

# *VisSim/Comm™ Wireless LAN*

## *User Guide*

Version 8.0



**Eritek, Inc.**

***Visual Solutions***

INCORPORATED

**Eritek, Inc.**  
**Visual Solutions, Inc.**

**VisSim/Comm WLAN User's Guide - Version 8.0**

---

<b>Copyright</b>	© 2011 Eritek, Inc. All rights reserved. Printed and bound in the USA. vewlug-080-01	Eritek, Inc. 18450 Sydnor Hill Ct. Leesburg, VA 20175	Visual Solutions, Inc. 487 Groton Road Westford, MA 01886
------------------	--	---	---

**Trademarks** VisSim is a trademark of Visual Solutions. VisSim/Comm is a joint trademark of Eritek and Visual Solutions. Other products mentioned in this manual are trademarks of their respective manufacturers.

**Copy and use restrictions** The information in this document is subject to change without notice and does not represent a commitment by either Eritek or Visual Solutions. Eritek and Visual Solutions do not assume responsibility for errors that may appear in this document.

No part of this manual may be reprinted or reproduced or utilized in any form or by any electronic, mechanical, or other means without permission in writing from Eritek. The Software may not be copied or reproduced in any form, except as stated in the terms of the Software license agreement.

Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in FAR 52.227-19, subparagraph (c)(i)(ii) of DOD FAR SUPP 252.227-7013, or equivalent government clause for other agencies.

# Contents

<b>Introduction.....</b>	<b>5</b>
Computer requirements.....	5
Starting VisSim/Comm WLAN.....	5
Manual conventions.....	5
Technical support service .....	6
<b>WLAN Module Overview.....</b>	<b>9</b>
Wireless system communication elements .....	9
Wireless blocks.....	9
Bluetooth Overview.....	10
802.11 Overview.....	10
802.11a/g Overview.....	11
802.11b Overview.....	12
Ultrawideband Overview.....	12
Sample wireless simulation .....	12
<b>WLAN Block Set.....</b>	<b>15</b>
BLUETOOTH BLOCKS.....	15
Bluetooth Hop Generator .....	15
Bluetooth Scrambler.....	16
GFSK Modulator.....	17
Shortened Hamming Decoder .....	17
Shortened Hamming Encoder.....	18
802.11 BLOCKS.....	18
Barker Sequence.....	18
CRC-16 Generator.....	19
802.11 Hop Generator .....	20
802.11 Descrambler.....	21
802.11 Scrambler.....	21
GFSK-2 Modulator.....	22
GFSK-4 Modulator.....	22
802.11a/g BLOCKS.....	23
802.11a/g Convolutional Encoder .....	23
802.11a/g Depuncture.....	23
802.11a/g Puncture.....	24
802.11a/g Interleaver.....	25
802.11a/g Scrambler.....	25
802.11a/g Viterbi Decoder .....	26
OFDM Demodulator .....	28
OFDM Modulator.....	28
OFDM Pilot Extract .....	30
OFDM Pilot Map.....	31

OFDM Vector Demodulator.....	32
OFDM Vector Modulator.....	33
802.11b BLOCKS.....	34
802.11b High Rate Hop Generator .....	34
CCK Demodulator.....	35
CCK Modulator.....	36
GENERIC WIRELESS BLOCKS.....	38
Frequency Hop .....	38
ULTRAWIDEBAND BLOCKS.....	39
Gated Integrate & Dump .....	39
UWB PPM Modulator .....	39
UWB Pulse.....	40
<b>Sample Block Diagrams .....</b>	<b>43</b>
<b>Acronyms and Abbreviations .....</b>	<b>44</b>
<b>Index .....</b>	<b>46</b>

# Introduction

The WLAN module extends the capabilities of the VisSim/Comm physical layer simulation environment by including support for Bluetooth, 802.11a/b/g and Ultrawideband (UWB) wireless designs. These standards are commonly used in the design of Wireless Local Area Networks (WLAN). In particular, the 802.11 standard has gained worldwide acceptance in the implementation of “Wi-Fi” networking components.

It is assumed in this manual that you are familiar with the use and operation of the VisSim/Comm simulation environment. Please consult the *VisSim User Guide* for details on the overall simulation environment and graphical user interface, or the *VisSim/Comm User Guide* for matters relating to other communication blocks.

Many thanks to Jian Sun and Dr. Matt Valenti of West Virginia University for collaborating in the early development of this Wireless module.

---

## Computer requirements

The WLAN for VisSim module requires the following components to run:

- Windows WinXP/Vista/Win7
- Visual Solutions VisSim 8.0 or later
- VisSim/Comm DLL version 8.0 or later

---

## Starting VisSim/Comm WLAN

Once the WLAN module has been successfully installed, all its features are accessed by starting VisSim/Comm and accessing the **Wireless** menu on the main toolbar.

During the install process, the VISSIM.INI file in the windows directory is updated to specify the path to the WLAN module DLL so that VisSim may load the DLL at program startup. Should you desire not to load the WLAN DLL at a future time, simply remove the “WLAN\_vis.dll” entry from the Addons listing under the Edit/Preferences menu command (highlight the entry and press delete).

---

## Manual conventions

The following conventions are used in this manual:

- Block descriptions are arranged alphabetically within each category (Bluetooth, 802.11, 802.11a, 802.11b, Generic Wireless, and UWB).

- Unless specifically stated otherwise, use the left mouse button whenever you are choosing a command, selecting a block, or activating a dialog box parameter. For example, when you read *press the OK button...*, you are to position the pointer over the specified object and click the left mouse button.
- To choose a menu command, you can use the mouse or you can press a sequence of keyboard keys. Only the mouse operations are documented.

The following visual conventions are used to make this manual easier to read.

Visual convention	Where it's used
Italics	To reference a book, chapter, or section. Also used to emphasize certain keywords.
small caps	To indicate the names of keys on the keyboard.
Shortcut key combinations	Shortcut key combinations are joined with the plus sign (+). For example, the command ctrl+c means to hold down the ctrl key while you press the C key.
ALL CAPS	To indicate directory names, filenames, and acronyms.
Initial Caps	To indicate menu names, command names, and dialog box options.

---

## Technical support service

When you need assistance with a Visual Solutions product, first look in the manual, review the readme file, and consult the online Help program. If you cannot find the answer, contact the Technical Support group via toll call between 9:00 am and 6:00 pm EST, Monday through Friday, excluding holidays. The phone number is **978-392-0100**.

For questions specific to the Wireless module, please contact Eritek via e-mail at: **support@eritek.com**

When you call in, please have the following information at hand:

- The version of VisSim, VisSim/Comm DLL, and WLAN DLL that you're using (Click on "Help/About VisSim...", "Help/About VisSim/Comm Module...", and "Help/About Wireless Module..." to obtain this information).
- Your VisSim/Comm serial number.
- The type of hardware that you're using and Windows Operating System.
- All screen messages.
- What you were doing when the problem happened.
- How you tried to solve the problem.

Visual Solutions may also be reached on the web at [www.vissim.com](http://www.vissim.com) or via the following fax and e-mail addresses:

<b>Address/Number</b>	<b>What it's for</b>
(978) 692-3102	Fax number
bugs@vissol.com	Bug report
doc@vissol.com	Documentation errors and suggestions
sales@vissol.com	Sales, pricing, and general information
tech@vissol.com	Technical support

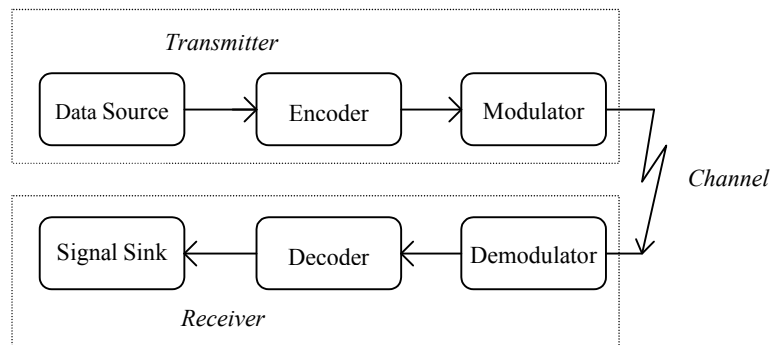


# WLAN Module Overview

---

## Wireless system communication elements

The physical layer of a wireless link usually includes the elements depicted in the following figure. Many of the blocks required to model these elements are provided by the standard VisSim/Comm block set. More complex wireless system functions, however, cannot be easily simulated using the standard Comm block set. The WLAN module fills this gap by including additional blocks specific to the Bluetooth, UWB, and IEEE 802.11 standards.



The new blocks included in the Wireless LAN module consist primarily of Encoder/Decoder and Modulator/Demodulator functions. A full listing of these blocks is provided in the next section.

---

## Wireless blocks

The blocks included in the Wireless LAN module are organized in six categories according to the specific Standard they support:

- Bluetooth
- 802.11 (1 Mbps and 2 Mbps modes)
- 802.11a /g (OFDM modulation at 6 ~ 54 Mbps)
- 802.11b (CCK modulation at 5.5 and 11 Mbps)
- Generic Wireless
- Ultrawideband (UWB)

### **Bluetooth**

GFSK, Hop Generator, Scrambler, Short Hamming Encoder, Short Hamming Decoder

### **802.11**

Barker Sequence, CRC-16, Descrambler, GFSK-2, GFSK-4, Hop Generator, Scrambler

### **802.11a/802.11g**

Convolutional Encoder, Depuncture, Interleaver, Puncture, Scrambler, Viterbi Decoder, OFDM Demodulator, OFDM Modulator, OFDM Pilot Extract, OFDM Pilot Map, OFDM Vector Demodulator, OFDM Vector Modulator

### **802.11b**

High Rate Hop Generator, CCK Baseband Modulator, CCK Demodulator

### **Generic Wireless**

Frequency Hop

### **Ultrawideband**

Gated Integrate & Dump, UWB PPM Modulator, UWB Pulse

---

## **Bluetooth Overview**

The Bluetooth standard provides wireless data and/or voice communication between multiple peripherals located in close proximity of each other (typical range is 10 meters). It can be operated in point-to-point or point-to-multipoint modes, and exhibits a channel symbol rate of 1 Mbps, with a maximum payload rate of 723 kbps. A Bluetooth network is also referred to as a “piconet”.

In each piconet one Bluetooth unit acts as the master and the remaining units act as slaves, with up to seven active slave units being allowed. Access to the channel is controlled by the master. A Time-Division Duplex (TDD) communication scheme is used, in which packets are exchanged between master and slave in alternating fashion.

Bluetooth operates in the 2.4 GHz ISM (Industrial Scientific Medicine) microwave band, and employs a frequency hopping scheme spanning 79 channels (1 MHz spacing) and with a hopping rate of 1.6 kHz. The pseudo-random hopping pattern is derived from a combination of the master clock time and the unique device address of each unit.

The modulation used in Bluetooth is Gaussian Frequency Shift Keying (GFSK) with a BT value of 0.5. A “1” is represented by a positive frequency shift and a “0” is represented by a negative frequency shift. The modulation index must be in the range of 0.28 – 0.35, which corresponds to a frequency deviation of 140 ~ 175 kHz.

The Bluetooth channel is organized into time slots of 625 us in duration. The start of each packet must be aligned with the beginning of a new time slot, and a packet may extend for up to five time slots in duration. The master always starts a transmission in even-numbered time slots, while the slave always starts its transmissions in odd-numbered time slots. Each packet is transmitted on a different hop frequency.

For more details on the Bluetooth specification, please visit the official Bluetooth web site at:

[www.bluetooth.com](http://www.bluetooth.com)

---

## **802.11 Overview**

The IEEE 802.11 standard was developed to provide Local Area Network (LAN) services in a wireless communications environment. The standard supports operation in the 2.4 GHz ISM band using either a Frequency Hopped Spread Spectrum (FHSS) approach or Direct Sequence Spread Spectrum (DSSS).

The FHSS format uses binary or 4-ary GFSK modulation ( $BT=0.5$ ) and supports data rates of 1 Mbps and 2 Mbps respectively. In the US, the set of operating transmit and receive channels for the FHSS specification includes 79 frequencies, with center frequencies ranging from 2402 MHz to 2480 MHz. Channels are spaced at 1 MHz intervals. In the US, the FHSS hopping patterns are grouped into 3 sets, each including 26 patterns.

The DSSS format also provides 1 and 2 Mbps data rates, but employs differential PSK baseband modulation formats. The 1 Mbps mode uses Differential Binary Phase Shift Keying (DBPSK), and the 2 Mbps mode uses Differential Quadrature Phase Shift Keying (DQPSK). The modulated baseband signals (both 1 Msps) are then spread using a chip rate of 11 MHz (i.e. there are 11 chips per symbol). The symbols are spread using an 11-chip Barker sequence, whose start time is aligned with the start of each symbol. In the US, 11 operating channels are available for the 802.11 DSSS specification, with center frequencies ranging from 2412 to 2462 MHz.

More details on the 802.11 standard may be obtained directly from the IEEE. The 802.11 specification is available for free download from the following URL:

<http://standards.ieee.org/getieee802/>

## 802.11a/g Overview

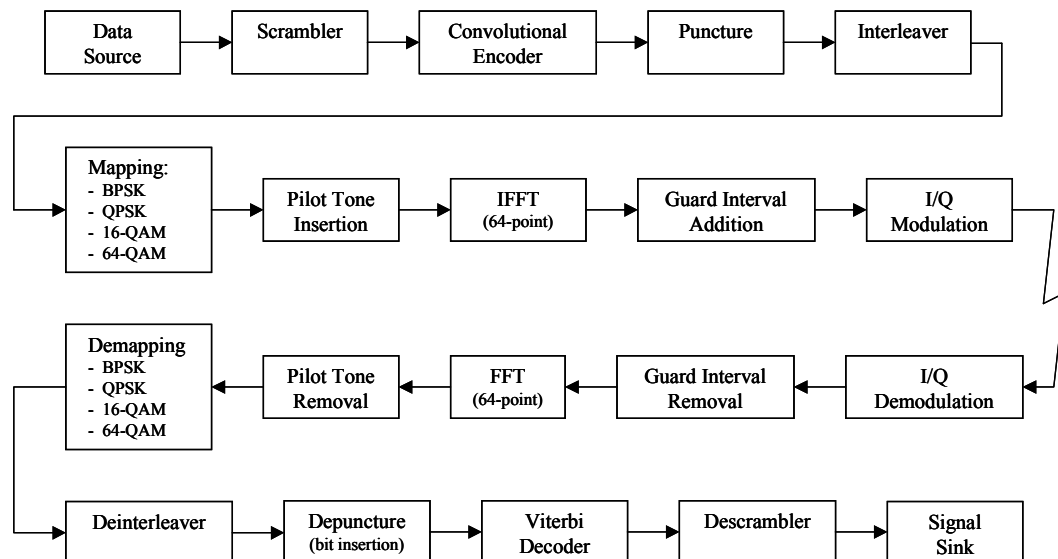
The IEEE 802.11a standard is an extension to the original 802.11 standard devised to significantly increase the system data rate. Unlike the 802.11b extension, 802.11a is not compatible with the original 802.11 frequency plan (2.4 GHz band), and operates instead in the 5 ~ 6 GHz band.

The 802.11a standard uses Orthogonal Frequency Division Multiplexing (OFDM) modulation and supports data rates ranging from 6 Mbps to 54 Mbps. The OFDM modulation uses 52 subcarriers that are modulated using BPSK, QPSK, 16-QAM, or 64-QAM. Convolutional coding is used for Forward Error Correction (FEC), at coding rates of 1/2, 2/3, or 3/4.

The newer 802.11g standard, combines the higher data rate capability of 802.11a while operating in the same 2.4 GHz band as 802.11b. It specifies the use of either PBCC modulation (not currently supported by the WLAN module) or OFDM as defined in the 802.11a specification.

In the US, the set of operating frequencies for the 802.11a OFDM specification includes 12 channels, with center frequencies ranging from 5180 to 5805 MHz.

A simplified block diagram of an 802.11a/g OFDM modulation physical layer link is shown in the following figure.



Simplified 802.11a/g Physical Layer Block Diagram

More details on the 802.11 standard may be obtained directly from the IEEE. The 802.11a and 802.11g specifications are available for free download from the following URL:

<http://standards.ieee.org/getieee802/>

---

## 802.11b Overview

The IEEE 802.11b standard provides a high speed physical layer extension to the original 802.11 DSSS specification. It operates in the 2.4 GHz ISM band, and remains compatible with the original 802.11 frequency plan (see above).

The 802.11b standard extends the capabilities of 802.11 by adding Complementary Code Keying (CCK) modulation at rates of 5.5 and 11 Mbps. As with 802.11, a chipping rate of 11 MHz is used. For compatibility purposes, the packet's preamble and headers are still transmitted using DBPSK and DQPSK at 1 Mbps and 2 Mbps.

An optional Packet Binary Convolutional Coding (PBCC) mode, providing a data rate of 5.5 Mbps or 11 Mbps, is also specified in the standard. The PBCC mode is not yet included in the WLAN module, but is planned for future releases.

The 802.11b standard also includes an optional Channel Agility mode, in which the signal can frequency-hop across a subset of the available DSSS channels.

More details on the 802.11b specification may be obtained directly from the IEEE. The 802.11b specification is available for free download from the following URL:

<http://standards.ieee.org/getieee802/>

---

## Ultrawideband Overview

Ultrawideband (UWB) signals are defined by the FCC as waveforms having a fractional bandwidth greater than 20% or occupying a bandwidth of more than 500 MHz. UWB signals typically consist of very short time duration pulses in the picosecond to nanosecond range, and are transmitted directly at RF without the use of a modulating carrier. Typical bandwidths exceed several GHz.

Common applications include short-range indoor communications. One of the advantages of UWB schemes is that they are capable of operating in heavy multipath environments due to the extremely short duration of the signaling pulses.

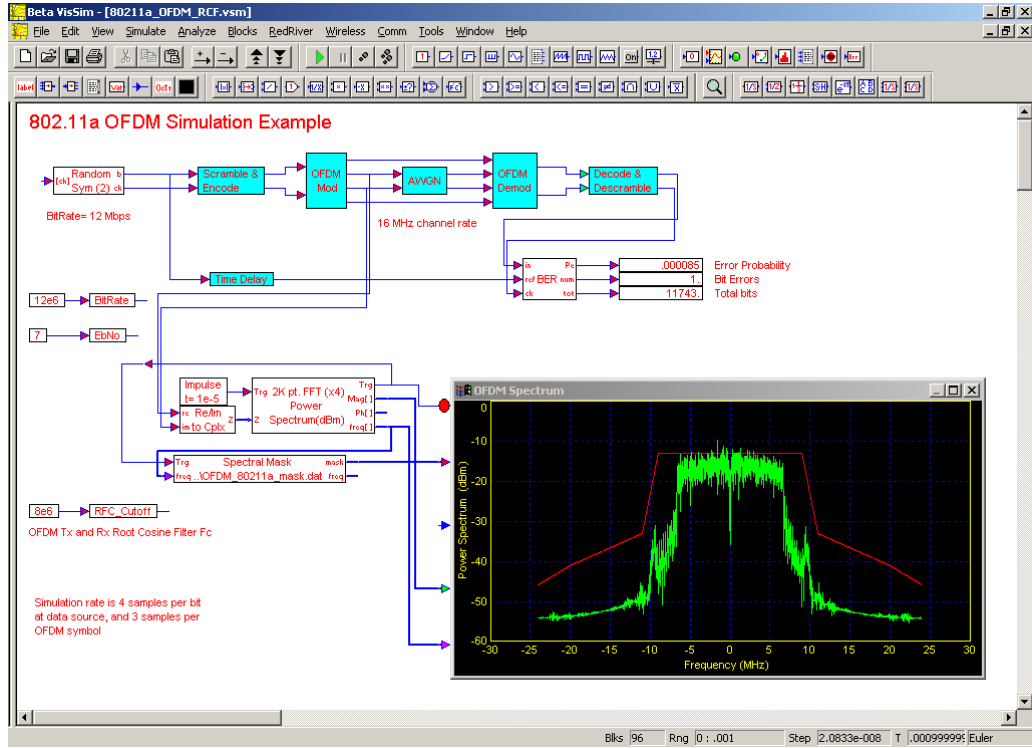
---

## Sample wireless simulation

An example of a communication system simulation using 802.11a components is shown below. The example diagram "80211a\_OFDM\_RCF.vsm" is located in the "Wireless" folder. The actual example file may differ slightly from the diagram that follows.

This example focuses on the 802.11a OFDM modulation format. The diagram displays a modulated OFDM spectrum for a 12 Mbps link along with the corresponding FCC mask. The diagram includes data scrambling, encoding, subcarrier QPSK modulation, pilot tone insertion and the OFDM modulation process via inverse FFT. The diagram also computes Bit Error Rate (BER) performance data assuming a Gaussian noise channel.

For more technical details on the OFDM modulation format, please refer to the OFDM Modulator block description in Chapter 3 (802.11a Section).





---

# WLAN Block Set

---

## BLUETOOTH BLOCKS

### Bluetooth Hop Generator

This block generates a Bluetooth frequency-hopping pattern and a Master Clock output. Different hop patterns are generated for the Master and Slave, and a toggle output is provided to indicate which time slot (master or slave) is active. The Bluetooth Master Clock has a cycle length of  $2^{28}$  (28 bit shift register), which corresponds to  $2^{27}$  time slots as hops occur only on every other clock pulse. Master and Slave take turns transmitting, with the Master using even hop slot numbers and the Slave odd ones. The default time slot duration is 625 us.

This block supports both the 79 and 23 hop specifications. The hop sequence is derived from the current Bluetooth Master Clock value and the Device Address, as described in Section 11 of the Bluetooth Specification. The default Bluetooth hop rate is 1.6 kHz, which corresponds to a Master clock pulse rate of 3.2 kHz (hops occur on every other clock pulse).

This block support both internal and external timing. When in external mode, the user must provide both the Master Clock counter value and pulse train. This block outputs a hop pattern consisting of channel numbers in the range of [0, 78] or [0, 22] depending on the Hop Mode selection. This block should be followed by a Frequency Hop block to implement the actual signal hopping.

$x_1$  = Optional Master Clock counter value

$x_2$  = Optional Master Clock pulse train

$y_1$  = Master Hop Channel # [0, 78] or [0, 22]

$y_2$  = Slave Hop Channel # [0, 78] or [0, 22]

$y_3$  = Master/Slave Slot Flag (0= Master; 1= Slave)

$y_4$  = Master Clock counter value

$y_5$  = Master Clock pulse train

### Device Address

Specifies the device address in hex. This value is used to select the hop sequence generated by the block.

### Timing Mode

#### *Internal*

Indicates internal clock timing. The hop rate, start time, and Master Clock initial value must be specified.

**External**

Indicates external timing. An external clock and Master Clock counter value must be provided to the block.

**Hop Mode**

**79 Hops**

The block generates a hop sequence using 79 possible channels. This mode is used in the US and most of Europe.

**23 Hops**

The block generates a hop sequence using 23 possible channels. This mode is use in Japan, Spain and France.

**Init Master Clock**

Specifies the initial counter setting for Bluetooth Master Clock. This parameter is only available when in Internal Timing mode.

**Hop Rate**

Specifies the hop rate for the block in hops per second. Note that the output clock rate will be twice this rate. This parameter is only available when in Internal Timing mode. The default hop rate is 1.6 kHz.

**Start Time**

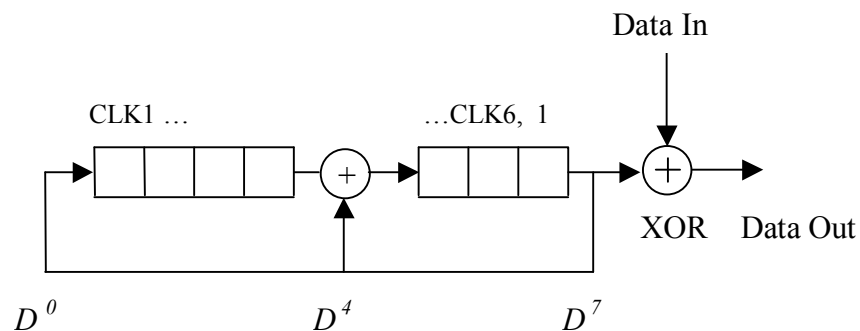
Specifies a starting time for the hop sequence and clock output. This parameter is only available when in Internal Timing mode.

**Bluetooth Scrambler**

This block provides a data scrambling and descrambling function as specified by the Bluetooth standard. The input data is XOR'ed with the output sequence from a feedback shift register of size  $N = 7$ . The shift register is re-initialized for each transmission (when a pulse is presented at the frame clock input) by using bits 1-6 of the current Bluetooth Master Clock and extending them with an MSB value of 1.

The generator polynomial for the feedback shift register is:

$$p(D) = D^7 + D^4 + 1$$



The user must provide an input bit stream and clock (pulse train) and periodically a frame clock pulse to re-initialize the internal shift register when desired. The user must also supply the Bluetooth Master Clock value (essentially the output of a binary counter), which can be obtained from a Bluetooth Hop Generator block.

The Bluetooth Scrambler block is used for both data scrambling and unscrambling.

$x_1$  = Input data

$x_2$  = Input data clock pulses

$x_3$  = Frame clock pulse  
 $x_4$  = Bluetooth Master Clock value  
 $y_1$  = Output data  
 $y_2$  = Output data clock pulses

*This block does not have any internal parameters.*

## GFSK Modulator

This block implements a Gaussian Frequency Shift Keying (GFSK) modulator as a compound block. In GFSK modulation, the digital information is transmitted by shifting the carrier frequency between two states. A “1” is represented by a positive frequency shift and a “0” is represented by a negative frequency shift.

This block employs a Gaussian FIR Filter and an FM Modulator as its internal components. The internal parameters of these blocks may need to be adjusted for proper operation, depending on the chosen data rate and simulation rate. The default parameters for the BT product, symbol rate, and FM deviation value reflect the Bluetooth specification for the 1 Mbps mode, which specifies a BT value of 0.5 and a frequency deviation range of +/- 140 ~ 175 kHz.

$x$  = Input data signal (binary [0, 1])  
 $y$  = Complex output signal [Re, Im]

## Shortened Hamming Decoder

This block implements a decoder for the shortened (15, 10) Hamming code as defined in the Bluetooth standard. The encoder matrix for this code is shown below. This code achieves a code rate of 2/3 and has the ability of correcting one bit error in a code word. The code is systematic and the parity-check matrix is shown below.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The decoding process includes computing the syndrome of the input vector and the parity-check matrix, locating any errors and correcting the error when possible. Since the code is systematic, the information bits reside in the first 10 bits of the codeword and the parity bits occupy the last 5 bits.

This block does not include any parameters. It accepts as input a coded vector of size 15 and outputs a decoded vector of size 10 elements.

$x_1$  = Input frame clock pulse  
 $x_2$  = Input coded vector (size 15)  
 $y_1$  = Output frame clock pulse  
 $y_2$  = Decoded output vector (size 10)

*This block does not have any internal parameters.*

## Shortened Hamming Encoder

This block implements an encoder for the shortened (15, 10) Hamming code as defined in the Bluetooth standard. The encoder matrix for this code is shown below. This code achieves a code rate of  $2/3$  and has the ability of correcting one bit error in a code word. The code is systematic and is described by the parity-check matrix  $H$  shown below.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The encoding process involves computing the parity bits for the given input word and appending these to the input to create the coded output word. Since the code is systematic, the information bits are preserved in the output codeword and reside in its first 10 bits. The last 5 bits represent the parity bits.

This block does not include any parameters. It accepts as input a data vector of size 10 and outputs an encoded vector of size 15 elements.

$x_1$  = Input frame clock pulse

$x_2$  = Input data vector (size 10)

$y_1$  = Output frame clock pulse

$y_2$  = Encoded output vector (size 15)

*This block does not have any internal parameters.*

---

## 802.11 BLOCKS

### Barker Sequence

This block generates a repeating Barker sequence. Barker codes are a family of pseudo-random sequences with quasi-ideal cross-correlation properties. Barker codes are often used in CDMA systems and are also part of the 802.11 2.4 GHz DSSS specification. The Barker code used in the 802.11 specification uses  $N=11$ .

The user can select the length of the Barker code, as well as the symbol rate and an initial delay. The following illustrates the output pattern for each available length:

$N=3$  1 1 0

$N=4$  1 1 0 1

$N=5$  1 1 1 0 1

$N=7$  1 1 1 0 0 1 0

$N=11$  1 0 1 1 0 1 1 1 0 0 0

$N=13$  1 1 1 1 1 0 0 1 1 0 1 0 1

This block can accept an external clock or operate from an internal source. A clock value greater than 0.5 is considered high.

$x_1$  = Optional external clock

$y_1$  = Barker code sequence

$y_2$  = Output clock pulses

### Sequence Length

Specifies the length  $N$  of Barker code. Available lengths include 3, 4, 5, 7, 11 and 13. The default value for the 802.11 specification is 11.

### Sequence Offset

Specifies a starting offset for the Barker sequence. Valid range is 0 to  $N-1$ .

### Output Mode

#### **Bilevel**

The signal amplitudes associated with the output sequence are  $\{-1, 1\}$ .

#### **Binary**

The signal amplitudes associated with the output sequence are  $\{0, 1\}$ .

### Timing

#### **Internal**

Indicates internal clock timing. The bit rate and start time need to be specified.

#### **External**

Indicates external timing. An external clock must be provided at the x1 input.

### Bit Rate

Specifies the Barker sequence bit rate in bits per second. This parameter is only available when in Internal Timing mode.

### Start Time

Specifies a start time, in seconds, for the Barker sequence. This parameter is only available when in Internal Timing mode.

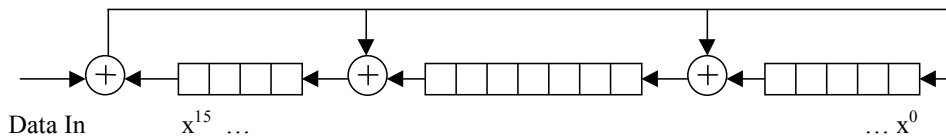
## CRC-16 Generator

This block computes the CCITT-16 CRC (Cyclic Redundancy Code) based on an input binary serial data stream. At each input clock pulse, the block will update the CRC output vector, which is comprised of 16 elements ( $x^{15}, x^{14}, \dots, x^1, x^0$ ). The output can be provided as is, or flipped (one's complement mode).

A CRC is commonly used to verify that a data sequence is received error free by including its "state" at the end of a data packet. If the CRC computed at the receiver (based on the received data) matches the CRC sent by the transmitter, then it's highly likely that the block was received correctly.

The CRC shift register can be reset to its starting value (all 1's) by providing a pulse on the Reset input. The first element of the output vector is the MSB of the CRC ( $x^{15}$  cell). The generator polynomial and internal shift register structure are shown below.

$$p(x) = x^{16} + x^{12} + x^5 + 1$$



$x_1$  = Binary data [0, 1]

$x_2$  = Data clock (pulse train)

$x_3$  = Reset (pulse)

$y_1$  = CRC output vector (size 16)

**Output Mode*****Regular***

The block outputs the internal shift register state as is.

***One's Complement***

The block outputs the one's complement of internal shift register state (all 0's are flipped to 1's and all 1's are flipped to 0). This mode is used in the 802.11 specification.

**802.11 Hop Generator**

This block generates an 802.11 frequency-hopping pattern. This block supports both the 79 and 23 hop specifications, depending on the specified Operating Region. For specific details on the 802.11 hop sequences, please refer to Section 14.6.4 – 14.6.8 of the 802.11 specification.

This block outputs a hopping pattern represented by a hop index (for driving a Frequency Hop block) and a Channel number for information purposes. The hop index range is [0, 78] or [0, 22] depending on the Hop Mode selection. This block support both internal and external timing. When in external mode, the user must provide a pulse train indicating when the next hop slot is starting.

Note: the output Hop Index specifies the hop operating frequency using the standard 1 or 2 Mbps (Low Rate) channel assignments beginning with index value 0. For example, Channel #37 (centered at 2437 MHz) will be output as Hop Index #35, since channel numbers start with Channel #2 (2402 MHz).

This block should be followed by a Frequency Hop block to implement the actual hopping (use the Hop Index output as the drive signal).

$x_1$  = Pattern selection input

$x_2$  = Optional external hop clock (pulse train)

$y_1$  = Hop Index # [0, 78] or [0, 22]

$y_2$  = Output hop clock pulse

$y_3$  = Hop Channel # [2, 80] or [2, 24]

**Region**

Specifies the 802.11 operating geographical region. Choices include North America/Canada, Europe, Japan, Spain and France.

**Timing Mode*****Internal***

Indicates internal clock timing.

***External***

Indicates external timing. An external clock signal must be provided.

**Initial Hop Index**

Specifies the initial internal counter value for the 802.11 hopping pattern.

**Hop Rate**

Specifies the hop rate for the block in hops per second. This parameter is only available when in Internal Timing mode. The default hop rate is TBD kHz.

**Start Time**

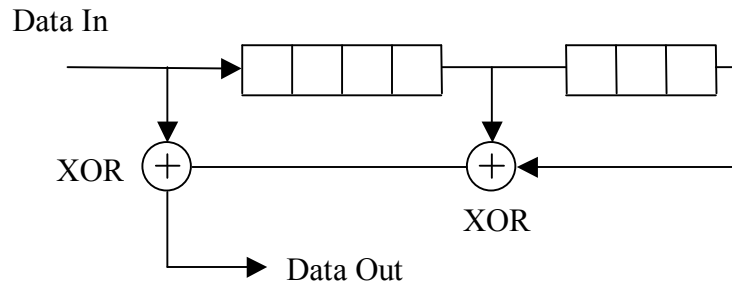
Specifies a starting time for the hop sequence. This parameter is only available when in Internal Timing mode.

## 802.11 Descrambler

This block provides a data descrambling function as specified in the 802.11 and 802.11b standards. The input data is XOR'ed with the output sequence from a feedback shift register of size  $N = 7$ . The shift register is re-initialized whenever an external pulse is presented at the frame clock input. Depending on the operating mode – either long or short PLCP preamble – the shift register is initialized with either hex 0x6C (1101100) or 0x1B (0011011) respectively, with the LSB corresponding to the rightmost shift register cell. The feed-through implementation on the scrambler and descrambler is self-synchronizing, which does not require any knowledge of the initial Tx shift register state for receive processing.

The generator polynomial for the feedback shift register is:

$$p(x) = x^7 + x^4 + 1$$



The user must supply an input bit stream and clock (pulse train) and periodically a frame clock pulse to re-initialize the internal shift register when desired.

$x_1$  = Input data

$x_2$  = Input data clock pulses

$x_3$  = Reset clock pulse

$y_1$  = Output data

$y_2$  = Output data clock pulses

### Register Init Value

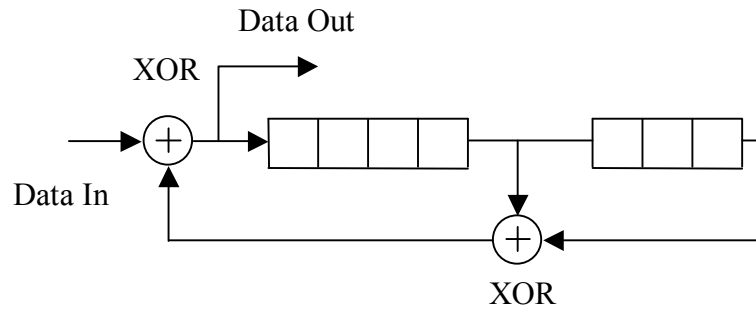
Specifies the initial value of the shift register in hex format. Since the descrambler is self-synchronizing, providing an initial value simply provides a means of achieving an immediate synchronization with the transmitter, instead of having to wait for seven clock cycles.

## 802.11 Scrambler

This block provides a data scrambling function as specified in the 802.11 and 802.11b standards. The input data is XOR'ed with the output sequence from a feedback shift register of size  $N = 7$ . The shift register is re-initialized whenever an external pulse is presented at the frame clock input. Depending on the operating mode – either long or short PLCP preamble – the shift register is initialized with either hex 0x6C (1101100) or 0x1B (0011011) respectively, with the LSB corresponding to the rightmost shift register cell.

The generator polynomial for the feedback shift register is:

$$p(x) = x^7 + x^4 + 1$$



The user must supply an input bit stream and clock (pulse train) and periodically a frame clock pulse to re-initialize the internal shift register when desired.

$x_1$  = Input data

$x_2$  = Input data clock pulses

$x_3$  = Frame clock pulse

$y_1$  = Output data

$y_2$  = Output data clock pulses

### Register Initialization

#### **Long Preamble**

The register is initialized to hex 0x6C as specified in the 802.11 specification for use with the long preamble.

#### **Short Preamble**

The register is initialized to hex 0x1B as specified in the 802.11 specification for use with the short preamble.

## GFSK-2 Modulator

This block implements a two level Gaussian Frequency Shift Keying (GFSK) modulator as a compound block. In GFSK-2 modulation, the digital information is transmitted by shifting the carrier frequency between two states. The 802.11 GFSK-2 mode specifies the following frequency deviations:

<i>Input Symbol</i>	<i>Carrier Deviation</i>
1	$1/2 \times h_2 \times R = 160 \text{ kHz}$ (for $h_2 = 0.32$ and $R = 1 \text{ Msps}$ )
0	$-1/2 \times h_2 \times R = -160 \text{ kHz}$

This block employs a Gaussian FIR Filter and an FM Modulator as its internal components. The internal parameters of these blocks may need to be adjusted for proper operation, depending on the chosen data rate and simulation rate. The default parameters for the BT product, symbol rate, and FM deviation value reflect the 802.11 specification for the 1 Mbps mode, which specifies a BT value of 0.5 and a nominal frequency deviation +/- 160 kHz.

$x$  = Input data signal (binary [0, 1])

$y$  = Complex output signal [Re, Im]

## GFSK-4 Modulator

This block implements a four level Gaussian Frequency Shift Keying (GFSK) modulator as a compound block. In GFSK-4 modulation, the digital information is transmitted by shifting the carrier frequency between four states. The 802.11 GFSK-4 mode specifies the following frequency deviation values depending on the input data symbol (first received bit is the LSB):

<i>Input Symbol</i>	<i>Carrier Deviation</i>
---------------------	--------------------------

1 0	$3/2 \times h4 \times R = 216 \text{ kHz}$	(for $h4= 0.144$ and $R = 1 \text{ Msps}$ )
1 1	$1/2 \times h4 \times R = 72 \text{ kHz}$	
0 1	$-1/2 \times h4 \times R = -72 \text{ kHz}$	
0 0	$-3/2 \times h4 \times R = -216 \text{ kHz}$	

This block employs a Gaussian FIR Filter and an FM Modulator as its internal components. The internal parameters of these blocks may need to be adjusted for proper operation, depending on the chosen data rate and simulation rate. The default parameters for the BT product, symbol rate, and FM deviation value reflect the 802.11 specification for the 2 Mbps mode, which specifies a BT value of 0.5 and the frequency deviations shown above.

$x$  = Input data signal (symbol value [0, 1, 2, 3] )

$y$  = Complex output signal [Re, Im]

## 802.11a/g BLOCKS

### 802.11a/g Convolutional Encoder

This implements an 802.11a/g rate  $1/2$  convolutional encoder. The block's settings allow specification of the generator coefficients and constraint length, but the default values correspond to those of the 802.11a/g specification ( $k=7$ ,  $G1= 133$ ,  $G2= 171$  octal). All other code rates other than rate  $1/2$  are achieved by puncturing the output of this block (see Puncture block).

At each input clock pulse, the blocks internal shift register is shifted to the right and a new input data bit is read. The convolutional encoder's A and B outputs are then updated.

$x_1$  = Input data bits [0, 1]

$x_2$  = Input clock (pulse train)

$y_1$  = A output

$y_2$  = B output

$y_3$  = Output clock pulse

#### Generator #1

Specifies the value of the first generator coefficient in octal format. The default is 133, which corresponds to "1011011" in binary.

#### Generator #2

Specifies the value of the first generator coefficient in octal format. The default is 171, which corresponds to "1111001" in binary.

#### Constraint Length

Specifies the length of the internal shift register in bits.

### 802.11a/g Depuncture

This block converts a received serial bit stream into pairs (A B) of data suitable for use by the 802.11a/g Viterbi Decoder block. For code rates other than rate  $1/2$ , this block also provides a depuncture function, which inserts dummy bits (erasures) in the locations where bits were "stolen" at the encoder side.

Knowledge of the output symbol rate (A and B pairs) is required to so that the block can properly output the depunctured bit pairs. The output data stream is always re-synched with the input serial clock at the start of each new pattern period (every 2 input clock pulses for a rate  $1/2$  code; every 3 input clock pulses for rate  $2/3$ ; and every 4 input clock pulses for rate  $3/4$ ). For further details on code puncturing see the description of the 802.11a/g Puncture block.

$x_1$  = Input bits

$x_2$  = Input clock (pulse train)

$y_1$  = A output  
 $y_2$  = B output  
 $y_3$  = Output clock (pulse train)

**Code Rate**

Specifies the desired code rate as Rate 1/2, Rate 2/3 or Rate 3/4.

**Output Symbol Rate**

Specifies the output symbol rate (A B pairs) so that the block can properly output the depunctured bit pairs. This value should match the symbol rate used at the encoder block.

**Erasure Value**

Specifies the received value to be used for the dummy bits inserted at the erasure locations. This value is typically set to 0 since a soft-decision Viterbi block (which usually follows this block) is typically set to operate with bi-level data [-1, +1].

**802.11a/g Puncture**

This block is used to convert the output of the 802.11a/g Convolutional Encoder into a serial bit stream and also implement puncturing when the selected code rate is not rate 1/2. Puncturing is implemented by “stealing” specific coded bits from the encoder output (i.e. not transmitting such bits). At the receiver, these stolen bit slots are filled with dummy bit values.

Knowledge of the input symbol rate (clock rate) is required so that the block can derive the appropriate output serial timing. This block does not require that the output bit interval be comprised of an integral number of simulation samples. As a result, the duration of individual output bits can differ slightly from bit to bit within the underlying “pattern period”, which is defined as the periodic time interval (in input clock cycles) required to implement each specific puncture pattern. The output stream is always re-synched with the input clock at the start of each new pattern period.

At each input clock new A and B value are read, and represented herein by increasing subscripts (i.e.  $A_0 B_0$ ,  $A_1 B_1$ ,  $A_2 B_2$ ). The following describes the puncture pattern for each code rate (i.e. which bits are output).

*Rate 1/2:*

Pattern Period= 1 clock cycle      Output pattern:  $A_0 B_0$

*Rate 2/3:*

Pattern Period= 2 clock cycles      Output pattern:  $A_0 B_0 A_1$

*Rate 3/4:*

Pattern Period= 3 clock cycles      Output pattern:  $A_0 B_0 A_1 B_2$

$x_1$  = Input A value  
 $x_2$  = Input B value  
 $x_3$  = Input clock pulse  
 $y_1$  = Serial punctured output  
 $y_2$  = Output clock (pulse train)

**Code Rate**

Specifies the desired code rate as Rate 1/2, Rate 2/3 or Rate 3/4.

**Input Symbol Rate**

Specifies the approximate input symbol rate (A B pairs) so that the block can properly compute the output serial timing for the selected code rate.

## 802.11a/g Interleaver

This implements an 802.11a/g Interleaver or Deinterleaver. The block size of the interleaver is set to correspond to the number of coded bits in a single OFDM frame, depending on the intended rate of the link per Section 17.3.5.6 of the 802.11a/g specification. This block is normally inserted after the convolutional encoding process.

This block basically implements a “block type” interleaver, and introduces a delay equal to the block size (i.e. number of coded bits in the OFDM frame). The block size is set as follows depending on the selected Rate Mode:

6 Mbps, 9 Mbps:	48 bits
12 Mbps, 18 Mbps:	96 bits
24 Mbps, 36 Mbps:	192 bits
48 Mbps, 54 Mbps:	288 bits

$x_1$  = Input data

$x_2$  = Input clock (pulse train)

$y_1$  = Output data

$y_2$  = Output clock (pulse train)

### 802.11a Rate Mode

Specifies the data rate of the 802.11a/g link being simulated (not necessarily the simulation rate).

#### Mode

##### ***Interleave***

The block operates as an interleaver.

##### ***Deinterleave***

The block operates as a deinterleaver.

## 802.11a/g Scrambler

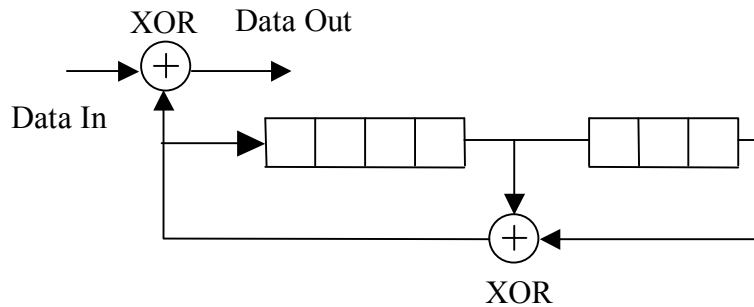
This block provides a data scrambling/descrambling function as specified in the 802.11a/g standard. The input bits are XOR’ed with the output sequence of a feedback shift register of size  $N = 7$ , which produces a repeating pattern of length 127.

The internal shift register is re-initialized whenever an external pulse is presented at the reset input. Depending on the operating mode – either internal or external initialization – the shift register is then initialized with either a fixed hex value or an externally provided value, where the LSB corresponds to the rightmost shift register cell in the diagram below. The initialization value is typically set to a pseudo-random non-zero value.

Note: when this block is used to scramble the 802.11a/g OFDM pilot tones, the shift register is typically initialized to all 1’s.

The generator polynomial for the feedback shift register is:

$$p(x) = x^7 + x^4 + 1$$



- $x_1$  = Input bits
- $x_2$  = Input clock (pulse train)
- $x_3$  = Register initialization value (external mode only)
- $x_4$  = Reset input (pulse)
- $y_1$  = Output data
- $y_2$  = Output clock (pulse train)

### Register Initialization

#### **External**

The register is initialized to the current value presented at the external “Init” input.

#### **Internal**

The register is initialized to the hex value specified in the Register Init Value parameter field.

### Register Init Value

Specifies the reset value for the internal shift register when in Internal register initialization mode. This value is specified in hex format.

## 802.11a/g Viterbi Decoder

This block implements a rate  $\frac{1}{2}$  soft decision Viterbi Decoder compatible with the 802.11a/g specification. This block is used to decode a convolutionally encoded bit stream. Parameters include the encoder constraint length ( $k$ ), the trellis truncation length  $M$ , the number of quantization bits, and the generator coefficients. An external Metric file must also be specified. This block accepts soft A and B outputs from a 802.11a/g Depuncture block, which should be in bilevel format (i.e. +/- 1 without any noise).

- $x_1$  = A Input  $\sim[-1, +1]$
- $x_2$  = B Input  $\sim[-1, +1]$
- $x_3$  = Input clock (pulse train)
- $y_1$  = Decoded bit stream (0, 1)
- $y_2$  = Output clock (pulse train)
- $y_3$  = Best path metric

### Generator #1

Specifies the value of the first generator coefficient in octal format. The default is 133, which corresponds to “1011011” in binary.

### Generator #2

Specifies the value of the first generator coefficient in octal format. The default is 171, which corresponds to “1111001” in binary.

**Constraint Length**

Specifies the constraint length  $k$  of the associated encoder.

**Trellis Truncation Length**

Specifies the trellis truncation length  $M$ . The maximum allowed value of  $M$  is 96.

**Quantization Bits**

Specifies the number of quantization bits used in the decoding process.

**Select File**

Opens the Select File dialog box for selecting the desired Metric file.

**Browse File**

Opens the selected Metric file using Notepad.

**Metric File Path**

Specifies the path and filename of the metric file to be used in decoding. The metric file format is described below:

*Header line (can be anything)*

$Q$              $m(A/0)$     $m(A/1)$

$T_{AB}$          $m(B/0)$     $m(B/1)$

$T_{BC}$          $m(C/0)$     $m(C/1)$

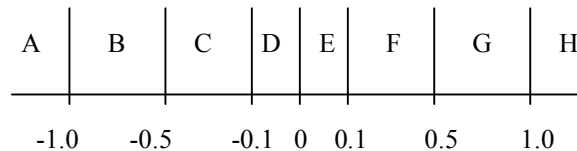
...            ...        ...

$Q$  = # quantization bits

$T_{AB}$  = threshold value between regions "A" and "B"

$m(C/1)$  = metric for region "C" given a "1" was sent

**Note:** File entries can be separated by a comma, tab, or whitespace. A metric value of 0 is considered best.

**Example Metric File**

Viterbi Decoder metric file for use with erasures (0), Quant bits = 3

3,            0.0,    7.0

-1.0,        1.0,    6.0

-0.5,        2.0,    5.0

-0.1,        3.0,    3.0

0,            3.0,    3.0

0.1,         5.0,    2.0

0.5,         6.0,    1.0

1.0,         7.0,    0.0

For example:

for an input  $0.5 \leq x < 1.0$ ,

the metric for a "0" bit is "6.0",

and the metric for a "1" bit is "1.0".

**Note:** In the above example, values in the range of  $[-0.1, 0.1]$ , which includes the erasure value (dummy bit) 0, are given an equal metric.

## OFDM Demodulator

This block demodulates a baseband Orthogonal Frequency Division Multiplexed (OFDM) modulated signal back to a group of data and pilot subcarriers, as specified in the IEEE 802.11a/g standard. For more details on OFDM, please refer to the OFDM Modulator block description.

The OFDM signal is demodulated by applying a forward FFT to the received I and Q complex baseband inputs. An external trigger pulse is used to signal the beginning of each received frame to the block.

Since a Guard Interval (GI) may be used in the transmission of the OFDM frame, this block allows the user to specify an FFT wraparound value to be applied to the received vector prior to performing the FFT. This feature is useful when the active samples used by the OFDM Demodulator include samples from the GI (instead of just FFT segment samples). Since the GI is generated from a cyclic shift of the FFT segment, this feature allows the OFDM Demodulator block to recover the proper IFFT samples.

The OFDM Demodulator block accepts a baseband OFDM modulated signal in I and Q component format. Other inputs include a sampling clock (pulse train) and a trigger used to signal the beginning of each OFDM frame. Block outputs include a frame clock and two data vectors representing the Real and Imaginary components of the recovered “subcarrier” data. This block can be followed by an OFDM Pilot Extract block to separate-out the data and pilot subcarriers.

$x_1$  = Beginning of frame trigger

$x_2$  = I baseband input

$x_3$  = Q baseband input

$x_4$  = Input sampling clock (pulses)

$y_1$  = Frame clock pulse

$y_2$  = Real output subcarrier vector (size  $2^N$ )

$y_3$  = Imaginary output subcarrier vector (size  $2^N$ )

### FFT Size

Specifies the size of the forward FFT (power of two) to be performed on the received data. Valid range is from 16 to 64K. This value should match the IFFT size used by the transmitter.

### FFT Wraparound Offset

Specifies a cyclic offset to be applied to the received data prior to performing the FFT operation. This parameter is useful when samples from the GI are included in the received frame.

## OFDM Modulator

This block generates a baseband Orthogonal Frequency Division Multiplexed (OFDM) modulated signal, as used in the IEEE 802.11a/g specification.

In OFDM modulation, the output is obtained by applying an Inverse FFT (typically a power of two) to an input vector of complex IQ points corresponding to data and pilot “subcarriers”. To reduce ISI, the inverse FFT output is often padded with an optional guard interval representing a cyclic shift of the inverse FFT output itself. The resulting OFDM “symbol” is then treated as a baseband time domain signal and used for transmission.

This block accepts two input vectors representing the Real and Imaginary components of the subcarrier data. An OFDM Pilot Map block can be used to merge the data subcarriers with any pilot or null subcarriers. The input vectors are read when a frame clock pulse is detected. Additional inputs include a guard interval selector, an extended FFT period flag, and an optional external clock for the output data. Block outputs include time domain baseband I and Q outputs representing the OFDM symbol, and an output sample clock.

The external Guard Interval (GI) selection input allows the user to vary the guard interval size during the simulation. The following describes the GI corresponding to each allowed input value:

- 0 → No GI
- 1 → GI = FFT Size / 2
- 2 → GI = FFT Size / 4
- 3 → GI = FFT Size / 8
- 4 → GI = FFT Size / 16
- 5 → GI = FFT Size / 32

The size of the output frame is equal to GI + FFT size, unless the extended FFT input is set high. The optional extended FFT period connector is used to signal the OFDM Modulator to output the FFT segment twice (i.e. the output equals GI + FFT + FFT).

The OFDM Modulator also supports windowing between successive OFDM frames to reduce spectral sidelobes. The user can specify an overlap ranging from 0 to 3 samples. The window function follows a  $\sin^2$  rolloff profile as specified in the 802.11a/g specification.

In this VisSim implementation a windowed overlap is generated by extending the GI at the beginning of the frame by a few samples, and carrying over the last few samples of the FFT section to the next output frame. For example, if an overlap of 2 samples is used, the GI length is increased by two samples (to which a rolloff is applied) and overlapped with the last two samples (also weighted by a rolloff) from the previous frame's FFT section. This implementation reduces simulation delay and internal buffering needs.

- $x_1$  = Input Frame clock pulse
- $x_2$  = Real input vector (size  $2^N$ )
- $x_3$  = Imaginary input vector (size  $2^N$ )
- $x_4$  = Guard Interval selector {0, 1, 2, 3, 4, 5}
- $x_5$  = Extended FFT period flag (High = extend FFT period)
- $x_6$  = Optional external sample clock
- $y_1$  = I signal output
- $y_2$  = Q signal output
- $y_3$  = Output clock pulses

### Timing

#### **Internal**

Indicates internal clock timing per the specified output rate.

#### **External**

Indicates external timing. An external clock must be provided at the  $x_6$  input.

### Use Simulation Sample Rate

When selected, this mode forces the OFDM Modulator block to use the current simulation sample rate as its output rate.

### Output Rate

Specifies the output rate for the OFDM output samples. This parameter is only available when in Internal Timing mode. The default value is set to the current simulation rate.

### FFT Size

Specifies the size of the inverse FFT (power of two) to be performed. Valid range is from 16 to 64K.

**Overlap Size**

Specifies an overlap between successive OFDM output frames. Specifying an overlap results in a windowing operation and provides a smoother transition from frame to frame. Valid range is from 0 to 3 samples.

**OFDM Pilot Extract**

This block decomposes the Re/Im output of an OFDM Demodulator into two groups of subcarrier vectors (Data and Pilot) according to a user defined mapping file.

The subcarrier mapping file should match the one used by the OFDM transmitter, and serves to specify the ordering in the received vector of the data, pilot and any null subcarriers (which are discarded).

The user must specify the number of data and pilot subcarriers, as well as the input vector size, which must be a power of two (FFT size). The user can also specify the starting position within the input vector for element #0 in the map file as either 0 or  $N/2$ , where  $N$  is the FFT size. The Data and Pilot subcarrier values are written in sequential order to their respective outputs according to their order of appearance in the Map File.

For more details on the Map File format, please refer to the OFDM Pilot Map block description.

$x_1$  = Input Frame clock pulse

$x_2$  = Real output vector from OFDM Demod (size  $2^N$ )

$x_3$  = Imaginary output vector from OFDM Demod (size  $2^N$ )

$y_1$  = Frame clock pulse

$y_2$  = Data subcarrier I vector

$y_3$  = Data subcarrier Q vector

$y_4$  = Pilot subcarrier I vector

$y_5$  = Pilot subcarrier Q vector

**Input Insertion Point*****Begin at N/2 Location***

Forces an  $N/2$  circular shift in the input vector, where  $N$  is the size of the FFT used by the OFDM Demodulator. Element zero in the Map File is read from the  $N/2$  input vector location.

***Begin at 0 Location***

Element zero in the Map File is read from the 0 input vector location.

**Number of Data Subcarriers**

Specifies the size of the Data subcarrier I and Q output vectors.

**Number of Data Subcarriers**

Specifies the size of the Pilot subcarrier I and Q output vectors.

**Data Subcarrier Weight**

Specifies the weight applied to the I and Q Data subcarriers in the modulator. The received subcarrier is scaled (divided) by this value.

**Pilot Subcarrier Weight**

Specifies the weight applied to the I and Q Pilot subcarriers in the modulator. The received subcarrier is scaled (divided) by this value.

**FFT Size**

Specifies the block's input vector size  $N$ . This value should match the size of the FFT used by the preceding OFDM Demodulator, as well as the Map File's number of entries. Valid range is from 16 to 64K.

**Select File**

Launches a dialog box for selecting the desired Subcarrier Map file.

**Browse File**

Opens the selected Subcarrier Map file using Notepad.

**Subcarrier Mapping File**

Specifies the path and filename for the Subcarrier Map file. The format of the generated file is shown below. The keyword identifier specifies the subcarrier type to be used for each element. Allowed keywords are: DATA, PILOT, and NULL. Valid entry separators are commas, tabs and blank space.

*Header line (for storing user defined data)*

*element #0            keyword*

*element #1            keyword*

*...*

*element #N-1        keyword*

**OFDM Pilot Map**

This block generates a composite Re/Im pair of "subcarrier" vectors for use by an OFDM Modulator or Vector OFDM Modulator block.

This block accepts two pairs of input vectors (each with Real and Imaginary components) representing data subcarriers and pilot subcarriers. A user defined data mapping file is used to specify the ordering in the output vector of the data, pilot and any null subcarriers.

The user must specify the number of data and pilot subcarriers, as well as the output vector size, which must be a power of two (FFT size). The user can also specify the starting position within the output vector for the input data as either 0 or  $N/2$ , where  $N$  is the FFT size. The Data and Pilot subcarrier values are read in sequential order from their respective input ports and placed in the output vector according to the output ordering specified by the Map File.

The format of the "Map File" is as follows. The number of entries in the file should match the intended OFDM FFT size. A one line header is assumed. Thereafter, the first entry of each row represents the output vector element #, and is followed by a keyword identifying whether that location is to contain a data subcarrier (DATA), a pilot subcarrier (PILOT), or a null subcarrier (NULL). Any additional characters on the line are ignored. If the "Begin at  $N/2$ " option is selected, then the output vector is cyclically shifted by  $N/2$  before being output (i.e. the entry for the #0 element will be output at the  $N/2$  vector location).

$x_1$  = Input Frame clock pulse

$x_2$  = Data subcarrier I vector

$x_3$  = Data subcarrier Q vector

$x_4$  = Pilot subcarrier I vector

$x_5$  = Pilot subcarrier Q vector

$y_1$  = Frame clock pulse

$y_2$  = Output vector (Real component) (size  $2^N$ )

$y_3$  = Output vector (Imaginary component) (size  $2^N$ )

**Input Insertion Point*****Begin at N/2 Location***

Forces an  $N/2$  circular shift in the output vector, where  $N$  is the size of the FFT used by the OFDM Modulator. Element zero from the Map File is output in the  $N/2$  output vector location.

***Begin at 0 Location***

Element zero from the Map File is output in the 0 output vector location.

**Number of Data Subcarriers**

Specifies the size of the Data subcarrier I and Q input vectors.

**Number of Pilot Subcarriers**

Specifies the size of the Pilot subcarrier I and Q input vectors.

**Data Subcarrier Weight**

Specifies a weight to be applied to the I and Q Data subcarriers.

**Pilot Subcarrier Weight**

Specifies a weight to be applied to the I and Q Pilot subcarriers.

**FFT Size**

Specifies the block's output vector size  $N$ . This value should match the size of the FFT used by the OFDM Modulator that will follow, as well as the Map File's number of entries. Valid range is from 16 to 64K.

**Select File**

Launches a dialog box for selecting the desired Subcarrier Map file.

**Browse File**

Opens the selected Subcarrier Map file using Notepad.

**Subcarrier Mapping File**

Specifies the path and filename for the Subcarrier Map file. The format of the generated file is shown below. The keyword identifier specifies the subcarrier type to be used for each element. Allowed keywords are: DATA, PILOT, and NULL. Valid entry separators are commas, tabs and blank space.

*Header line (for storing user defined data)*

*element #0            keyword*

*element #1            keyword*

*...*

*element #N-1        keyword*

**OFDM Vector Demodulator**

This block demodulates a baseband Orthogonal Frequency Division Multiplexed (OFDM) modulated signal, as used in the IEEE 802.11a/g specification. For more details on OFDM, please refer to the OFDM Modulator block description.

This block differs from the regular OFDM Demodulator block in that it uses vector inputs for the I and Q received signals, instead of scalars. The OFDM signal is demodulated by applying a forward FFT to the received I and Q complex baseband samples. An external frame clock pulse is used to signal each new OFDM symbol.

Since a Guard Interval (GI) may be used in the transmission of the OFDM symbol, this block allows the user to specify a limited FFT offset value to be applied to the received vector prior to performing the FFT. This feature is useful when the input vector used by the Vector OFDM Demodulator includes some GI samples (instead of just FFT segment samples). Since the GI is generated from a cyclic shift of the FFT segment, this feature allows the Vector OFDM Demodulator block to recover the proper IFFT samples.

The Vector OFDM Demodulator block accepts a baseband OFDM modulated signal in I and Q vector format and a frame (symbol) clock. Block outputs include a frame clock and two data vectors representing the Real and Imaginary components of the recovered “subcarrier” data. This block can be followed by an OFDM Pilot Extract block to separate-out the data and pilot subcarriers.

$x_1$  = Input frame clock

$x_2$  = I baseband vector (size  $2^N$ )

$x_3$  = Q baseband vector (size  $2^N$ )

$y_1$  = Frame clock pulse

$y_2$  = Real output data vector (size  $2^N$ )

$y_3$  = Imaginary output data vector (size  $2^N$ )

#### **FFT Size**

Specifies the size of the forward FFT (power of two) to be performed on the received data. Valid range is from 16 to 64K. This value should match the IFFT size used by the transmitter.

#### **FFT Offset**

Specifies a cyclic offset to be applied to the received data vector to performing the FFT operation. This parameter is useful when samples from the GI are included in the received frame. Valid range is 0 to 3.

### **OFDM Vector Modulator**

This block generates a baseband Orthogonal Frequency Division Multiplexed (OFDM) modulated signal, as used in the IEEE 802.11a/g specification.

This block differs from the regular OFDM Modulator block in that it uses vector outputs for the I and Q signals, instead of scalars. This version of the block does not support varying the Guard Interval (GI) size during a simulation, nor does it support the extended FFT mode. For more details on OFDM, please refer to the OFDM Modulator block description.

This block accepts two input vectors representing the Real and Imaginary components of the subcarrier data. An OFDM Pilot Map block can be used to merge the data subcarriers with any pilot or null subcarriers. The input vectors are read when a frame clock pulse is detected. Block outputs include I and Q vector OFDM outputs, and an output frame clock. The size of the output vector is equal to GI + FFT size. The GI size can range from none to 1/32 of the FFT size.

The Vector OFDM Modulator supports windowing between successive OFDM frames to reduce spectral sidelobes. The user can specify an overlap ranging from 0 to 3 samples. The window function follows a  $\sin^2$  rolloff profile as specified in the 802.11a/g specification.

In this VisSim implementation, a windowed overlap is generated by extending the GI at the beginning of the frame by a few samples, and carrying over the last few samples of the FFT section to the next output frame. For example, if an overlap of 2 samples is used, the GI length is increased by two samples (to which a rolloff is applied) and overlapped with the last two samples (also weighted by a rolloff) from the previous frame’s FFT section. This implementation reduces simulation delay and internal buffering needs.

$x_1$  = Input Frame clock pulse

$x_2$  = Real input vector (size  $2^N$ )

$x_3$  = Imaginary input vector (size  $2^N$ )  
 $y_1$  = Output frame clock pulse  
 $y_1$  = I signal output  
 $y_2$  = I signal output vector (size GI + FFT)  
 $y_3$  = Q signal output vector (size GI + FFT)

**FFT Size**

Specifies the size of the inverse FFT (power of two) to be performed. Valid range is from 16 to 64K.

**Overlap Size**

Specifies an overlap between successive OFDM output frames. Specifying an overlap results in a windowing operation and provides a smoother transition from frame to frame. Valid range is from 0 to 3 samples.

**Guard Interval Size**

Specifies the size of the desired Guard Interval as a fraction of the IFFT size. Choices include: None,  $\frac{1}{2}$ ,  $\frac{1}{4}$ ,  $\frac{1}{8}$ ,  $\frac{1}{16}$  and  $\frac{1}{32}$  of the IFFT size.

**802.11b BLOCKS****802.11b High Rate Hop Generator**

This block generates an 802.11b frequency-hopping pattern optionally used with the High Rate modes (5.5 and 11 Mbps). This block supports both the Set 1 and Set 2 channel selections for either North America or Europe (except France and Spain).

This block supports either internal or external hop timing. When in external mode, the user must provide an input pulse train indicating when the next hop slot is starting. This block outputs a hop pattern consisting of a hop index (for driving a Frequency Hop block) and either a High Rate or Low Rate channel number as appropriate for information purposes. A High/Low flag is used to characterize the channel number as this can vary during the hop process. High Rate channels can range from 1 to 13, while Low Rate channels range from 2 to 80.

*North America Set 1*

Set 1 hop channels include High Rate Channels 1, 6 and 11.

For Set 1 only two hop patterns are specified:

Pattern#1: 1, 6, 11, 1, 6, 11...

Pattern#2: 1, 11, 6, 1, 11, 6...

*North America Set 2*

Set 2 hop channels include High Rate Channels 1, 3, 5, 7, 9 and 11.

For Set 2 the input pattern number AND the current Hop Index determine the next hop channel number according to section 18.4.6.7.3 of the 802.11b specification.

*Europe Set 1 (except France and Spain)*

Set 1 hop channels include High Rate Channels 1, 7 and 13.

For Set 1 only two hop patterns are specified:

Pattern#1: 1, 7, 13, 1, 7, 13...

Pattern#2: 1, 13, 7, 1, 13, 7...

*Europe Set 2 (except France and Spain)*

Set 2 hop channels include High Rate Channels 1, 3, 5, 7, 9, 11 and 13.

For Set 2 the input pattern number AND the current Hop Index determine the next hop channel number according to section 18.4.6.7.3 of the 802.11b specification.

This block should be followed by a Frequency Hop block to implement the actual hopping (use the Hop Index output as the drive signal).

**Note:** the output Hop Index specifies the hop operating frequency using the Low Rate (1, 2 Mbps) channel assignments beginning with index value 0. For example High Rate channel #6, centered at 2437 MHz and corresponding to Low Rate Channel #37, will be output as Hop Index #35. The offset value of two is due to the fact that the 802.11 channel numbers start with Channel #2 (2402 MHz).

$x_1$  = Pattern number selection input

$x_2$  = Optional external hop clock (pulse train)

$y_1$  = Hop Index # [0, 78]

$y_2$  = Output hop clock pulse

$y_3$  = Hop Channel # (see notes above)

$y_4$  = High/Low Rate flag

### Region

Specifies the 802.11b operating geographical region. Choices include North America/Canada, and Europe (except Spain and France).

### Timing Mode

#### *Internal*

Indicates internal hop timing.

#### *External*

Indicates external timing. An external clock signal (pulse train) must be provided to indicate each hop.

### Hop Set

Specifies which hop set to use; either Set 1 or Set 2.

### Initial Hop Index

Specifies the initial internal counter value for the 802.11b hopping pattern.

### Hop Rate

Specifies the hop rate for the block in hops per second. This parameter is only available when in Internal Timing mode. The default hop rate is TBD kHz.

### Start Time

Specifies a starting time for the hop sequence. This parameter is only available when in Internal Timing mode.

## CCK Demodulator

This block provides a symbol detection function for Complementary Code Keying (CCK) modulated signals. CCK modulation, as defined in the IEEE 802.11b specification, is used to provide a High Rate mode operating at either 5.5 Mbps or 11 Mbps, which is compatible with the pre-existing lower rate data modes (1 and 2 Mbps) of the 802.11 standard. The CCK chips are modulated using QPSK. For additional details on CCK modulation, see the description of the CCK Modulator.

This block accepts a complex baseband input and a chip clock, and produces a decoded bits vector of size 4 or 8 (depending on the CCK mode). The received sequence is compared internally with the CCK waveform set, and the one with maximum correlation is selected as the best estimation of the transmitted signal.

Each time eight new chips are received, a new output vector and a frame clock pulse are presented at the block's output. An external reset capability for the internal chip counter is also provided for synchronization purposes.

- $x_1$  = Complex baseband input (Re, Im)
- $x_2$  = Chip clock (pulse train)
- $x_3$  = Reference Phase {0, 1, 2, 3}
- $x_4$  = Ref. Phase initialization strobe (pulse)
- $x_5$  = Frame counter Reset
- $y_1$  = Output frame clock pulse
- $y_2$  = Output data bits vector (size 4 or 8)
- $y_3$  = Internal "C" chip vector (size 8) {0, 1, 2, 3}

**CCK Mode**

**(8, 4) CCK**

Specifies (8, 4) CCK mode, as used in the IEEE 802.11b 5.5 Mbps rate.

**(8, 8) CCK**

Specifies (8, 8) CCK mode, as used in the IEEE 802.11b 11 Mbps rate.

**CCK Modulator**

This block generates a complex baseband modulated Complementary Code Keying (CCK) signal. CCK modulation, as defined in the IEEE 802.11b specification, is used to provide a High Rate mode operating at either 5.5 Mbps or 11 Mbps, and which is compatible with the pre-existing lower rate data modes (1 and 2 Mbps) of the 802.11 standard. CCK constructs orthogonality or semi-orthogonality from the input signal vector using an expanded Hadamard transform. The CCK chips are modulated using QPSK.

The block's function can be broken down into two main functions. The first is a mapping (encoding) of the input data vector (size 4 or 8 bits) into an eight element CCK chip vector (available as an external output). The second is the conversion of this "C vector" into a baseband time domain complex IQ signal.

For the 5.5 Mbps specification, 4 input data bits are mapped to 8 CCK chips. In this case, herein referred to as (8, 4) CCK modulation, the input data words are  $\{d_0, d_1, d_2, d_3\}$ , where  $d_0$  comes first in time. CCK modulation maps the data bits into phases  $\{\varphi_0, \varphi_1, \varphi_2, \varphi_3\}$  by pairs.

The first pair  $(d_0, d_1)$  controls the value of  $\varphi_0$ , which is mapped differentially according to the previous data words, with the phase shift defined in Table 1. The remaining phase terms are

defined as  $\varphi_1 = d_2 \times \pi + \frac{\pi}{2}$ ,  $\varphi_2 = 0$ , and  $\varphi_3 = d_3 \times \pi$ .

Dibit pattern (d0, d1)	Even symbols Phase change (+j $\omega$ )	Odd symbols Phase change (+j $\omega$ )
00	0	$\pi$
01	$\pi/2$	$3\pi/2$
11	$\pi$	0
10	$3\pi/2$	$\pi/2$

**Table 1. DQPSK encoding table**

For the 11 Mbps specification, 8 input data bits are mapped to 8 CCK chips. In this case, herein referred to as (8, 8) CCK modulation, the input data words are

$\{d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ , where  $d_0$  comes first in time. The pair  $(d_0, d_1)$  is mapped differentially according to the previous data words as in the CCK (8, 4) case. The remaining pairs  $(d_i, d_{i+1})$  are mapped to the remaining phase terms as defined in Table 2.

Bit pairs $(d_i, d_{i+1})$	Phase
00	0
01	$\pi/2$
10	$\pi$
11	$3\pi/2$

**Table 2. QPSK encoding table**

The resulting four phase terms are then combined as follows to obtain the actual CCK modulated code word, which is comprised of eight “chips”. These 8 chips represent the output of the CCK Encoder block.

$$\begin{aligned} \bar{c} &= \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7\} \\ &= \{e^{j(\varphi_1+\varphi_2+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_2+\varphi_4)}, -e^{j(\varphi_1+\varphi_4)}, \\ &\quad e^{j(\varphi_1+\varphi_2+\varphi_3)}, e^{j(\varphi_1+\varphi_3)}, -e^{j(\varphi_1+\varphi_2)}, e^{j\varphi_1}\} \end{aligned}$$

This block accepts an input vector of either 4 or 8 data bits and outputs a QPSK modulated baseband signal. An auxiliary output is also provided to reveal the values of the eight CCK output chips (C vector), which can assume one of four possible phase states  $\{0, 1, 2, 3\}$ . If desired, the modulated IQ output signal can be translated to a specific carrier frequency by using an IQ Modulator block.

- $x_1$  = Input frame clock (pulse train)
- $x_2$  = Input bits data vector (size 4 or 8)
- $x_3$  = Reference Phase  $\{0, 1, 2, 3\}$
- $x_4$  = Ref. Phase initialization strobe (pulse)
- $x_5$  = External chip clock (pulse train) [optional]
- $y_1$  = I signal output
- $y_2$  = Q signal output
- $y_3$  = Chip clock (pulse train)
- $y_4$  = Internal “C” chip vector  $\{0, 1, 2, 3\}$

### CCK Mode

#### **(8, 4) CCK**

Specifies (8, 4) CCK mode, as used in the IEEE 802.11b 5.5 Mbps rate.

#### **(8, 8) CCK**

Specifies (8, 8) CCK mode, as used in the IEEE 802.11b 11 Mbps rate.

### Timing

#### **Internal**

Indicates internal clock timing per the specified chip rate.

**External**

Indicates external timing. An external clock must be provided at the x5 input.

**Chip Output Rate**

Specifies the output chip rate in bits per second. This parameter is only available when in Internal Timing mode. The 802.11b default is 11 MHz.

**GENERIC WIRELESS BLOCKS****Frequency Hop**

This block imparts frequency hopping to a complex baseband input signal according to the external channel selection input. This block does not affect the amplitude of the input signal and merely provides a frequency translation operation.

$x_1$  = Input signal (complex)

$x_2$  = Hop channel selection input

$y_1$  = Frequency hopped output (complex)

$$f_{out} = x_1 \cdot (f_{min} + x_2 \cdot \Delta f)$$

**Number of Hops**

Specifies the number  $N$  of available hop channels. This value is used internally for range checking purposes only.

**Lowest Hop Channel**

Specifies the carrier frequency corresponding to the lowest hop channel in *hertz*. This corresponds to input channel # 0.

**Hop Carrier Spacing**

Specifies the frequency spacing  $\Delta f$  between adjacent hop channels in *hertz*.

**Initial Phase**

Specifies the initial carrier phase in *degrees*. This option is only available when in Continuous Phase mode.

**Phase Mode****Continuous**

Indicates that the signal phase is continuous across channel hops. This simulates the use of a VCO or NCO in generating the output signal.

**Discontinuous**

Indicates that the signal phase is not continuous across channel hops. This simulates the use of multiple free running oscillators to generate each of the hopping channel carriers.

---

## ULTRAWIDEBAND BLOCKS

### Gated Integrate & Dump

This block provides an Integrate and Dump function where the integration process is only active during a narrow user-defined time “window”. The user may specify both the pulse width of the window as well as its start time relative to the last input “frame clock”.

This block is useful in UWB diagrams when only a narrow time window (out of a larger frame time) is to be processed for pulse detection purposes.

$x_1$  = Input signal

$x_2$  = Frame clock pulse

$y_1$  = Integrated output

$y_2$  = Dump clock

$$y_1(t)_k = \int_a^b x_1(t) \partial t \quad \text{where } a = (t_p)_k + T_{start} \quad b = a + T_{width}$$

and  $(t_p)_k$  = time of kth frame pulse

#### Window Start Time

Specifies the start time in *seconds* of the active integration window referenced to the last input frame clock pulse (input  $x_2$ ).

#### Window Width

Specifies the active time duration in *seconds* of the integration window.

#### Suppress First Output Clock

When this option is selected, the very first input frame clock still internally dumps the I&D block (essentially a reset), but the corresponding output dump pulse is suppressed. This option makes it easier to achieve frame synchronization in some diagrams by making the first output pulse coincide with the first valid I&D dumped value.

#### Output Mode

##### **Continuous**

Indicates that the integrated output is continuously updated. Note: forward (Euler) integration is used, resulting in a one-sample delay between the receipt of an input value and its contribution appearing in the integrated output.

##### **Held**

Indicates that the integrated output is updated with each input frame clock and held constant during the entire frame’s duration. As a result, a one frame period delay is present in the integrated output.

### UWB PPM Modulator

This block implements a Pulse Position Modulator (PPM) suitable for use in UWB simulations. The block operates by delaying an input clock pulse by a variable amount depending on the symbol value of its data input. The user can control the number of pulse position locations and specify an optional guard interval at the end of each data frame.

$x_1$  = Input data symbol  $k$  in range  $[0, N-1]$

$x_2$  = Frame clock pulse

$y_1$  = PPM modulated unit pulse

$y_2$  = Frame clock pulse

$$T_{slot} = \frac{1}{N} \cdot \left( \frac{1}{R_S} - T_g \right) \quad \Delta T(k) = k \cdot T_{slot} \quad k \in \{0, \dots, N-1\}$$

### Number of Levels

Specifies the number  $N$  of desired pulse positions. The input signal should be in the range of  $[0, \dots, N-1]$ .

### Symbol Rate

Specifies the input symbol data rate  $R_s$  in *Hertz*.

### Guard Interval

This value specifies an optional time interval  $T_g$  in *seconds* between the last valid pulse position slot and the beginning of the next symbol frame.

## UWB Pulse

This block generates a wide variety of commonly used ultrawideband pulses, including the Gaussian pulse shape and its time derivatives. The block operates by generating the specified pulse shape each time an impulse is presented at the block's input. The block is also designed to sense the sign of the incoming pulse and produce a same-sign output waveform. For the block to register an input pulse, its amplitude must be larger than 0.5.

User block parameters include the defined pulse duration, pulse width, pulse amplitude and pulse time offset as appropriate. The defined pulse duration refers to the actual time span for which VisSim/Comm computes the pulse shape. Outside of this period, the output is zero. Pulse width is defined differently for each type of pulse shape (see below).

Note: Each time a new input pulse is sensed, the block immediately begins to output the defined pulse shape from its starting position.

This block supports the following pulse shapes:

- Monopulse
- Gaussian Pulse
- Gaussian Monocycle (1<sup>st</sup> derivative of Gaussian pulse)
- Gaussian Doublet (2<sup>nd</sup> derivative of Gaussian pulse)
- Gated Sinusoid

Please note that the list of block parameters associated with each pulse shape vary from type to type. In the section that follows, the block parameters which are common to all the pulse shapes are defined first. The remaining parameters are described under each pulse type.

$x_1$  = Input impulse train signal  $[-1, 1]$

$y_1$  = UWB output

### Pulse Amplitude Mode

#### **Peak Amplitude**

Indicates that pulse amplitude is defined by specifying the max amplitude of the pulse waveform.

**Held**

Indicates that pulse amplitude is defined by specifying the total power in each pulse assuming a theoretical non-time-limited pulse.

**Defined Pulse Duration**

Specifies the time interval in *seconds* for which VisSim/Comm computes the value of the pulse waveform.

**Pulse Amplitude****Pulse Power**

Specifies either the pulse amplitude  $A$  in *Volts* or the pulse power in *dBm* depending on the selected Pulse Amplitude Mode setting.

**Time Units**

Specifies the time units associated with the pulse duration, time offset and pulse width parameters. Choices include *seconds*, *milliseconds*, *microseconds*, *nanoseconds* and *picoseconds*.

**View Pulse Shape**

Invokes the VisSim/Comm Graphics Viewer to display the selected pulse waveform in both the time domain and frequency domain.

**UWB Pulse Type**

Specifies the desired UWB pulse shape. Available choices include: Monopulse, Gaussian Pulse, Gaussian Monocycle, Gaussian Doublet, and Gated Sinusoid. Descriptions for each pulse follow.

**Monopulse**

$$y(t) = A \cdot e^{\left(1 - \frac{(t-t_0)}{\tau}\right)} \quad \text{for } t \geq t_0 \quad y(t) = 0 \quad \text{otherwise}$$

**Time Offset**

Specifies the start time  $t_0$  of the Monocycle pulse relative to the received input pulse time.

**Pulse Width**

Defines the width of the pulse by specifying the time interval  $\tau$  between the start of the pulse and its peak amplitude time.

**Gaussian Pulse**

$$y(t) = A \cdot e^{-\left(\frac{2\sqrt{\ln 2}(t-t_0)}{\tau}\right)^2}$$

**Pulse Peak Time**

Specifies the time offset  $t_0$ , relative to the received input pulse time, when the Gaussian pulse is to reach its peak value.

**Pulse Width**

Defines the width of the pulse by specifying the time interval  $\tau$  between the pulse's half amplitude points.

### **Gaussian Monocycle**

$$y(t) = 2A\sqrt{e} \cdot \frac{(t-t_0)}{\tau} \cdot e^{-2\left(\frac{(t-t_0)}{\tau}\right)^2}$$

#### **Zero Crossing Time**

Specifies the time offset  $t_0$ , relative to the received input pulse time, when the Gaussian monocycle is to cross through zero.

#### **Pulse Width**

Defines the width of the pulse by specifying the time interval  $\tau$  between the peaks of the monocycle waveform.

### **Gaussian Doublet**

$$y(t) = \frac{A}{\tau} \cdot e^{-2\left(\frac{(t-t_0)}{\tau}\right)^2} \cdot \left(1 - \frac{4(t-t_0)}{\tau}\right)$$

#### **Pulse Peak Time**

Specifies the time offset  $t_0$ , relative to the received input pulse time, when the Gaussian Doublet is to reach its peak value.

#### **Pulse Width**

Defines the width of the pulse by specifying the time interval  $\tau$  between the pulse's zero crossing points.

### **Gated Sinusoid**

$$y(t) = A \cdot \sin\left(2\pi f_c (t-t_0) + \frac{\theta\pi}{180}\right) \quad \text{for } t \in [t_0, T) \quad y(t) = 0 \quad \text{otherwise}$$

#### **Start Time Offset**

Specifies the start time  $t_0$  of the sinusoid burst relative to the received input pulse time.

#### **Tone Duration**

Specifies the time period  $T$  during which the sinusoidal waveform is ON.

#### **Tone Frequency**

Specifies the center frequency  $f_c$  in *Hertz* of the gated sinusoid.

#### **Starting Phase**

Specifies the starting phase  $\theta$  in *degrees* of the sinusoidal waveform.

# Sample Block Diagrams

The following sample application diagrams are included with the VisSim/Comm WLAN module, and are located in the “Wireless” or “UWB” folders in the Comm Examples directory.

<b>Block Diagram</b>	<b>Description</b>
80211a_OFDM.vsm	Example of OFDM modulation w/o the use of a guard interval
80211a_OFDM_GI.vsm	Example of OFDM modulation using a guard interval (1/4 FFT size)
80211a_interleaver.vsm	Illustrates the behavior of the 802.11a/g interleaver block
80211b_CCK.vsm	Illustrates the use of CCK modulation (5.5 Mbps mode)
80211b_DBPSK.vsm	Differential BPSK modulation example including Barker code usage
Bluetooth_GFSK.vsm	GFSK modulation example as used in the Bluetooth specification
Bluetooth_Spectrum.vsm	Bluetooth hopping spectrum simulation
BluetoothHopGen.vsm	Illustrates the hopping sequences associated with Bluetooth
BPSK_OFDM_BER.vsm	Shows OFDM performance using BPSK mapping and no encoding
CCITT_CRC16.vsm	CRC generation example
OFDM_GI_BER.vsm	Shows OFDM performance using BPSK mapping and no encoding, but including a guard interval
Scrambler_errors.vsm	Illustrates error propagation associated with the use of the 802.11a/g scrambler
Short_Hamming.vsm	Illustrates the error correction capability of the (15,10) Hamming code
Uwb_ppm.vsm	Illustrates behavior of the UWB PPM block
Uwb_ppm_spectrum.vsm	Measures the spectrum of a UWB PPM signal
Uwb_ppm_TxRx.vsm	Transmit and receive example for UWB PPM
Uwb_ppm_AWGN.vsm	End-to-end UWB PPM BER calculation in Gaussian noise channel
Uwb_Pulses.vsm	Illustrates various pulse shapes the UWB Pulse block can produce
Uwb_gated_IandD.vsm	Illustrates use of the Gated Integrate & Dump block

# Acronyms and Abbreviations

<b>Term</b>	<b>Definition</b>
16-QAM	16-level quadrature amplitude modulation
64-QAM	64-level quadrature amplitude modulation
AM	amplitude modulation
AWGN	additive white Gaussian noise
BER	bit error rate
BPSK	binary phase shift keying
CCK	complementary code keying
DPSK	differential phase shift keying
DQPSK	differential quadrature phase shift keying
FIR	finite impulse response
FM	frequency modulation
FSK	frequency shift keying
GFSK	Gaussian frequency shift keying
ISI	inter-symbol interference
LSB	least significant bit
MSB	most significant bit
MSK	minimum shift keying
OFDM	orthogonal frequency division multiplexing
PBCC	packet binary convolutional coding
PN	pseudo noise
PPM	pulse position modulation
QAM	quadrature amplitude modulation
QPSK	quadrature phase shift keying
SER	symbol error rate

SNR            signal to noise ratio  
UWB            ultrawideband

# Index

802.11 Descrambler, 21  
802.11 GFSK-2 Modulator, 22  
802.11 GFSK-4 Modulator, 22  
802.11 Hop Generator, 20  
802.11 Overview, 10  
802.11 Scrambler, 21  
802.11a/g Convolutional Encoder, 23  
802.11a/g Depuncture, 23  
802.11a/g Interleaver, 25  
802.11a/g Overview, 11  
802.11a/g Puncture, 24  
802.11a/g Scrambler, 25  
802.11a/g Viterbi Decoder, 26  
802.11b High Rate Hop Generator, 34  
802.11b Overview, 12

## A

Abbreviations, 44  
Acronyms, 44

## B

Barker Sequence, 18  
Blocks  
    802.11, 18  
    802.11a/g, 23  
    802.11b, 34  
    Bluetooth, 15  
    Generic Wireless, 38  
    Ultrawideband, 39  
Bluetooth GFSK Modulator, 17  
Bluetooth Hop Generator, 15  
Bluetooth Overview, 10  
Bluetooth Scrambler, 16

## C

CCK Demodulator, 35  
CCK Modulator, 36  
Communication system key elements, 9  
Computer requirements, 5  
CRC-16 Generator, 19

## D

Doublet, 42

## E

Example wireless simulation, 12

## F

Frequency Hop, 38

## G

Gated Integrate & Dump, 39  
Gated Sinusoid, 42  
Gaussian Doublet, 42  
Gaussian Monocycle, 42  
Gaussian Pulse, 41  
GFSK Modulator, 17  
GFSK-2 Modulator, 22  
GFSK-4 Modulator, 22

## M

Manual conventions, 5  
Monocycle, 42  
Monopulse, 41

## O

OFDM Demodulator, 28  
OFDM Modulator, 28  
OFDM Pilot Extract, 30  
OFDM Pilot Map, 31  
OFDM Vector Demodulator, 32  
OFDM Vector Modulator, 33

## S

Shortened Hamming Decoder, 17  
Shortened Hamming Encoder, 18  
Starting VisSim/Comm WLAN, 5

## T

Tech support, 6

**U**

Ultrawideband Overview, 12  
UWB PPM Modulator, 39  
UWB Pulse, 40

**W**

Wireless blocks, summary of, 9