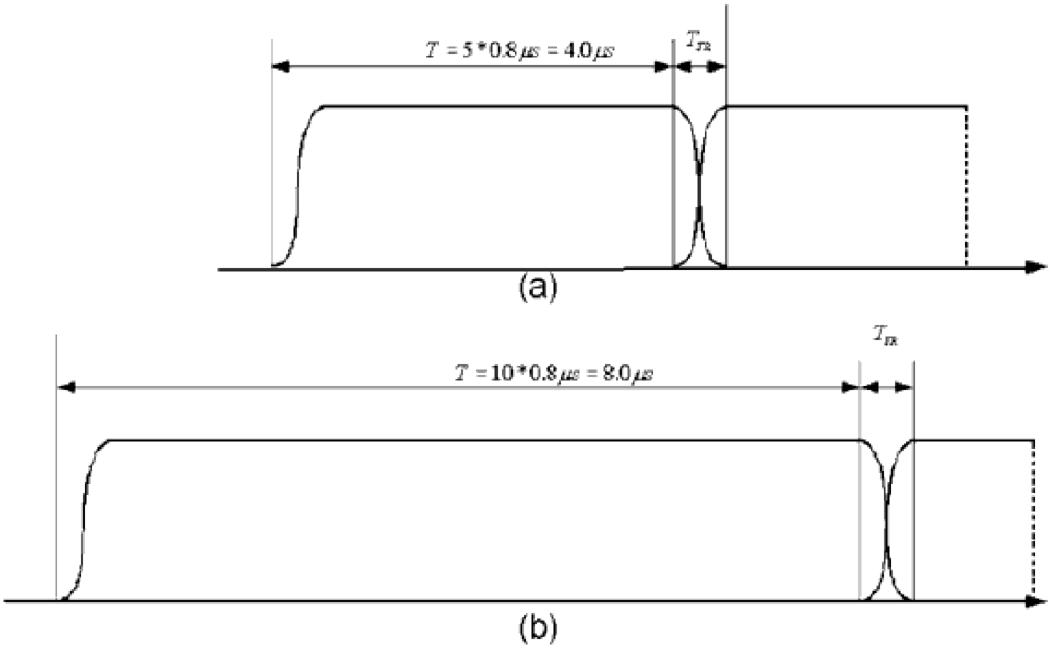
   $W_T[i]$ represents the ith coefficient of cosine-rolloff windowing, the width of window is determined by the Order parameter that determines the size of FFT. If Order=6, windowLength is 1; if Order=7, windowLength is 3; if Order=8, windowLength is 7.
   Windowing is determined by the type of burst. The windowing modes based on symbols are illustrated in the following figure. According to reference[1] 5.7 PHY bursts:

   section 1 and section 5 use the windowing mode illustrated in the following figure (a)
   section 2 uses the widowing mode illustrated in the following figure (d), with the guard interval equal to 1/4 of $T_{FFT}$

   section 3, 4, 6, and 8 use the windowing mode illustrated in the following

figure (c)
section 7 uses windowing mode illustrated in the following figure (b)
data symbols use the windowing mode illustrated in the following figure (d), and parameter $T_{Guard}$ is set by the GuardType parameter defined by the designer.

The parameter $T_{TR}$ is implemented in order to smooth the transitions between the consecutive subsections. This creates a small overlap, of duration $T_{TR}$, as shown in the following figure. In our cosine-rolloff window design, the $T_{TR}$ is approximately 100 nsec. Smoothing the transition is required in order to reduce the spectral side-lobes of the transmitted waveform. However, the binding requirements are the spectral mask and modulation accuracy requirements, as detailed in reference[1] 5.8.2.2 and 5.9. Time domain windowing, as described here, is just one way to achieve those objectives. Other methods, such as frequency domain filtering, can be used to achieve the same goal; therefore, the transition shape and duration of the transition are informative parameters.

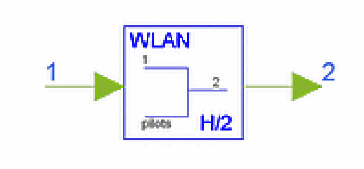


**OFDM Frame with Cyclic Extension and Windowing: (a) 5 short symbols Type A; (b) 10 short symbols Type B (c) 2 long symbols; (d) data symbols**

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_H2MuxOFDMSym



**Description OFDM symbol multiplexer for HiperLAN2**
**Library WLAN, Multiplex**
**Class SDFWLAN_H2MuxOFDMSym**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Data | number of input data in one OFDM symbol | 48 | int | {48} |
| Phase | initial phase of pilots | 0 | int | [0, 126] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | data | data input | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | OFDM symbol data output | complex |

## Notes/Equations

1. The model is used to multiplex data, pilots into the HiperLAN/2 OFDM symbol.
2. The stream of complex numbers is divided into groups of $N_{sd}$ = 48 complex numbers $d_{k,n}$ , where k is the subcarrier of OFDM symbol n.

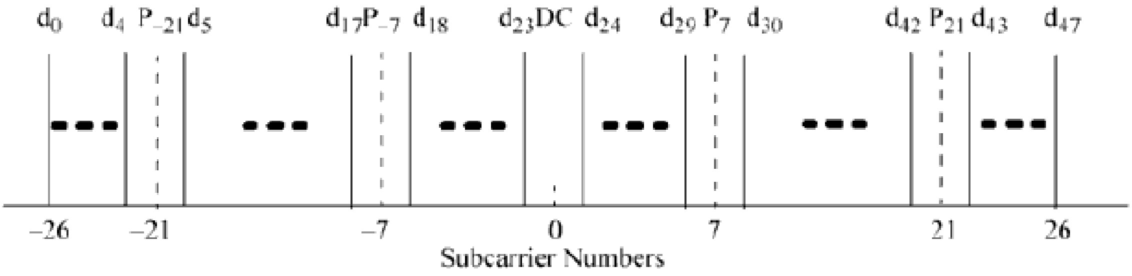
   The contribution of the pilot subcarriers for the nth OFDM symbol is produced by Fourier transform of sequence $P$ , given by

   $$P_{-26, 26} = \{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,1,0,0,0,0,0,0,0, 0,0,0,0,0,1,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0\}$$

   The polarity of the pilot subcarriers is controlled by the sequence $p_n$ , which is a cyclic extension of the 127 elements sequence and is given by

   $$p_{0...126} = \{1,1,1,1,-1,-1,-1,1, -1,-1,-1,-1, 1,1,-1,1, -1,-1,1,1, -1,1,1,-1, 1,1,1,1, 1,1,-1,1, 1,1,-1,1, 1,-1, -1,1, 1,1,-1,1, -1,-1,-1,1, -1,1,-1,-1, 1, -1,-1,1, 1,1,1,1, -1,-1,1,1, -1,-1,1,-1, 1,-1, 1,1, -1,-1, -1, 1, 1,-1,-1,-1, -1, 1,-1, -1, 1,-1,1,1, 1,1,-1,1, -1,1,-1,1, -1,-1, -1,-1, -1,1,-1,1, 1,-1,1,-1, 1,1,1,-1, -1, 1,-1,-1, -1,1, 1,1, -1,-1,-1,-1, -1,-1,-1\}$$

Each sequence element is used for one OFDM symbol. The Phase parameter controls the start position of the cyclic sequence.
Subcarrier frequency allocation is illustrated in the following figure.



**Subcarrier Frequency Allocation**

This model combines 48 input complex data and four pilots into an OFDM symbol. Pilot positions are -21, -7, 7 and 21; these pilots are

$P -21 = p_n$

$P -7 = p_n$

$P\ 7 = p_n$

$P\ 21 = - p_n$

where n represent nth OFDM symbol in the each Burst.
Data and pilots are combined according to subcarrier allocation; output data $y\ 0$, $y\ 1$, ... , $y\ 51$ equations are:

$y_i = d_i \quad i = 0, 1, 2, 3, 4$

$y_5 = P -21$

$y_i + 1 = d_i \quad i = 5, ..., 17$

$y_{19} = P -7$

$y_i + 2 = d_i \quad i = 18, ..., 29$

$y_{32} = P\ 7$

$y_i + 3 = d_i \quad i = 30, ..., 42$

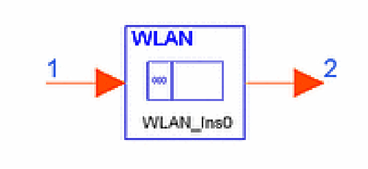
$y_{46} = P\ 21$

$y_i + 4 = d_i \quad i = 43, 44, 45, 46, 47$

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.

# WLAN_InsertZero



**Description Insert zero to data with input particle number control**
**Library WLAN, Multiplex**
**Class SDFWLAN_InsertZero**

## Parameters

| Name | Description | Default | Sym | Type | Range |
|---|---|---|---|---|---|
| NumInsert | number of zeros inserted before input data | 0 | N | int | [0, ∞) |
| NumInput | number of particles from input | 1 | M | int | (0, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input | anytype |

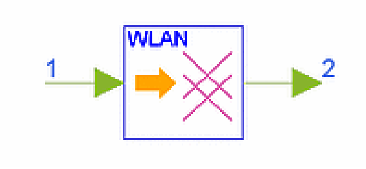
## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | output after zero inserted | anytype |

## Notes/Equations

1. This component inserts N zeros before M input data, thus adding idle time between two bursts.

# WLAN_LoadIFFTBuff



**Description Data stream loader into IFFT buffer**
**Library WLAN, Multiplex**
**Class SDFWLAN_LoadIFFTBuff**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Order | IFFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | transmitted signal before IFFT | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | IFFT input signal, zero padded | complex |

## Notes/Equations

1. The model is used to load transmission data into the IFFT buffer.
2. The Order parameter is the order of FFT. It must satisfy

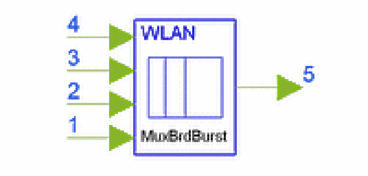
   $2^{Order}$ Š *Carriers*
3. Assume *x* (0), *x* (1), ... , *x* (51) are input signals, *y* (0), *y* (1), ... , *y* ( *M* -1) are output signals, where, M = $2^{Order}$ ; data loading is:

$$\begin{array}{ll} y(i) = x(26+i) & i = 1, \ldots, 26 \\ y(i) = 0 & i = 0, 27, \ldots, M-26-1 \\ y(i) = x(i-M+26) & i = M-26, \ldots, M-1 \end{array}$$

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.
2. ETSI TS 101 475 v1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," April, 2000.

# WLAN_MuxBrdBurst



**Description Broadcast burst multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxBrdBurst**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmblA | short preamble A | complex |
| 2 | SPrmblB | short preamble B | complex |
| 3 | LPrmbl | long preamble | complex |
| 4 | input | OFDM symbols of broadcast PDU train | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 5 | output | broadcast burst signal | complex |

## Notes/Equations

1. This model is used to multiplex short and long preambles, broadcast PDU train or broadcast PDU train plus FCH and ACH PDU train OFDM symbols into a broadcast burst. Guard interval insertion is implemented.
2. The broadcast burst consists of a preamble $t_{preamble}$ = 16.0 μsec and a payload section

   $N_{SYM} \times T_S$

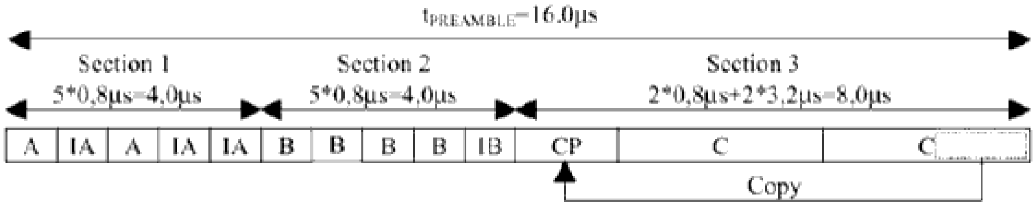
   where

   $N_{SYM}$ is the number of OFDM symbols in the payload section, set in the NSYM parameter

   $T_S$ is the OFDM symbol interval

   ( $T_S$ = 4.0 μsec if GuardType=T_4, $T_S$ = 3.2 μsec if GuardType=T_8).

The broadcast burst preamble structure is illustrated in the following figure.

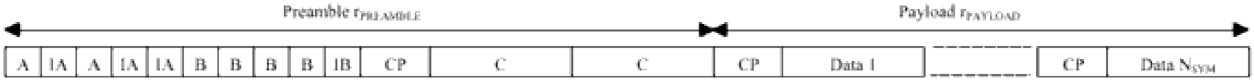


**Broadcast Burst Preamble**

The broadcast burst preamble sections illustrated in the previous figure are described here. The term *short OFDM symbol* refers to length that is 16 samples instead of a regular OFDM symbol of 64 samples used in HiperLAN/2 systems.

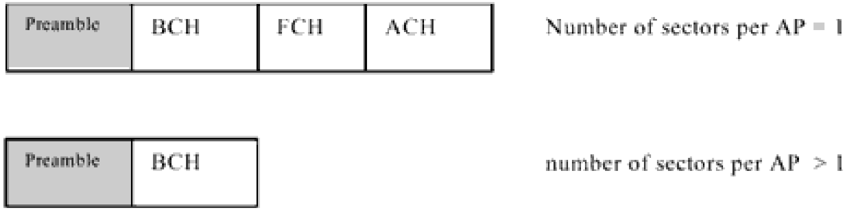
> Section 1 consists of 5 specific short OFDM symbols denoted A and IA. The first 4 short OFDM symbols (A, IA, A, IA) constitute a regular OFDM symbol consisting of 12 loaded sub-carriers (±2, ±6, ±10, ±14, ±18, and ±22) given by the frequency-domain sequence SA
>
> SA-26...26 = $\sqrt{13/6}$ × {0,0,0,0,-1+j,0,0,0,1+j,0,0, 0,1-j,0,0,0,-1-j,0,0,0,-1+j, 0, 0,0,-1-j, 0, 0, 0, -1+j, 0, 0, 0, -1-j, 0, 0, 0, -1+j, 0, 0, 0, -1-j, 0, 0, 0, 1-j, 0, 0, 0, 1+j, 0, 0, 0, 0}
> The last short symbol IA is a repetition of the preceding 16 time-domain samples.
> Section 2 consists of 5 specific short OFDM symbols denoted B and IB. The first 4 short OFDM symbols (B, B, B, B) constitute a regular OFDM symbol consisting of 12 loaded sub-carriers (±4, ±8, ±12, ±16, ±20, and ±24) given by the frequency-domain sequence SB
>
> SB-26...26 = $\sqrt{13/6}$ × {0,0,1+j, 0, 0, 0,-1-j, 0, 0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0}
> The last short symbol IB is a sign-inverted copy of the preceding short symbol B, i.e. IB=-B.
> Section 3 consists of two OFDM symbols (C) of normal length preceded by a cyclic prefix (CP) of the symbols. All 52 sub-carriers are in use and are modulated by the elements of the frequency-domain sequence SC given by SC-26...26 = {1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1,-1,-1,1,1,-1, 1,-1, 1,-1, -1,-1,-1, -1,1, 1,-1,-1,1,-1,1,-1, 1,1, 1,1}
> The cyclic prefix CP is a copy of the $2^{(Order-1)}$ last samples of the C symbols and is thus double in length compared to the cyclic prefix of normal data symbols.

The broadcast burst formed by concatenating the above described preamble with the data payload is illustrated in the following figure.



**PHY Burst Structure for Broadcast Burst**

The broadcast PDU train format, based on the number of sectors the AP uses (single or multiple), is illustrated in the following figure. In the case of multiple sectors, each BCH is transmitted using an individual broadcast PDU train.



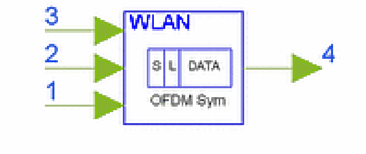
**Broadcast PDU Train Formats**

The number of OFDM symbols per transport channels is shown in the following table.

**Number of OFDM Symbols per Transport Channel (Excluding Physical Layer Preambles)**

| PHY mode | BCH, 15oct. | FCH, 27oct. | ACH, 9oct. | SCH, 9oct. | LCH, 54oct. | RCH, 9oct. |
|---|---|---|---|---|---|---|
| BPSK, code rate=1/2 | 5 | 9 | 3 | 3 | 18 | 3 |
| BPSK, code rate=3/4 | | | | 2 | 12 | |
| QPSK, code rate=1/2 | | | | | 9 | |
| QPSK, code rate=3/4 | | | | 1 | 6 | |
| 16QAM, code rate=9/16 | | | | | 4 | |
| 16QAM, code rate=3/4 | | | | | 3 | |
| 64QAM, code rate=3/4 | | | | | 2 | |

# WLAN_MuxBurst



**Description Burst multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxBurst**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmbl | short preamble | complex |
| 2 | LPrmbl | long preamble | complex |
| 3 | input | SIGNAL and DATA OFDM symbols | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | burst signal | complex |

## Notes/Equations

1. This model is used to multiplex short and long preambles and SIGNAL and DATA OFDM symbols into a burst. The guard interval insertion and the window function are also implemented. The burst is the time of the PPDU frame format.
2. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols (DATA part), $N_{SYM}$ as follows

$$N_{SYM} = Ceiling\ ((16 + 8 \times Length + 6) / N_{DBPS})$$

where $N_{DBPS}$ is determined by data rate, shown in the following table.

**Rate-Dependent Values**

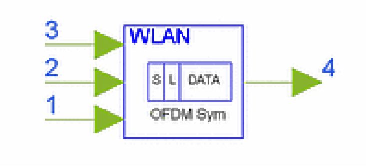
| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

The number of input tokens for input *SPrmbl* pin 1, *LPrmbl* pin 2 and *input* pin 3 are $2^{Order}$, $2^{Order}$, $2^{Order} \times (N_{SYM} + 1)$, respectively. The number of output tokens is $(2^{Order}, 2^{Order-2}) \times (N_{SYM} + 5)$, which includes all complex signals in one burst.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_MuxBurstNW



**Description Burst multiplexer without window function**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxBurstNW**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |
| GuardType | type of guard interval: T/2, T/4, T/8, T/16, T/32, UserDefined | T/4 | enum | |
| GuardInterval | guard interval defined by user | 16 | int | $[0, 2^{Order}]$ |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmbl | short preamble | complex |
| 2 | LPrmbl | long preamble | complex |
| 3 | input | SIGNAL and DATA OFDM symbols | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | burst signal | complex |

## Notes/Equations

1. This model multiplexes the short preambles, long preambles, and SIGNAL and DATA OFDM symbols into a burst. Guard interval insertion is also implemented. The burst is the PPDU frame format time.
2. Length and Rate parameters determine the number of complex signals in one burst. The number of OFDM symbols (DATA part), $N_{SYM}$ is:

$$N_{SYM} = Ceiling\ ((16 + 8 \times\ Length + 6) / N_{DBPS})$$

where $N_{DBPS}$ is data rate given in the following table.

### Rate-Dependent Parameters

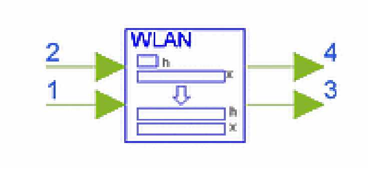
| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

$2^{Order}$, $2^{Order}$, $2^{Order} \times (N_{SYM} + 1)$ tokens are input at SPrmbl pin 1, LPrmbl pin 2, and input pin 3, respectively.
$(2^{Order}, 2^{Order-2}) \times (N_{SYM} + 5)$ tokens are output, which includes all complex signals in one burst.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_MuxDataChEst



**Description Data and estimated channel impluse response multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxDataChEst**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 256 | int | [1, 4095] |
| Rate | date rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | input signals from FFT | complex |
| 2 | Coef | input estimated channel impulse response | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | output signals | complex |
| 4 | chl | output estimated channel impulse response | complex |

## Notes/Equations

1. This model is used to multiplex the data signal and estimated channel impulse response.
   There is only one OFDM estimated channel impulse response and several OFDM DATA or SIGNAL signals. WLAN_PhaseTrack or WLAN_RmvNullCarrier models work per OFDM symbol. In order to match WLAN_PhaseTrack or WLAN_RmvNullCarrier, this model is needed in the receiver.
2. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols (DATA part), $N_{SYM}$ are calculated as follows.

   $$N_{SYM} = Ceiling\ ((16 + 8 \times\ Length + 6) / N_{DBPS})$$

   where $N_{DBPS}$ is determined by data rate listed in the following table.

   **Rate-Dependent Values**

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

So, there are $N_{SYM} + 1$ OFDM symbols in one burst.

The signal at *input* pin 1 is output directly at *output* pin 3. The number of signals input and output is $2^{Order} \times (N_{SYM} + 1)$.

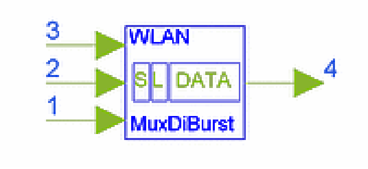
The number of signals at input *Coef* pin 2 is 52 (the number of active carriers in one OFDM symbol). The WLAN_PhaseTrack and WLAN_RmvNullCarrier models can be used after WLAN_MuxDataChEst, both models using one OFDM symbol. The number of output signals at *chl* pin 4 is $52 \times (N_{SYM} + 1)$, which is generated by repeating the *Coef* input signal 52 times. It is implemented as follows:

```
for (k=0;k<NSYM+1;k++)
 {
   for (i=0;i<52;i++)
   {
     in1 = Coef%(52-1-i);
     chl%(52*(NSYM+1-k)-1-i)<<in1;
   }
 }
```

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_MuxDiBurst



**Description Direct link burst multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxDiBurst**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmblB | short preamble | complex |
| 2 | LPrmbl | long preamble | complex |
| 3 | input | OFDM symbols of Direct link PDU train | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | Direct link burst signal | complex |

## Notes/Equations

1. This model is used to multiplex short and long preambles and a direct link PDU train into a direct link burst signal. Guard interval insertion is implemented.
2. Direct link burst consists of a preamble $t_{preamble}$ = 16.0 μsec and a payload section

   $N_{SYM} \times T_S$

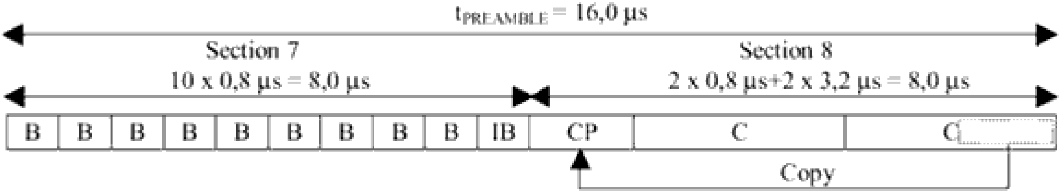
   where

   $N_{SYM}$ is the number of OFDM symbols in the payload section, set in the NSYM parameter

   $T_S$ is the OFDM symbol interval

   ( $T_S$ = 4.0 μsec if GuardType=T_4, $T_S$ = 3.2 μsec if GuardType=T_8).

   The direct link burst preamble structure is illustrated in the following figure.

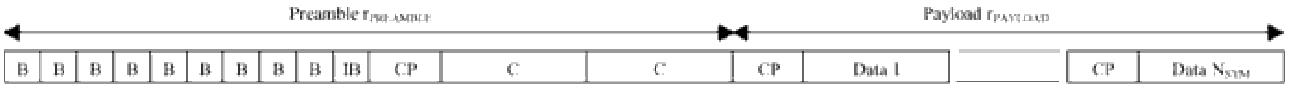


**Direct Link Burst Preamble**

The direct link burst preamble sections illustrated the previous figure are described here. The term *short OFDM symbol* refers to its length that is $2^{Order-2}$ samples instead of a regular OFDM symbol of $2^{Order}$ samples used in HiperLAN/2 systems.

> Section 7 consists of 10 specific short OFDM symbols denoted B and IB. The first 4 short OFDM symbols in this section (B, B, B, B) constitute a regular OFDM symbol consisting of 12 loaded sub-carriers (±4, ±8, ±12, ±16, ±20, and ±24) given by the frequency sequence SB:
>
> $SB_{-26, \ldots, -26} = \sqrt{13/6} \times$ {0,0,1+j,0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0, 0, 0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0}
>
> The last short symbol in section 7 (IB) is a sign-inverted copy of the preceding short symbol B, i.e. IB = -B.
>
> Section 8 consists of two OFDM symbols (C) of normal length preceded by a cyclic repetition (CP) of the symbols. All 52 sub-carriers are used and are modulated by the elements of the frequency-domain sequence SC given by:
>
> $SC_{-26, \ldots, -26}$ = {1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1,-1,-1, 1,1,-1, 1, -1, 1, -1,-1, -1,-1, -1, 1,1, -1,-1, 1, -1,1,-1,1, 1, 1,1}
>
> The cyclic repetition CP is a copy of the last $2^{Order-1}$ samples of the C symbols and is thus double in length compared to the cyclic prefix of the normal data symbols.

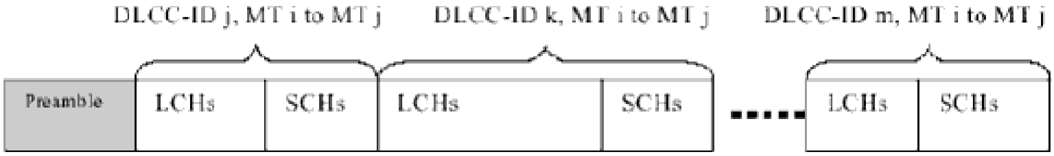
The direct link burst, formed by concatenating the above described preamble with the data payload, is illustrated in the following figure.



**PHY Burst Structure for Direct Link Burst**

One preamble must be added at the beginning of each direct link PDU train; see the following figure. The preamble of the direct link PDU train must have a length of 4 OFDM symbols; see reference [1].

A direct link PDU train must consist of all LCHs and SCHs belonging to the same pair of source and destination MAC IDs. A set of SCHs and LCHs is granted for each DLCC by one RG. An MT cannot receive more than one direct link PDU train containing UDCHs, DCCHs, and LCCHs per MAC frame per source MAC ID, that is, all corresponding DLCCs must be grouped in a single PDU train. A receiver can receive the RBCH, UMCHs, and UBCHs from the same transmitter in separate PDU trains.



**Direct Link PDU Train**

The number of OFDM symbols per transport channel is shown in the following table.

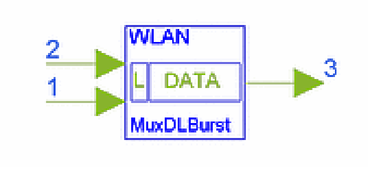
**Number of OFDM Symbols per Transport Channel (Excluding Physical Layer Preambles)**

| PHY mode | BCH, 15oct. | FCH, 27oct. | ACH, 9oct. | SCH, 9oct. | LCH, 54oct. | RCH, 9oct. |
|---|---|---|---|---|---|---|
| BPSK, code rate=1/2 | 5 | 9 | 3 | 3 | 18 | 3 |
| BPSK, code rate=3/4 | | | | 2 | 12 | |
| QPSK, code rate=1/2 | | | | | 9 | |
| QPSK, code rate=3/4 | | | | 1 | 6 | |
| 16QAM, code rate=9/16 | | | | | 4 | |
| 16QAM, code rate=3/4 | | | | | 3 | |
| 64QAM, code rate=3/4 | | | | | 2 | |

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_MuxDLBurst



**Description Downlink burst multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxDLBurst**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | LPrmbl | long preamble | complex |
| 2 | input | OFDM symbols of Downlink PDU train | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | Downlink burst signal | complex |

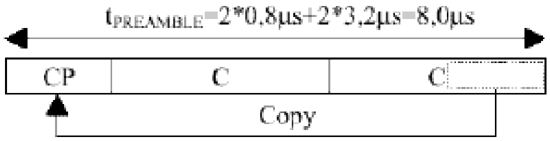
## Notes/Equations

1. This model is used to multiplex long preambles and downlink PDU train OFDM symbols into a downlink burst. Guard interval insertion is implemented.
2. The downlink burst consists of a preamble of length $t_{preamble}$ = 8.0 μsec and a payload section of length $N_{SYM} \times T_S$,

   where

   $N_{SYM}$ is the number of OFDM symbols in the payload section, set in the NSYM parameter
   $T_S$ is the OFDM symbol interval
   ( $T_S$ = 4.0 μsec if GuardType=T_4, $T_S$ = 3.6 μsec if GuardType=T_8).

   The downlink burst preamble structure is illustrated in the following figure.

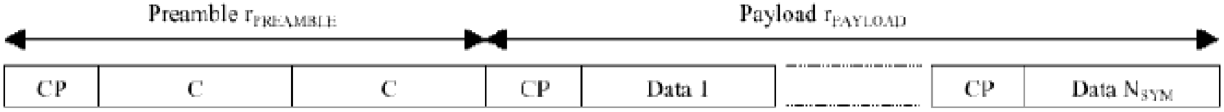


**Downlink Burst Preamble**

The downlink burst preamble is equal to Section 3 of the broadcast burst preamble. It is composed of two OFDM symbols (C) of normal length preceded by a cyclic repetition (CP) of the symbols. All the 52 sub-carriers are in use and are modulated by elements of the frequency-domain sequence SC given by

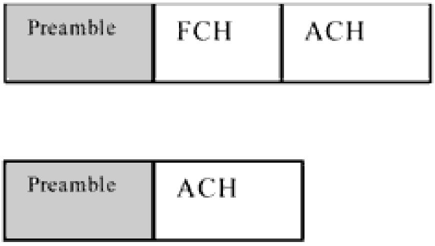
$SC_{-26, ..., 26}$ =
{1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0,
1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1}

The cyclic prefix CP is a copy of the $2^{Order-1}$ last samples of the C symbols and is thus double in length compared to the cyclic prefix of the normal data symbols.
The downlink burst formed by concatenating the above described preamble with the data payload is illustrated in the following figure.



**PHY Burst Structure for Downlink Burst**

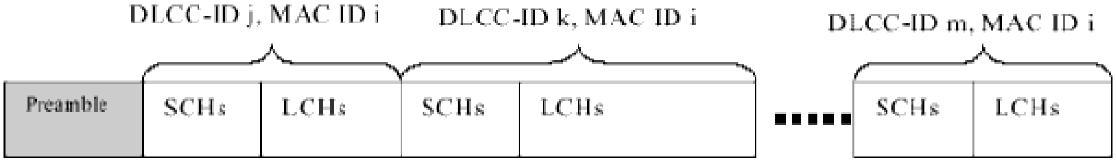
The downlink PDU train is mapped onto the downlink burst when Number of sectors per AP=1; the FCH-and-ACH PDU train is mapped onto the Downlink burst when Number of sectors per AP>1.
One preamble must be added in the beginning of each FCH-and-ACH PDU train if multiple sectors are used per AP.
The preamble of the FCH-and-ACH PDU train must be 2 OFDM symbols (reference [1]). Possible FCH-and-ACH PDU trains are shown in the following figure. The upper drawing shows the case where an FCH is present, whereas the length of the FCH is zero in the lower drawing.



**Possible FCH-and-ACH PDU Trains**

One preamble is added at the beginning of each downlink PDU train, see the following figure. The preamble of the downlink PDU train must have a length of 2

OFDM symbols (reference [1] ).



**Possible downlink PDU Trains**

A set of SCHs and LCHs is granted for each DLCC by one RG. An MT cannot receive more than one downlink PDU train containing UDCHs, the DCCH and LCCHs per MAC frame; that is, all corresponding DLCCs must be grouped in a single PDU train. RBCH, UMCHs and UBCHs are received in separate PDU trains.
The following table lists the number of OFDM symbols per transport channel.

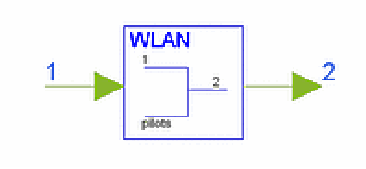
**Number of OFDM Symbols per Transport Channel (Excluding Physical Layer Preambles)**

| **PHY mode** | **BCH, 15oct.** | **FCH, 27oct.** | **ACH, 9oct.** | **SCH, 9oct.** | **LCH, 54oct.** | **RCH, 9oct.** |
|---|---|---|---|---|---|---|
| BPSK, code rate=1/2 | 5 | 9 | 3 | 3 | 18 | 3 |
| BPSK, code rate=3/4 | | | | 2 | 12 | |
| QPSK, code rate=1/2 | | | | | 9 | |
| QPSK, code rate=3/4 | | | | 1 | 6 | |
| 16QAM, code rate=9/16 | | | | | 4 | |
| 16QAM, code rate=3/4 | | | | | 3 | |
| 64QAM, code rate=3/4 | | | | | 2 | |

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_MuxOFDMSym



**Description OFDM symbol multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxOFDMSym**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Carriers | number of carriers in one OFDM symbol | 52 | int | {52} |
| Data | number of input data in one OFDM symbol | 48 | int | {48} |
| Phase | initial phase of pilots | 126 | int | [0, 126] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | data | data input | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | output | OFDM symbol data output | complex |

## Notes/Equations

1. This model is used to multiplex data and pilots into the IEEE 802.11a OFDM symbol.
2. The stream of complex numbers is divided into groups of $N_{sd}$ = 48 complex numbers. This is denoted by writing the complex number $d_{k,n}$, which corresponds to subcarrier k of OFDM symbol n.
   The contribution of the pilot subcarriers for the nth OFDM symbol is produced by Fourier transform of sequence P, given by

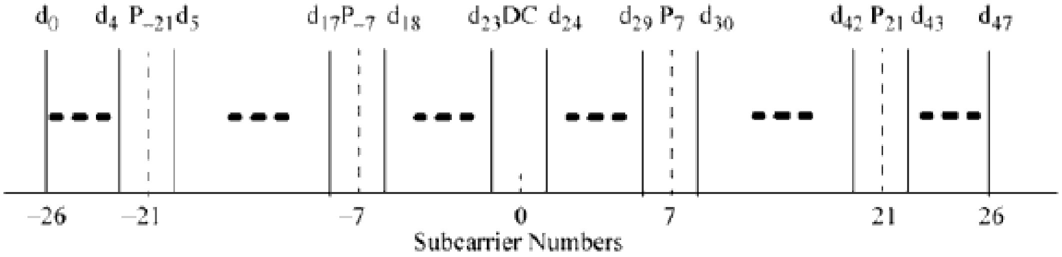
   $P_{-26, 26}$ = {0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
   0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0}

   The polarity of the pilot subcarriers is controlled by sequence $p_n$
   which is a cyclic extension of the 127 elements sequence and is given by

   $P_{0 ... 126}$ =
   {1,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,1,1,-1,1,-1,-1,1,1,-1,1,1,-1,1,1,1,1, 1,1,
   -1,1,1,1,-1,1, 1,-1,-1,1, 1,1,-1,1, -1,-1,-1,1, -1,1,-1,-1, 1,-1,-1,1, 1,1,

1,1, -1,-1,1,1,-1,-1,1,-1, 1,-1,1,1, -1,-1,-1,1, 1,-1,-1,-1, -1,1,-1,-1, 1,
-1,1,1, 1,1,-1,1,-1,1,-1,1,-1,-1,-1,-1, -1,1,-1,1, 1,-1,1,-1, 1,1,1,-1,
-1,1,-1,-1, -1,1,1,1, -1,-1,-1,-1,-1,-1,-1}

Each sequence element is used for one OFDM symbol. The first element $P_0$ multiplies the pilot subcarriers of the SIGNAL symbol, while the elements from $P_1$ are used for DATA symbols.
Subcarrier frequency allocation is shown in the following figure.



**Subcarrier Frequency Allocation**

This model combines 48 input complex data and four pilots into an OFDM symbol.
Pilot positions are -21, -7, 7 and 21. These pilots are

$P_{-21} = p_n$

$P_{-7} = p_n$

$P_7 = p_n$

$P_{21} = -p_n$

where n represents nth OFDM symbols in the Burst.
Data and pilots are combined according to subcarrier allocation in the previous figure, output data $y_0, y_1, \ldots, y_{51}$ equations are:

$y_i = d_i \; i = 0, 1, 2, 3, 4$

$y_5 = P_{-21}$

$y_i + 1 = d_i \; i = 5, \ldots, 17$

$y_{19} = P_{-7}$

$y_i + 2 = d_i \; i = 18, \ldots, 29$

$y_{32} = P_7$

$y_i + 3 = d_i \; i = 30, \ldots, 42$

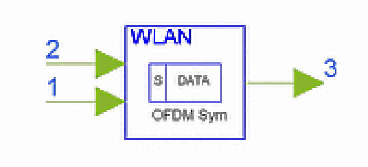
$y_{46} = P_{21}$

$y_i + 4 = d_i \; i = 43, 44, 45, 46, 47$

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_MuxSigData



**Description SIGNAL and DATA multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxSigData**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SIGNAL | SIGNAL signals | complex |
| 2 | DATA | DATA signals | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | output signals | complex |

## Notes/Equations

1. This model is used to multiplex the SIGNAL and DATA signals into the output signals. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols (DATA part), $N_{SYM}$ as follows

   $$N_{SYM} = Ceiling\ ((16 + 8 \times \ Length + 6) / N_{DBPS})$$

   where $N_{DBPS}$ is determined by the data rate in the following table.
   This model multiplexes one SIGNAL OFDM symbol and $N_{SYM}$ DATA OFDM symbols into $N_{SYM} + 1$ OFDM symbols for output. The SIGNAL OFDM symbol is output first, then the DATA OFDM symbols are output.

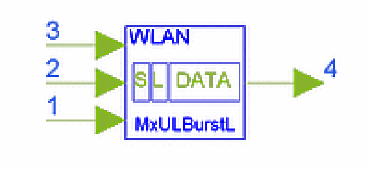
   ### Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_MuxULBurstL



**Description Uplink burst with long preamble multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxULBurstL**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmblB | short preamble | complex |
| 2 | LPrmbl | long preamble | complex |
| 3 | input | OFDM symbols of uplink PDU train | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | uplink burst signal with long preamble | complex |

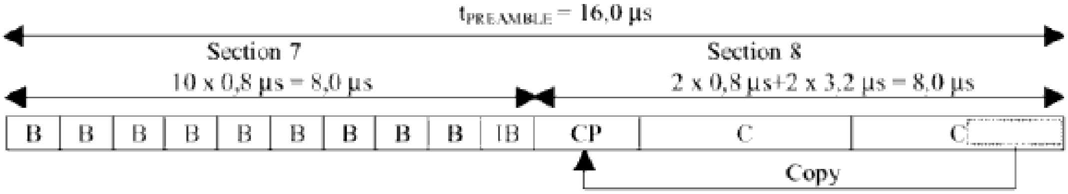
## Notes/Equations

1. This model is used to multiplex short and long preambles, and uplink PDU train into an uplink burst signal with a long preamble. Guard interval insertion is implemented.
2. Uplink burst with long preamble consists of a preamble of length $t_{preamble}$ = 16.0 µsec and a payload section of length $N_{SYM} \times T_S$,

   where

   $N_{SYM}$ is the number of OFDM symbols in the payload section, set in the NSYM parameter
   $T_S$ is the OFDM symbol interval
   ( $T_S$ = 4.0 µsec if GuardType=T_4, $T_S$ = 3.6 µsec if GuardType=T_8).

   The uplink burst structure with long preamble is illustrated in the following figure and described here.

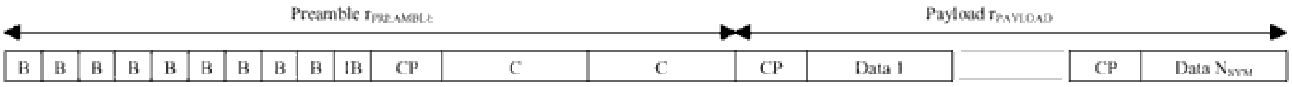


**Uplink Burst with Long Preamble**

The term *short OFDM symbol* refers only to its length that is $2^{Order\ -2}$ samples instead of a regular OFDM symbol of $2^{Order}$ samples used in HiperLAN/2 systems.

> Section 7 consists of 10 specific short OFDM symbols denoted in figure 1 by B and IB. The first 4 short OFDM symbols in this section (B, B, B, B) constitute a regular OFDM symbol consisting of 12 loaded sub-carriers (±4, ±8, ±12, ±16, ±20, and ±24) given by the frequency-domain sequence SB:
> SB -26...-26 = !wlan-06-25-56.gif!× {0,0,1+j,0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0,0,0,0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0}
> The last short symbol in section 7 (IB) is a sign-inverted copy of the preceding short symbol B, i.e. IB = -B.
> Section 8 consists of two OFDM symbols (C) of normal length preceded by a cyclic repetition (CP) of the symbols. All 52 sub-carriers are in use and these are modulated by the elements of the frequency-domain sequence SC given by:
> $SC_{-26...-26}$ = {1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1}
> The cyclic repetition CP is a copy of the $2^{Order\ -1}$ last samples of the C symbols and is thus double in length compared to the cyclic prefix of the normal data symbols. Thus section 8 is equal to section 3, section 4, and section 6.

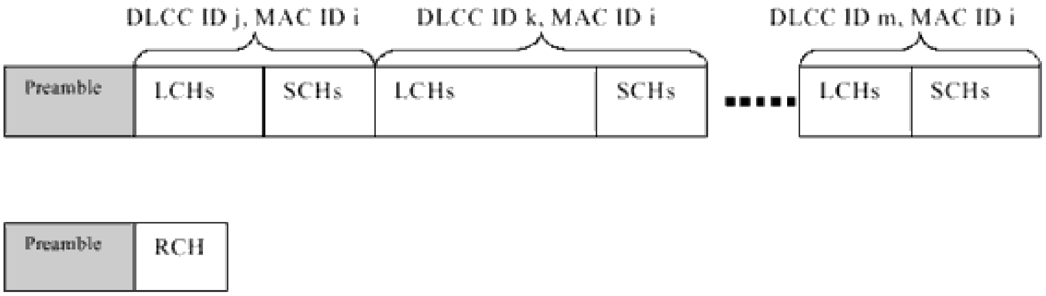
The uplink burst with long preamble is formed by concatenating the above described preamble with the data payload.
The resulting uplink burst is illustrated in the following figure.



**PHY Burst Structure for Uplink Burst with Long Preamble**

One preamble must be added at the beginning of each uplink PDU train, the following figure. The preamble used for uplink PDU trains is presented in the BCCH in the *uplink preamble* field, which is set to 1 for the long preamble. The preamble of the uplink PDU train with long preamble must have a length of 4 OFDM symbols, see reference [1].
A set of SCHs and LCHs is granted for each DLCC by one RG. An MT cannot receive

more than one uplink PDU train for the transmission of data, that is, all corresponding DLCCs must be grouped in a single PDU train.
RCH access is possible.



**Possible Uplink PDU Train with Long Preamble**

The number of OFDM symbols per transport channels is shown in the following table.

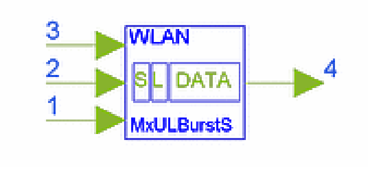
**Number of OFDM Symbols per Transport Channel (Excluding Physical Layer Preambles)**

| PHY mode | BCH, 15oct. | FCH, 27oct. | ACH, 9oct. | SCH, 9oct. | LCH, 54oct. | RCH, 9oct. |
|---|---|---|---|---|---|---|
| BPSK, code rate=1/2 | 5 | 9 | 3 | 3 | 18 | 3 |
| BPSK, code rate=3/4 | | | | 2 | 12 | |
| QPSK, code rate=1/2 | | | | | 9 | |
| QPSK, code rate=3/4 | | | | 1 | 6 | |
| 16QAM, code rate=9/16 | | | | | 4 | |
| 16QAM, code rate=3/4 | | | | | 3 | |
| 64QAM, code rate=3/4 | | | | | 2 | |

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_MuxULBurstS



**Description Uplink burst with short preamble multiplexer**
**Library WLAN, Multiplex**
**Class SDFWLAN_MuxULBurstS**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| NSYM | number of OFDM symbols | 1 | int | [1, ∞) |
| GuardType | type of guard interval: T_2, T_4, T_8, T_16, T_32 | T_4 | enum | |
| Order | FFT points=2^Order | 6 | int | [6, 11] |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | SPrmblB | short preamble | complex |
| 2 | LPrmbl | long preamble | complex |
| 3 | input | OFDM symbols of uplink PDU train | complex |

## Pin Outputs

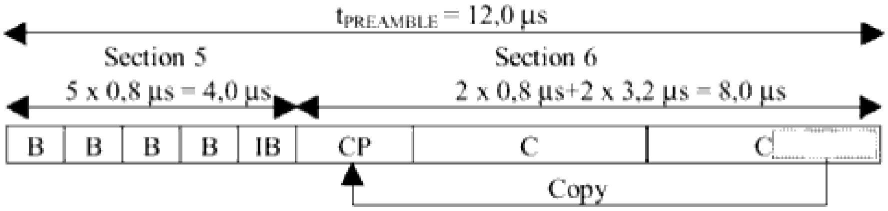
| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 4 | output | uplink burst signal with short preamble | complex |

## Notes/Equations

1. This model is used to multiplex short preamble, long preamble, and uplink PDU train into a uplink burst signals with short preamble. Guard interval insertion is implemented.
2. Uplink burst with short preamble consists of a preamble of length $t_{preamble}$ =12.0 μsec and a payload section of length $N_{SYM} \times T_S$,

   $N_{SYM}$ is the number of OFDM symbols in the payload section, set in the NSYM parameter
   $T_S$ is the OFDM symbol interval
   ( $T_S$ = 4.0 μsec if GuardType=T_4, $T_S$ = 3.6 μsec if GuardType=T_8).

   The short preamble structure for uplink bursts is illustrated in the following figure and described here.

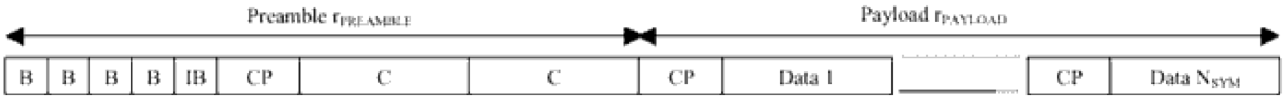


**Short Preamble for Uplink Bursts**

The term *short OFDM symbol* refers to length that is 2 $^{Order\ -2}$ samples instead of a regular OFDM symbol of 2 $^{Order}$ samples used in HiperLAN/2 systems.
Sections 5 and 6 are equal to the broadcast burst preamble sections 2 and 3, respectively.

> Section 5 consists of 5 specific short OFDM symbols denoted B and IB. The first 4 short OFDM symbols (B, B, B, B) constitute a regular OFDM symbol consisting of 12 loaded sub-carriers (±4, ±8, ±12, ±16, ±20, and ±24) given by the frequency-domain sequence SB:
> SB $_{-26\ \ldots\ 26}$ = !wlan-06-26-61.gif!× {0,0,1+j,0,0,0,-1-j,0,0,0,1+j,0,0,0,-1-j,0,0,0,-1-j,0,0,0,1+j,0,0,0,0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0}
> The last short symbol (IB) is a sign-inverted copy of the preceding short symbol B, i.e. IB = -B.
> Section 6 consists of two OFDM symbols (C) of normal length preceded by a cyclic repetition (CP) of the symbols. All 52 sub-carriers are in use and are modulated by the elements of the frequency-domain sequence SC given by:
> SC $_{-26\ \ldots\ 26}$ ={1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1}
> The cyclic prefix CP is a copy of the 2 $^{Order\ -1}$ last samples of the C symbols and is thus double in length compared to the cyclic prefix of the normal data symbols.

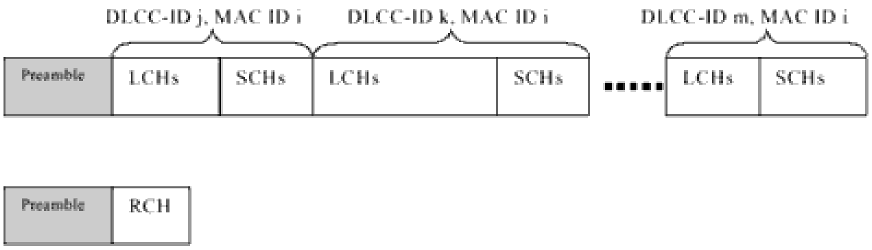
The uplink burst is formed by concatenating the above described preamble with the data payload. The resulted uplink burst is as illustrated in the following figure.



**PHY Burst Structure for Uplink Burst with Short Preamble**

One preamble must be added at the beginning of each uplink PDU train, see the following figure. The preamble used for uplink PDU trains is presented in the BCCH in the *uplink preamble* field which is set to zero for the short preamble. The preamble of the uplink PDU train with short preamble must have a length of 3 OFDM symbols, see reference [1].
A number of SCHs and LCHs is granted for each DLCC by one RG. An MT cannot receive more than one uplink PDU train for the transmission of data, that is, all

corresponding DLCCs must be grouped in a single PDU train.
RCH access is possible.



**Example Uplink PDU Train with Short Preamble**

The number of OFDM symbols per transport channels is shown in the following table.

**Number of OFDM Symbols per Transport Channel (Excluding Physical Layer Preambles)**

| **PHY mode** | **BCH, 15oct.** | **FCH, 27oct.** | **ACH, 9oct.** | **SCH, 9oct.** | **LCH, 54oct.** | **RCH, 9oct.** |
|---|---|---|---|---|---|---|
| BPSK, code rate=1/2 | 5 | 9 | 3 | 3 | 18 | 3 |
| BPSK, code rate=3/4 | | | | 2 | 12 | |
| QPSK, code rate=1/2 | | | | | 9 | |
| QPSK, code rate=3/4 | | | | 1 | 6 | |
| 16QAM, code rate=9/16 | | | | | 4 | |
| 16QAM, code rate=3/4 | | | | | 3 | |
| 64QAM, code rate=3/4 | | | | | 2 | |

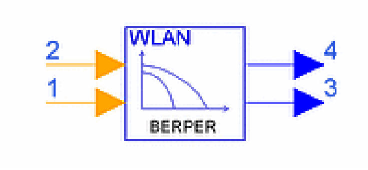
## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# Test Components for WLAN Design Library

- *WLAN BERPER* (wlan)
- *WLAN EVM* (wlan)
- *WLAN RF CCDF* (wlan)
- *WLAN RF PowMeas* (wlan)

# WLAN_BERPER



**Description: Bit and packet error rate measurements**
**Library: WLAN, Test**
**Class: SDFWLAN_BERPER**

### Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| Length | octet number of PSDU | 100 | | int | [1:4095] |
| Delay | delay number of PSDUs | 2 | | int | [0, ∞) |

### Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | test | received PSDU | int |
| 2 | ref | reference input PSDU | int |

### Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | ber | output BER value | real |
| 4 | per | output PER value | real |

### Notes/Equations

1. This model is used to measure the input signal bit error rate (BER) and packet error rate (PER).
   1 token at pin ber and 1 token at pin per are produced when Length × 8 input tokens are consumed. Note that the Length parameter is in bytes while test and ref signals are in bits.
2. The model calculates the error rate after the specified Delay time. Its procedure is as follows:

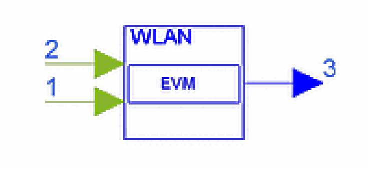
```
count = 0;
perr = 0;
err = 0;
brate = 0.0;
prate = 0.0;
if (count>=delay)
{
prev=err;_
for (i=0;i<length;i++)
{
t1 = test%(i);
r1 = ref%(i);
t1 = t1&1;
```

```
r1 = r1&1;
if (t1!=r1)
err = err+1;
}
if (err>prev)
perr = perr+1;
}
count++;
if ( count>delay )
{ brate = ((float) err)/((count-delay)*length); prate = (float) perr/(count-delay);}
ber%(0) << brate;
per%(0) << prate;
```

## References

1. ETSI TS 101 475 v1.2.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer," November, 2000.
2. ETSI TS 101 761-1 v1.1.1, "Broadband Radio Access Networks (BRAN); Data Link Control (DLC) Layer Part1: Basic Data Transport Functions," April, 2000.

# WLAN_EVM



**Description Error Vector Magnitude**
**Library WLAN, Test**
**Class SDFWLAN_EVM**

## Parameters

| Name | Description | Default | Type | Range |
|---|---|---|---|---|
| Length | octet number of PSDU | 100 | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | enum | |
| Carriers | number of carriers in one OFDM symbol | 52 | int | 48 or 52 |
| Nf | number of frames for the measurement | 30 | int | [1, ∞) see Note 4 |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input1 | ideal signal | complex |
| 2 | input2 | observed signal | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 3 | output | EVM output | real |

## Notes/Equations

1. This model is used to determine RMS average of the error vector magnitude. Each firing, NSYM × 52 or NSYM × 48 tokens are consumed at pins input1 and input2, whereas 52 or 48 is determined by the input signal in which 4 pilot carriers are removed, 48 is selected, otherwise, 52 is correct. Then, 1 token of EVM value is produced.
2. Length and Rate parameters determine the number of complex signals in one burst. The number of OFDM symbols, *NSYM* can be calculated as:

$$N_{SYM} = Ceiling((16 + 8 \times Length + 6)/N_{DBPS})$$

where $N_{DBPS}$ is determined by data rate, shown in the following table.

### Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

3. The averaged RMS EVM can be caculated as follows:
    - the RMS average of all errors in a packet is calculated using the formula

$$Error_{RMS} = \frac{\sum_{i=1}^{N_f} \sqrt{\frac{\sum_{j=1} \left[ \sum_{k=1} \{(I(i,j,k) - I_0(i,j,k))^2 + (Q(i,j,k) - Q_0(i,j,k))^2\} \right]}{Carriers \times N_{SYM} \times P_0}}}{N_f}$$

    where
    *Carriers* is the number of subcarriers (48 or 52) in one OFDM symbol
    $N_{SYM}$ is the length of the packet

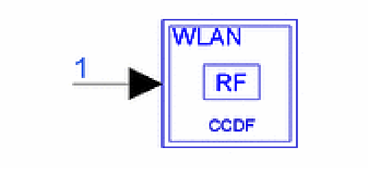
    $N_f$ is the number of frames for the measurement

    ( $I_0(i, j, k)$, $Q_0(i, j, k)$) denotes the ideal symbol point of the $i^{th}$ frame, *j* ^_th
    ^ *OFDM symbol of the frame, _k* $^{th}$ subcarrier of the OFDM symbol in the complex plane
    ( $I(i, j, k)$, $Q(i, j, k)$) denotes the observed point of the $i^{th}$ frame, $j^{th}$ OFDM
    symbol of the frame, $k^{th}$ subcarrier of the OFDM symbol in the complex plane
    $P_0$ is the average power of the constellation
4. The test must be performed over at least 20 frames $N_f$, and the $N_f$ RMS EVM are averaged as one averaged RMS EVM. The packets under test must be at least 16 OFDM symbols long.

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.

# WLAN_RF_CCDF



**Description RF signal complementary cumulative distribution function**
**Library WLAN, Test**
**Class TSDFWLAN_RF_CCDF**

## Parameters

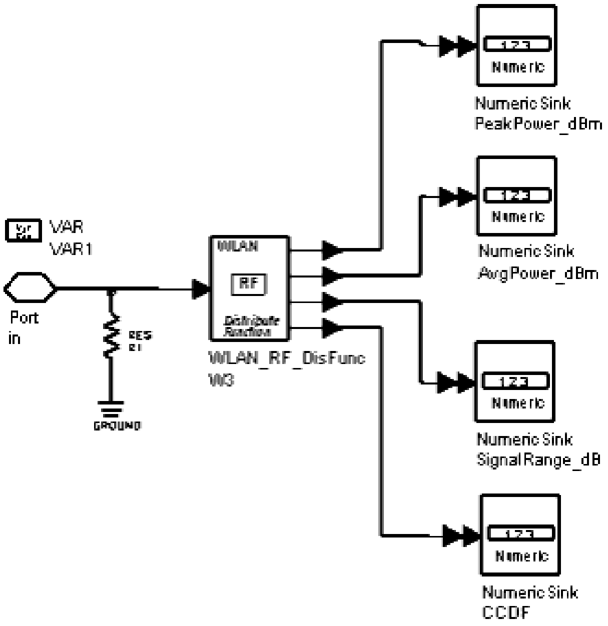
| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| StartSym | symbol from which measurement begins | 0 | | int | [0, ∞) |
| BurstLen | length of input signal burst | 2560 | | int | (1, ∞) |
| OutputPoint | output precision | 100 | | int | (0, BurstLen * BurstNum] |
| BurstNum | number of bursts | 1 | | int | (0, ∞) |
| RLoad | reference resistor | 50.0 | Ohm | real | (0, ∞) |
| RTemp | temperature of reference resistor | -273.15 | | real | [-273.15, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | in | input signals | timed |

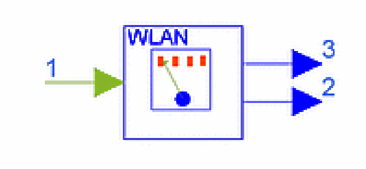
## Notes/Equations

1. This subnetwork is used to measure the complementary cumulative distribution function (CCDF) according to the input signals.
   Each firing, BurstLen × BurstNum signals are input (BurstLen is the length of frame and BurstNum is the number of frames to be measured).
   The schematic is shown in the following figure.
   WLAN_RF_DisFunc measures the distribution function according to the input signal power; results are collected by the NumericSink components. The distribution range is sent to the *SignalRange* NumericSink and is divided into segments according to the OutputPoint parameter. The corresponding distribution probability is calculated based on these segments and sent to the *CCDF* NumericSink.
   WLAN_RF_DisFunc calculates peak power of 99.9% probability and average power of input signals. These results are collected by the *PeakPower* and *AvgPower* NumericSinks.
   Note that the units of *PeakPower and AvgPower* are dBm; *SignalRange* is the transient absolute signal power minus *AvgPower* , so the unit of SignalRange is dB.



### WLAN_RF_CCDF Schematic

# WLAN_RF_PowMeas



**Description Power level measurement**
**Library WLAN, Test**
**Class SDFWLAN_RF_PowMeas**

## Parameters

| Name | Description | Default | Unit | Type | Range |
|---|---|---|---|---|---|
| SignalType | type of signal to be measured: Baseband, RF | RF | | enum | |
| RefR | reference resistance | 50.0 | Ohm | real | (0, ∞) |
| Length | octet number of PSDU | 100 | | int | [1, 4095] |
| Rate | data rate: Mbps_6, Mbps_9, Mbps_12, Mbps_18, Mbps_24, Mbps_27, Mbps_36, Mbps_48, Mbps_54 | Mbps_6 | | enum | |
| Order | FFT points=2^Order | 6 | | int | [6, 11] |
| Nf | number of frames for the measurement | 30 | | int | [1, ∞) |

## Pin Inputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 1 | input | signal to be measured | complex |

## Pin Outputs

| Pin | Name | Description | Signal Type |
|---|---|---|---|
| 2 | outputInst | output instant power | real |
| 3 | outputavg | output average power | real |

## Notes/Equations

1. This model is used to calculate instant and average power according to the received signals. Power calculation is described as follows; the complex signal at instant i is denoted as $R_i$ ; the output is in dBm.
    - If an RF signal is measured, power is expressed as $10\log(R_i^2/(2\times RefR))$
    - If a baseband signal is measured, power is expressed as $10\log(R_i^2)$
2. Length and Rate parameters are used to determine the number of complex signals in one burst. The number of OFDM symbols $N_{SYM}$ is calculated:

    $N_{SYM} = Ceiling\ ((16 + 8 \times Length + 6)/\ N_{DBPS}\ )$

    where $N_{DBPS}$ is determined by the data rate listed in the following table.

    After determining $N$ , the number of input tokens $N$ is calculated as follows:

$_{SYM}$ $_{total}$

$N_{total} = (2^{Order} + 2^{Order\ (-2)}) \times (N_{SYM} + 5)$

The buffer length for input and output pins is $N_{total}$ ; $N_{total}$ tokens are fired each operation. Instant power is calculated according to the magnitude at each point. For the first frame, average power equals the instant power; for the second frame, average power equals the average of 2 instant power. Accordingly, for Nf frame, average power equals the average of Nf instant power.

### Rate-Dependent Values

| **Data Rate (Mbps)** | **Modulation** | **Coding Rate (R)** | **Coded Bits per Subcarrier (NBPSC)** | **Coded Bits per OFDM Symbol (NCBPS)** | **Data Bits per OFDM Symbol (NDBPS)** |
|---|---|---|---|---|---|
| 6 | BPSK | 1/2 | 1 | 48 | 24 |
| 9 | BPSK | 3/4 | 1 | 48 | 36 |
| 12 | QPSK | 1/2 | 2 | 96 | 48 |
| 18 | QPSK | 3/4 | 2 | 96 | 72 |
| 24 | 16-QAM | 1/2 | 4 | 192 | 96 |
| 27 | 16-QAM | 9/16 | 4 | 192 | 108 |
| 36 | 16-QAM | 3/4 | 4 | 192 | 144 |
| 48 | 64-QAM | 2/3 | 6 | 288 | 192 |
| 54 | 64-QAM | 3/4 | 6 | 288 | 216 |

## References

1. IEEE Standard 802.11a-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band," 1999.