IEEE STANDARDS ASSOCIATION

IEEE Standard for
Local and metropolitan area networks—

Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

5 September 2011

IEEE Std 802.15.4™-2011
(Revision of
IEEE Std 802.15.4-2006)

IEEE Std 802.15.4™-2011
(Revision of
IEEE Std 802.15.4-2006)

IEEE Standard for
Local and metropolitan area networks—

Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 16 June 2011

IEEE-SA Standards Board

Approved 14 August 2012

American National Standards Institute

Abstract: The protocol and compatible interconnection for data communication devices using low-data-rate, low-power, and low-complexity short-range radio frequency (RF) transmissions in a wireless personal area network (WPAN) were defined in IEEE Std 802.15.4-2006. In this revision, the market applicability of IEEE Std 802.15.4 is extended, the ambiguities in the standard are removed, and the improvements learned from implementations of IEEE Std 802.15.4-2006 are included.

Keywords: ad hoc network, IEEE 802.15.4, low data rate, low power, LR-WPAN, mobility, PAN, personal area network, radio frequency, RF, short range, wireless, wireless personal area network, WPAN

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 978-0-7381-6683-4 STD97126
PDF: ISBN 978-0-7381-6684-1 STDPD97126

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE. Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board

445 Hoes Lane

Piscataway, NJ 08854-4141

USA

Introduction

This introduction is not part of IEEE Std 802.15.4-2011, IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).

This is the second revision of IEEE Std 802.15.4. From the beginning, the goal of 802.15.4 was to produce a standard that enabled very low-cost, low-power communications. The initial standard, IEEE Std 802.15.4-2003, defined two optional PHYs, operating in different frequency bands with a very simple, but effective, MAC.

In 2006, the standard was revised, adding two more PHY options. The MAC was backward-compatible, but it added MAC frames with an increased version number, new security features, and a variety of MAC enhancements, including:

— Support for a shared time base with a data time stamping mechanism
— Support for beacon scheduling
— Synchronization of broadcast messages in beacon-enabled PANs

In 2007, two new PHYs were added as an amendment, one of which supported accurate ranging. As a part of this amendment, MAC capability to support ranging was added.

In 2009, two new PHY amendments were approved, one to provide operation in frequency bands specific in China and the other for operation in frequency bands specific to Japan.

The current revision of the standard was created to roll in the previous three amendments into a single document. However, IEEE Std 802.15.4 had become very popular, and there were three additional amendments, 2 PHY and 1 MAC, in process at that time. It was clear that the original organization of the standard was inadequate for the variety of applications, optional PHYs and optional MAC features to which the 802.15.4 base standard would be applied.

Thus, the major changes in the current revision are not technical but editorial. The organization of the standard was changed so that each PHY now has a separate clause. The MAC clause was split into functional description, interface specification, and security specification. In additon, a great deal of informative text, including the coexisting annex and regulatory annex, were deleted so that the document would focus on only those technical requirements needed for interoperability. The revised organization is the consensus decision of a broad group of 802.15 members, including people who were part of the original standard as well as individuals developing amendments to the standard for new applications.

The PAR for IEEE Std 802.15.4-2011 was first proposed in July 2010 and was approved in September 2010 by NesCom. After a total of 10 drafts, 3 working group ballots ,and 4 sponsor ballots, the final standard was approved in June 2011, less than one year from start to finish.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does

not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this recommended practice may require use of subject matter covered by patent rights. By publication of this recommended practice, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this recommended practice are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time the draft of this standard was sent to sponsor ballot, the IEEE P802.15 Working Group had the following voting members:

Robert F. Heile, *Chair*
Rick Alfvin, *Co-Vice Chair*
Patrick W. Kinney, *Co-Vice Chair*
James P. K. Gilb, *Working Group Technical Editor*
Patrick W. Kinney, *Secretary*

James P. K. Gilb, *Task Group 4i Chair and Technical Editor*

Emad Afifi
Gahng-Seop Ahn
Roberto Aiello
Arthur Astrin
Taehan Bae
Michael Bahr
John Barr
Anuj Batra
Tuncer Baykas
Philip E. Beecher
Ashutosh Bhatia
Ghulam Bhatti
Gary Birk
Mathew Boytim
Peter David Bradley
Nancy Bravin
David Britz
Monique B. Brown
Sverre Brubk
Brian Buchanan
John Buffington
Kiran Bynam
Brent Cain
Edgar H. Callaway
Chris Calvert
Ruben Cardozo
Douglas Castor
Jaesang Cha
Russell Chandler
Kuor-Hsin Chang
Soo-Young Chang
Clint Chaplin
Hind Chebbo
Chang-Soon Choi
Sangsung Choi
Ciaran Connell
David Cypher
Matthew Dahl
David Davenport
Mark Dawkins
Hendricus De Ruijter
Upkar Dhaliwal
Gang Ding
Paul Dixon
Guido Dolmans
Igor Dotlic
Michael Dow
Dietmar Eggert
David Evans
Charles Farlow
John Farserotu
Jeffrey Fischbeck
Mike Fischer
George Flammer
Ryosuke Fujiwara
Noriyasu Fukatsu
Kiyoshi Fukui
John Geiger
Gregory Gillooly
Tim Godfrey
Paul Gorday
Elad Gottlib
Robert Hall
Shinsuke Hara
Hiroshi Harada
Timothy Harrington
Rodney Hemminger
Marco Hernandez
Garth Hillman
Jin-Meng Ho
Wei Hong
Srinath Hosur
David Howard
Jung-Hwan Hwang
Taeho Hwang
Ichirou Ida
Tetsushi Ikegami
Akio Iso
Yeong Min Jang
Adrian Jennings
Wuncheol Jeong
Steven Jillings
Noh-Gyoung Kang
Tae-Gyu Kang
Shuzo Kato
Tatsuya Kato
Jeritt Kent
Prithpal Khakuria
Dae Ho Kim
Dong-Sun Kim
Dukhyun Kim
Jaehwan Kim
Jeffrey King
Ryuji Kohno
Fumihide Kojima
Bruce Kraemer
Raymond Krasinski
Masahiro Kuroda
John Lampe
Zhou Lan
Khanh Le
Cheolhyo Lee
Hyungsoo Lee
Myung Lee
Daniel Lewis
Huan-Bang Li
Liang Li
Sang-Kyu Lim
Jeremy Link
Mike Lynch
Robert Mason
Tomokuni Matsumura
Jeff McCullough
Michael McGillan
Michael D. McInnis
Michael McLaughlin
Charles Millet
Siamak Mirnezami
Rishi Mohindra
Emmanuel Monnerie
Rajendra Moorti
Robert Moskowitz
Hamilton Moy
Peter Murray
Theodore Myers
Chiu Ngo
Paul Nikolich
Hirohito Nishiyama
David Olson
Okundu Omeni
Ryoji Ono
Laurent Ouvry
James Pace
Hyung-Il Park
Jahng Park
Seung-Hoon Park
Taejoon Park
Ranjeet Patro
Al Petrick
Dalibor Pokrajac
Daniel Popa
Stephen Pope
Clinton C. Powell
Richard Powell
Chang-Woo Pyo
Mohammad Rahman
Sridhar Rajagopal

Jayaram Ramasastry
Marc Reed
Ivan Reede
Richard Roberts
Craig Rodine
June Chul Roh
Benjamin Rolfe
Seung-Moon Ryu
Didier Sagan
Kentaro Sakamoto
Will San Filippo
H. Sanderford
Kamran Sayrafian
Timothy Schmidl
Michael Schmidt
Jean Schwoerer
Cristina Seibert
Neal Seidl
Kunal Shah
Steve Shearer
Stephen Shellhammer
Shusaku Shimada
Chang Sub Shin
Cheol Ho Shin
Michael Sim
Jonathan Simon
Jaeseung Son
Paul Stadnik
René Struik
Chin-Sean Sum
Hui-Hsia Sung
Gu Sungi
Kenichi Takizawa
Hirokazu Tanaka
Larry Taylor
Mark Thompson
James Tomcik
Ichihiko Toyoda
David Tracey
Khanh Tran
Jerry Upton
Jana van Greunen
Hartman van Wyk
Michel Veillette
Billy Verso
Bhupender Virk
Joachim Walewski
Junyi Wang
Quan Wang
Xiang Wang
Andy Ward
Scott Weikel
Nicholas West
Mark Wilbur
Ludwig Winkel
Eun Tae Won
Alan Chi Wai Wong
Tao Xing
Wen-Bin Yang
Yang Yang
Kazuyuki Yasukawa
Kamya Yazdandoost
Kaoru Yokoo
Mu Zhao
Bin Zhen

Major contributions were received from the following individuals:

Philip E. Beecher
Vern Brethour
Monique B. Brown
Edgar H. Callaway
Kuor-Hsin Chang
Clint Chaplin
James P. K. Gilb
Patrick W. Kinney
Michael D. McInnis
Clinton C. Powell
Benjamin Rolfe
Timothy Schmidl
René Struik
Billy Verso

The following members of the balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Jon Adams
Emad Afifi
Roberto Aiello
Rick Alfvin
Nobumitsu Amachi
Mark Anderson
Tuncer Baykas
Philip E. Beecher
H. Stephen Berger
Maciej Borowka
Nancy Bravin
Vern Brethour
John Buffington
William Byrd
Edgar H. Callaway
Radhakrishna Canchi
Ruben Cardozo
Juan Carreon
Dave Cavalcanti
Kuor-Hsin Chang
Clint Chaplin
Keith Chow
Charles Cook
Todor Cooklev
Joseph Decuir
Patrick Diamond
Russell Dietz
Thomas Dineen
Sourav Dutta
Richard Eckard
Devon Gayle
John Geiger
James P. K. Gilb
Gregory Gillooly
Randall Groves
Jose Gutierrez
C. Guy
Rainer Hach
Robert F. Heile
Marco Hernandez
Oliver Hoffmann
Wei Hong
Heqing Huang
David Hunter
Noriyuki Ikeuchi
Akio Iso
Atsushi Ito
Raj Jain
Oyvind Janbu
Tal Kaitz
Shinkyo Kaku
Stuart J. Kerry
Brian Kiernan
Yongbum Kim
Youngsoo Kim
Jeffrey King
Bruce Kraemer
Geoff Ladwig
Paul Lambert
Jeremy Landt
Zhongding Lei
Jan-Ray Liao
Arthur Light
Lu Liru
HaiTao Liu
William Lumpkins
Greg Luri
Mike Lynch
Elvis Maculuba
Wayne W. Manges
Michael D. McInnis
Gary Michel
Apurva Mody
Emmanuel Monnerie
Jose Morales
Peter Murray
Michael S. Newman
Nick S. A. Nikjoo
Paul Nikolich
John Notor
Satoshi Obara
Chris Osterloh
Clinton C. Powell
Venkatesha Prasad
Mohammad Rahman
Jayaram Ramasastry
Robert Robinson
Benjamin Rolfe
Randall Safier
Kazuyuki Sakoda
Naotaka Sato
Bartien Sayogo
Timothy Schmidl
Cristina Seibert
Jie Shen
Suresh Shrimavle
Gil Shultz
Kapil Sood
Robert Soranno
Thomas Starai
René Struik
Walter Struppler
Mark Sturza
Chin-Sean Sum
Larry Taylor
Dmitri Varsanofiev
Prabodh Varshney
Bhupender Virk
George Vlantis
Xiang Wang
Scott Weikel
Stephen Weinstein
Andreas Wolf
Tao Xing
Yang Yang
Tan Pek Yew
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 16 June 2011, it had the following membership:

Richard H. Hulett, *Chair*
John Kulick, *Vice Chair*
Robert Grow, *Past Chair*
Judith Gorman, *Secretary*

Masayuki Ariyoshi
William Bartley
Ted Burse
Clint Chaplin
Wael Diab
Jean-Philippe Faure
Alex Gelman
Paul Houzé
Jim Hughes
Joseph L. Koepfinger*
David Law
Thomas Lee
Hung Ling
Oleg Logvinov
Ted Olsen
Gary Robinson
Jon Rosdahl
Sam Sciacca
Mike Seavey
Curtis Siller
Phil Winston
Howard Wolfman
Don Wright

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan H. Cookson, *NIST Representative*

Catherine Berger
IEEE Standards Program Manager, Document Development

Patricia A. Gerdon
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Standard for
Local and metropolitan area networks—

Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)

1. Overview

1.1 General

Wireless personal area networks (WPANs) are used to convey information over relatively short distances. Unlike wireless local area networks (WLANs), connections effected via WPANs involve little or no infrastructure. This feature allows small, power-efficient, inexpensive solutions to be implemented for a wide range of devices.

1.2 Scope

This standard defines the physical layer (PHY) and medium access control (MAC) sublayer specifications for low-data-rate wireless connectivity with fixed, portable, and moving devices with no battery or very limited battery consumption requirements typically operating in the personal operating space (POS) of 10 m.

Physical layers (PHYs) are defined for

- — Devices operating in the license-free 868–868.6 MHz, 902–928 MHz, and 2400–2483.5 MHz bands
- — Devices with precision ranging, extended range, and enhanced robustness and mobility
- — Devices operating according the Chinese regulations, Radio Management of P. R. of China doc. #6326360786867187500 or current document, for one or more of the 314–316 MHz, 430–434 MHz, and 779–787 MHz frequency bands
- — Devices operating in the 950–956 MHz allocation in Japan and coexisting with passive tag systems in the band

1.3 Purpose

The standard provides for ultra low complexity, ultra low cost, ultra low power consumption, and low data rate wireless connectivity among inexpensive devices. The raw data rate is high enough (250 kb/s) to satisfy a set of applications but is also scaleable down to the needs of sensor and automation needs (20 kb/s or below) for wireless communications.

In addition, one of the alternate PHYs provides precision ranging capability that is accurate to one meter.

Multiple PHYs are defined to support a variety of frequency bands including

— 868–868.6 MHz
— 902–928 MHz
— 2400–2483.5 MHz
— 314–316 MHz, 430–434 MHz, and 779–787 MHz band for LR-WPAN systems in China
— 950–956 MHz in Japan

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

FIPS Pub 197, Advanced Encryption Standard (AES).[1]

NITS/CWPAN Part 15.4, Chinese standard for the Wireless Medium Access Control (MAC) and the Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks, GB/T 15629.15-2010.[2]

[1]FIPS publications are available from the National Technical Information Service (NTIS), U.S. Dept. of Commerce, 5285 Port Royal Rd., Springfield, VA 22161, USA (http://www.ntis.org/).

[2]CWPAN publications are available from either the China Electronics Standardization Institute (CESI), No.1 Andingmen East Street, Dongcheng District, Beijng, China (http://www.cssn.net.cn) or the China National Institute of Standardization (CNIS), No. 4 Zhichun Road, Haidian District, Beijng, China.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. T*he IEEE Standards Dictionary: Glossary of Terms & Definitions* should be consulted for terms not defined in this clause.[3]

association: The service used to establish membership for a device in a network.

authentication tag: Information that allows the verification of authenticity of a message.

beacon-enabled personal area network (PAN): A PAN in which all coordinators emit regular beacons.

chirp: Linear frequency sweep (frequency may either increase or decrease).

contention access period: The period of time immediately following a beacon frame during which devices wishing to transmit will compete for channel access using a slotted carrier sense multiple access with collision avoidance mechanism.

coordinator: A device in an low rate wireless personal area network (LR WPAN) that provides synchronization services to other devices in the LR WPAN.

data authentication: The process whereby an entity receiving a message corroborates evidence about the true source of the information in the message and, thereby, evidence that the message has not been modified in transit.

data authenticity: Assurance about the source of information.

device: Any entity containing an implementation of the IEEE 802.15.4 medium access control and physical interface to the wireless medium. A device may be a reduced-function device or a full-function device.

encryption: The transformation of a message into a new representation so that privileged information is required to recover the original representation.

frame: The format of aggregated bits from a medium access control sublayer entity that are transmitted together in time.

full-function device: A device capable of operating as a coordinator.

group key: A key that is known only to a set of devices.

key: Privileged information that may be used, for example, to protect information from disclosure to, and/or undetectable modification by, parties that do not have access to this privileged information.

key establishment: A process whereby two or more parties establish a key.

keying material: The combination of a key and associated security information (e.g., a nonce value).

key management: The collection of processes for the establishment and maintenance of keying relationships over a system's lifetime.

[3]*The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

link key: A secret key that is shared between precisely two devices.

mobile device: A device whose location in the network may change during use.

nonbeacon-enabled personal area network (PAN): A PAN in which coordinators do not emit regular beacons.

nonce: A nonrepeating value, such as an increasing counter, a sufficiently long random string, or a time stamp.

orphaned device: A device that has lost contact with its associated coordinator.

packet: The formatted, aggregated bits that are transmitted together in time across the physical medium.

payload data: The contents of a data message that is being transmitted.

personal area network (PAN) coordinator: A coordinator that is the principal controller of a PAN. An IEEE 802.15.4 network has exactly one PAN coordinator.

plain text: A string of unscrambled information.

ranging-capable device: A device containing an implementation capable of supporting ranging.

reduced-function device (RFD): A device that is not capable of operating as a coordinator.

symmetric key: A secret key that is shared between two or more parties that may be used for encryption/decryption or integrity protection/integrity verification, depending on its intended use.

transaction: The exchange of related, consecutive frames between two peer medium access control (MAC) entities, required for a successful transmission of a MAC command or data frame.

3.2 Acronyms and abbreviations

ADC	analog-to-digital converter
AES	advanced encryption standard
AGC	automatic gain control
AR	acknowledgment request
ASK	amplitude shift keying
AWGN	additive white Gaussian noise
BER	bit error rate
BLE	battery life extension
BPM	burst position modulation
BPSK	binary phase-shift keying
BSN	beacon sequence number
CAP	contention access period
CBC-MAC	cipher block chaining message authentication code
CCA	clear channel assessment
CCM	counter mode encryption and cipher block chaining message authentication code
CCM*	extension of counter mode encryption and cipher block chaining message authentication code
CFP	contention-free period

CRC	cyclic redundancy check
CoU	chirp on ultra-wide band
CS	continuous spectrum
CSK	chirp-shift keying
CSMA-CA	carrier sense multiple access with collision avoidance
CSS	chirp spread spectrum
CTR	counter mode
CWPAN	Chinese wireless personal area network
DAA	detect and avoid
DEMUX	de-multiplexer
DPS	dynamic preamble selection
DQCSK	differential quadrature chirp-shift keying
DQPSK	differential quadrature phase-shift keying
DSN	data sequence number
DSSS	direct sequence spread spectrum
ED	energy detection
EIRP	effective isotropic radiated power
ERP	effective radiated power
EVM	error-vector magnitude
FCS	frame check sequence
FEC	forward error correction
FFD	full-function device
FH	frequency hopping
FHSS	frequency hopping spread spectrum
FoM	figure of merit
GDOP	geometric dilution of precision
GFSK	Gaussian frequency-shift keying
GTS	guaranteed time slot
IFS	interframe space or spacing
ISM	industrial, scientific, and medical
LBT	listen before talk
LCP	linear combination of pulses
LFSR	linear feedback shift register
LIFS	long interframe spacing
LOS	line-of-sight
LQI	link quality indication
LPDU	logical link control protocol data unit
LR-WPAN	low-rate wireless personal area network
LSB	least significant bit
MAC	medium access control
MCPS	MAC common part sublayer
MCPS-SAP	MAC common part sublayer service access point
MFR	MAC footer
MHR	MAC header
MIC	message integrity code
MLME	MAC sublayer management entity

MLME-SAP	MAC sublayer management entity service access point
MSB	most significant bit
MPDU	MAC protocol data unit
MPSK	m-ary phase shift keying
MSDU	MAC service data unit
NLOS	non-line-of-sight
OCDM	orthogonal code division multiplexing
O-QPSK	offset quadrature phase-shift keying
PAN	personal area network
PC	personal computer
PD	PHY data
PD-SAP	PHY data service access point
PER	packet error rate
PHR	PHY header
PHY	physical layer
PIB	personal area network information base
PICS	protocol implementation conformance statement
PLL	phase-locked loop
PLME	physical layer management entity
PLME-SAP	physical layer management entity service access point
PN	pseudo-random noise
PPDU	PHY protocol data unit
PPM	pulse position modulation
PRBS	pseudo-random binary sequence
PRF	pulse repetition frequency
PSD	power spectral density
PSDU	PHY service data unit
PSSS	parallel sequence spread spectrum
RDEV	ranging-capable device
RF	radio frequency
RFD	reduced-function device
RFRAME	ranging frame
RMARKER	ranging marker
RSSI	receive signal strength indicator
RX	receive or receiver
SDS-TWR	symmetric double-sided two-way ranging
SFD	start-of-frame delimiter
SHR	synchronization header
SIFS	short interframe spacing
SNR	signal-to-noise ratio
TRX	transceiver
TX	transmit or transmitter
UWB	ultra-wide band
WPAN	wireless personal area network

4. General description

4.1 General

An LR-WPAN is a simple, low-cost communication network that allows wireless connectivity in applications with limited power and relaxed throughput requirements. The main objectives of an LR-WPAN are ease of installation, reliable data transfer, extremely low cost, and a reasonable battery life, while maintaining a simple and flexible protocol.

Some of the capabilities provided by this standard are as follows:

— Star or peer-to-peer operation
— Unique 64-bit extended address or allocated 16-bit short address
— Optional allocation of guaranteed time slots (GTSs)
— Carrier sense multiple access with collision avoidance (CSMA-CA) or ALOHA channel access
— Fully acknowledged protocol for transfer reliability
— Low power consumption
— Energy detection (ED)
— Link quality indication (LQI)

This standard defines multiple PHYs operating in a variety of frequency bands, as described in 8.1.1.

Two different device types can participate in an IEEE 802.15.4 network: a full-function device (FFD) and a reduced-function device (RFD). An FFD is a device that is capable of serving as a personal area network (PAN) coordinator or a coordinator. An RFD is a device that is not capable of serving as either a PAN coordinator or a coordinator. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; it does not have the need to send large amounts of data and only associates with a single FFD at a time. Consequently, the RFD can be implemented using minimal resources and memory capacity.

4.2 Components of the IEEE 802.15.4 WPAN

A system conforming to this standard consists of several components. The most basic is the device. Two or more devices communicating on the same physical channel constitute a WPAN. However, this WPAN includes at least one FFD, which operates as the PAN coordinator.

A well-defined coverage area does not exist for wireless media because propagation characteristics are dynamic and uncertain. Small changes in position or direction often result in drastic differences in the signal strength or quality of the communication link. These effects occur whether a device is stationary or mobile, as moving objects affect station-to-station propagation.

4.3 Network topologies

Depending on the application requirements, an IEEE 802.15.4 LR-WPAN operates in either of two topologies: the star topology or the peer-to-peer topology. Both are shown in Figure 1. In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator. A device typically has some associated application and is either the initiation point or the termination point for network communications. A PAN coordinator can also have a specific application, but it can be used to initiate, terminate, or route communication around the network. The PAN coordinator is the primary controller of the PAN. All devices operating on a network of either topology have unique addresses,

referred to as extended addresses. A device will use either the extended address for direct communication within the PAN or the short address that was allocated by the PAN coordinator when the device associated. The PAN coordinator will often be mains powered, while the devices will most likely be battery powered. Applications that benefit from a star topology include home automation, personal computer (PC) peripherals, games, and personal health care.

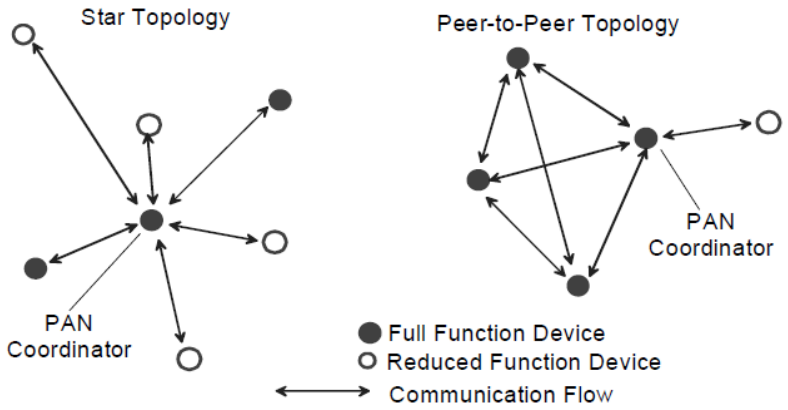


Figure 1—Star and peer-to-peer topology examples

The peer-to-peer topology also has a PAN coordinator; however, it differs from the star topology in that any device is able to communicate with any other device as long as they are in range of one another. Peer-to-peer topology allows more complex network formations to be implemented, such as mesh networking topology. Applications such as industrial control and monitoring, wireless sensor networks, asset and inventory tracking, intelligent agriculture, and security would benefit from such a network topology. A peer-to-peer network allows multiple hops to route messages from any device to any other device on the network. Such functions can be added at the higher layer, but they are not part of this standard.

Each independent PAN selects a unique identifier. This PAN identifier allows communication between devices within a network using short addresses and enables transmissions between devices across independent networks. The mechanism by which identifiers are chosen is outside the scope of this standard.

The network formation is performed by the higher layer, which is not part of this standard. However, 4.3.1 and 4.3.2 provide a brief overview on how each supported topology can be formed.

4.3.1 Star network formation

The basic structure of a star network is illustrated in Figure 1. After an FFD is activated, it can establish its own network and become the PAN coordinator. All star networks operate independently from all other star networks currently in operation. This is achieved by choosing a PAN identifier that is not currently used by any other network within the radio communications range. Once the PAN identifier is chosen, the PAN coordinator allows other devices, potentially both FFDs and RFDs, to join its network. The higher layer can use the procedures described in 5.1.2 and 5.1.3 to form a star network.

4.3.2 Peer-to-peer network formation

In a peer-to-peer topology, each device is capable of communicating with any other device within its radio communications range. One device is nominated as the PAN coordinator, for instance, by virtue of being the first device to communicate on the channel. Further network structures are constructed out of the peer-to-peer topology, and it is possible to impose topological restrictions on the formation of the network.

An example of the use of the peer-to-peer communications topology is the cluster tree. The cluster tree network is a special case of a peer-to-peer network in which most devices are FFDs. An RFD connects to a cluster tree network as a leaf device at the end of a branch because RFDs do not allow other devices to associate. Any FFD is able to act as a coordinator and provide synchronization services to other devices or other coordinators. Only one of these coordinators is the overall PAN coordinator, potentially because it has greater computational resources than any other device in the PAN. The PAN coordinator forms the first cluster by choosing an unused PAN identifier and broadcasting beacon frames to neighboring devices. A contention resolution mechanism is required if two or more FFDs simultaneously attempt to establish themselves as PAN coordinators; however, such a mechanism is outside the scope of this standard. A candidate device receiving a beacon frame is able to request to join the network at the PAN coordinator. If the PAN coordinator permits the device to join, it adds the new device as a child device in its neighbor list. Then the newly joined device adds the PAN coordinator as its parent in its neighbor list and begins transmitting periodic beacons; other candidate devices are able to then join the network at that device. If the original candidate device is not able to join the network at the PAN coordinator, it will search for another parent device. The detailed procedures describing how a PAN is started and how devices join a PAN are found in 5.1.2 and 5.1.3.

The simplest form of a cluster tree network is a single cluster network, but larger networks are possible by forming a mesh of multiple neighboring clusters. Once predetermined application or network requirements are met, the first PAN coordinator instructs a device to become the PAN coordinator of a new cluster adjacent to the first one. Other devices gradually connect and form a multicluster network structure, such as the one seen in Figure 2. The lines in Figure 2 represent the parent-child relationships of the devices and not the communication flow. The advantage of a multicluster structure is increased coverage area, while the disadvantage is an increase in message latency.

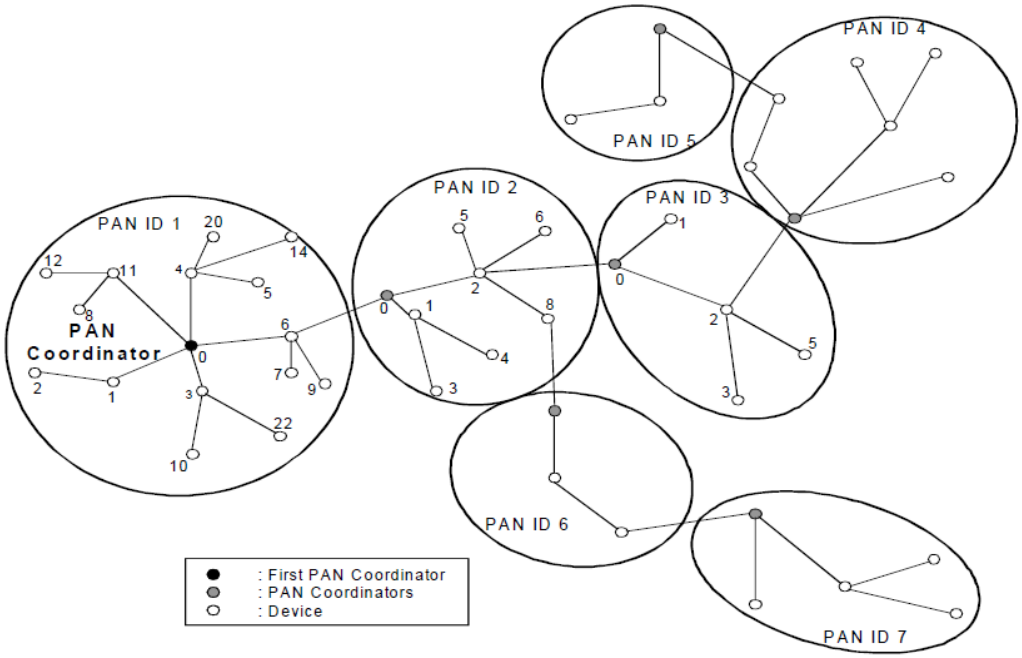


Figure 2—Cluster tree network

4.4 Architecture

The IEEE 802.15.4 architecture is defined in terms of a number of blocks in order to simplify the standard. These blocks are called layers. Each layer is responsible for one part of the standard and offers services to the higher layers.

The interfaces between the layers serve to define the logical links that are described in this standard.

An LR-WPAN device comprises at least one PHY, which contains the radio frequency (RF) transceiver along with its low-level control mechanism, and a MAC sublayer that provides access to the physical channel for all types of transfer. Figure 3 shows these blocks in a graphical representation, which are described in more detail in 4.4.1 and 4.4.2.

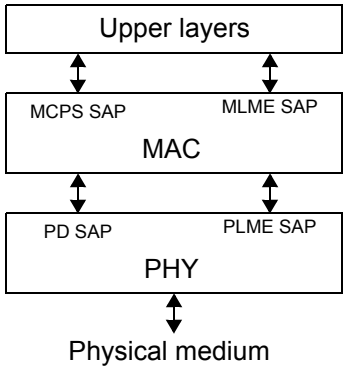


Figure 3—LR-WPAN device architecture

The upper layers, shown in Figure 3, consist of a network layer, which provides network configuration, manipulation, and message routing, and an application layer, which provides the intended function of the device. The definition of these upper layers is outside the scope of this standard.

4.4.1 Physical layer (PHY)

The PHY provides two services: the PHY data service and the PHY management service. The PHY data service enables the transmission and reception of PHY protocol data units (PPDUs) across the physical radio channel. The general PHY requirements are described in Clause 8.

The features of the PHY are activation and deactivation of the radio transceiver, ED, LQI, channel selection, clear channel assessment (CCA), and transmitting as well as receiving packets across the physical medium. The UWB PHY also has the feature of precision ranging.

A discussion of the coexistence of the various IEEE 802.15.4 PHYs with other wireless systems is given in "Coexistence analysis of IEEE Std 802.15.4 with other IEEE standards and proposed standards" [B4].[4]

4.4.2 MAC sublayer

The MAC sublayer provides two services: the MAC data service and the MAC management service interfacing to the MAC sublayer management entity (MLME) service access point (SAP) (known as MLME-SAP). The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service.

The features of the MAC sublayer are beacon management, channel access, GTS management, frame validation, acknowledged frame delivery, association, and disassociation. In addition, the MAC sublayer provides hooks for implementing application-appropriate security mechanisms.

Clause 5 and Clause 6 contains the specifications for the MAC sublayer.

[4]The numbers in brackets correspond to the numbers of the bibliography in Annex A.

4.5 Functional overview

A brief overview of the general functions of a LR-WPAN is given in this subclause.

4.5.1 Superframe structure

This standard allows the optional use of a superframe structure. The format of the superframe is defined by the coordinator. The superframe is bounded by network beacons sent by the coordinator, as illustrated in Figure 4a), and is divided into 16 slots of equal duration. Optionally, the superframe can have an active and an inactive portion, as illustrated in Figure 4b). During the inactive portion, the coordinator is able to enter a low-power mode. The beacon frame transmission starts at the beginning of the first slot of each superframe. If a coordinator does not wish to use a superframe structure, it will turn off the beacon transmissions. The beacons are used to synchronize the attached devices, to identify the PAN, and to describe the structure of the superframes.

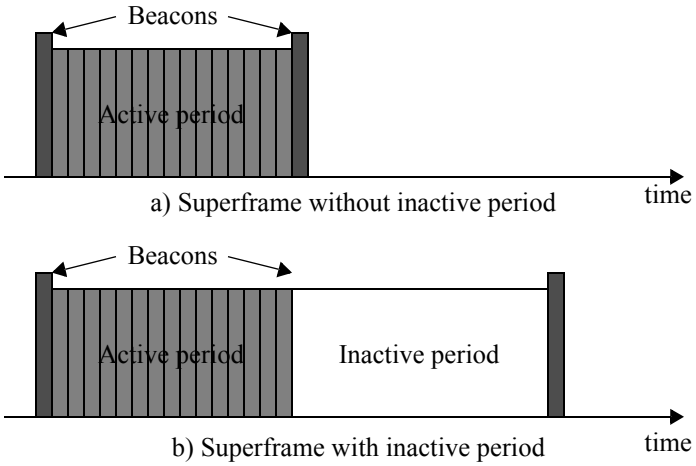


Figure 4—Superframe structure

Any device wishing to communicate during the contention access period (CAP) between two beacons competes with other devices using a slotted CSMA-CA or ALOHA mechanism, as appropriate. For low-latency applications or applications requiring specific data bandwidth, the PAN coordinator dedicates portions of the active superframe to that application. These portions are called guaranteed time slots (GTSs). The GTSs form the contention-free period (CFP), which always appears at the end of the active superframe starting at a slot boundary immediately following the CAP, as shown in Figure 5. The PAN coordinator allocates up to seven of these GTSs, and a GTS is allowed to occupy more than one slot period. However, a sufficient portion of the CAP remains for contention-based access of other networked devices or new devices wishing to join the network. All contention-based transactions are completed before the CFP begins. Also each device transmitting in a GTS ensures that its transaction is complete before the time of the next GTS or the end of the CFP. More information on the superframe structure can be found in 5.1.1.1.

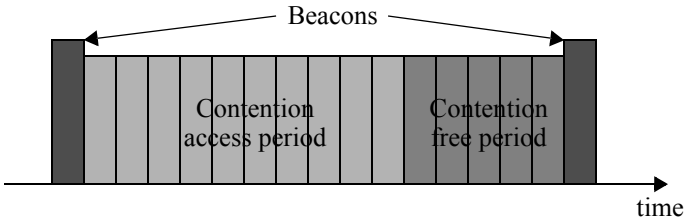


Figure 5—Structure of the active periods with GTSs

4.5.2 Data transfer model

Three types of data transfer transactions exist. The first one is the data transfer to a coordinator in which a device transmits the data. The second transaction is the data transfer from a coordinator in which the device receives the data. The third transaction is the data transfer between two peer devices. In star topology, only two of these transactions are used because data is exchanged only between the coordinator and a device. In a peer-to-peer topology, data is exchanged between any two devices on the network; consequently all three transactions are used in this topology.

The mechanisms for each transfer type depend on whether the network supports the transmission of periodic beacons. A beacon-enabled PAN is used in networks that either require synchronization or support for low-latency devices, such as PC peripherals. If the network does not need synchronization or support for low-latency devices, it can elect not to use the beacon for normal transfers. However, the beacon is still required for network discovery. The structure of the frames used for the data transfer is specified in 5.2.

4.5.2.1 Data transfer to a coordinator

When a device wishes to transfer data to a coordinator in a beacon-enabled PAN, it first listens for the network beacon. When the beacon is found, the device synchronizes to the superframe structure. At the appropriate time, the device transmits its data frame to the coordinator. The coordinator will acknowledge the successful reception of the data by transmitting an acknowledgment frame, if requested.

When a device wishes to transfer data in a nonbeacon-enabled PAN, it simply transmits its data frame to the coordinator. The coordinator acknowledges the successful reception of the data by transmitting an optional acknowledgment frame, completing the transaction.

4.5.2.2 Data transfer from a coordinator

When the coordinator wishes to transfer data to a device in a beacon-enabled PAN, it indicates in the network beacon that the data message is pending. The device periodically listens to the network beacon and, if a message is pending, transmits a MAC command requesting the data. The coordinator acknowledges the successful reception of the data request by transmitting an acknowledgment frame. The pending data frame is then sent by the coordinator. The device acknowledges the successful reception of the data by transmitting an acknowledgment frame, if requested. The transaction is now complete. Upon successful completion of the data transaction, the message is removed from the list of pending messages in the beacon.

When a coordinator wishes to transfer data to a device in a nonbeacon-enabled PAN, it stores the data for the appropriate device to make contact and request the data. A device requests data by transmitting a MAC command requesting the data to its coordinator. The coordinator acknowledges the successful reception of the data request by transmitting an acknowledgment frame. If a data frame is pending, the coordinator transmits the data frame. If a data frame is not pending, the coordinator indicates this fact either in the acknowledgment frame following the data request or in a data frame with a zero-length payload, as defined in 5.1.6.3. If requested, the device acknowledges the successful reception of the data frame by transmitting an acknowledgment frame.

4.5.2.3 Peer-to-peer data transfers

In a peer-to-peer PAN, every device communicates directly with every other device in its radio communications range. In order to do this effectively, the devices wishing to communicate will need to either receive constantly or synchronize with each other. In the former case, the device can simply transmit its data. In the latter case, other measures need to be taken in order to achieve synchronization. Such measures are beyond the scope of this standard.

4.5.3 Frame structure

The frame structures have been designed to keep the complexity to a minimum while at the same time making them sufficiently robust for transmission on a noisy channel. Each successive protocol layer adds to the structure with layer-specific headers and footers. This standard defines four MAC frame structures:

— A beacon frame, used by a coordinator to transmit beacons
— A data frame, used for all transfers of data
— An acknowledgment frame, used for confirming successful frame reception
— A MAC command frame, used for handling all MAC peer entity control transfers

The MAC frames are passed to the PHY as the PHY service data unit (PSDU), which becomes the PHY payload. The PPDU is illustrated in Figure 6.

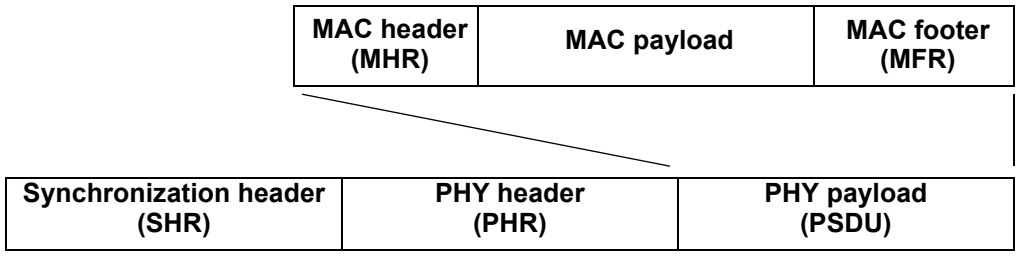


Figure 6—Schematic view of the PPDU

The format of the SHR and PHR is defined for each of the PHYs in their respective clause.

The MHR and MFR are defined in 5.2.1.

4.5.4 Improving probability of successful delivery

The IEEE 802.15.4 LR-WPAN employs various mechanisms to improve the probability of successful data transmission. These mechanisms are discussed in 4.5.4.1 through 4.5.4.4.

4.5.4.1 CSMA-CA mechanism

The IEEE 802.15.4 LR-WPAN uses two types of channel access mechanism, depending on the network configuration. Nonbeacon-enabled PANs use an unslotted CSMA-CA channel access mechanism, as described in 5.1.1.4. Each time a device wishes to transmit data frames or MAC commands, it waits for a random period. If the channel is found to be idle, following the random backoff, the device transmits its data. If the channel is found to be busy following the random backoff, the device waits for another random period before trying to access the channel again. Acknowledgment frames are sent without using a CSMA-CA mechanism.

Beacon-enabled PANs use a slotted CSMA-CA channel access mechanism, where the backoff periods are aligned with the start of the beacon transmission. The backoff periods of all devices within one PAN are aligned to the PAN coordinator. Each time a device wishes to transmit data frames during the CAP, it locates the boundary of the next backoff period and then waits for a random number of backoff periods. If the channel is busy, following this random backoff, the device waits for another random number of backoff periods before trying to access the channel again. If the channel is idle, the device begins transmitting on the next available backoff period boundary. Acknowledgment and beacon frames are sent without using a CSMA-CA mechanism.

4.5.4.2 ALOHA mechanism

In the ALOHA protocol, a device transmits without sensing the medium or waiting for a specific time slot. The ALOHA mechanism is appropriate for lightly loaded networks since the probability of collision is reasonably small if the probability of clear channel is sufficiently large.

4.5.4.3 Frame acknowledgment

A successful reception and validation of a data or MAC command frame is optionally confirmed with an acknowledgment, as described in 5.1.6.4. If the receiving device is unable to handle the received data frame for any reason, the message is not acknowledged.

If the originator does not receive an acknowledgment after some period, it assumes that the transmission was unsuccessful and retries the frame transmission. If an acknowledgment is still not received after several retries, the originator can choose either to terminate the transaction or to try again. When the acknowledgment is not required, the originator assumes the transmission was successful.

4.5.4.4 Data verification

A cyclic redundancy check (CRC) is used to detect errors in every PSDU, as defined in 5.2.1.9.

4.5.5 Power consumption considerations

In many applications that use this standard, devices will be battery powered, and battery replacement or recharging in relatively short intervals is impractical. Therefore, the power consumption is of significant concern. This standard was developed with limited power supply availability in mind. However, the physical implementation of this standard will require additional power management considerations that are beyond the scope of this standard.

The protocol has been developed to favor battery-powered devices. However, in certain applications, some of these devices could potentially be mains powered. Battery-powered devices will require duty-cycling to reduce power consumption. These devices will spend most of their operational life in a sleep state; however, each device periodically listens to the RF channel in order to determine whether a message is pending. This mechanism allows the application designer to decide on the balance between battery consumption and message latency. Higher powered devices have the option of listening to the RF channel continuously.

In addition to the power saving features of the LR-WPAN system, the UWB PHY also provides a hybrid modulation that enables very simple, noncoherent receiver architectures to further minimize power consumption and implementation complexity.

4.5.6 Security

From a security perspective, wireless ad hoc networks are no different from any other wireless network. They are vulnerable to passive eavesdropping attacks and active tampering because physical access to the wire is not required to participate in communications. The very nature of ad hoc networks and their cost objectives impose additional security constraints, which perhaps make these networks the most difficult environments to secure. Devices are low-cost and have limited capabilities in terms of computing power, available storage, and power drain; and it cannot always be assumed they have a trusted computing base nor a high-quality random number generator aboard. Communications cannot rely on the online availability of a fixed infrastructure and might involve short-term relationships between devices that never have communicated before. These constraints severely limit the choice of cryptographic algorithms and protocols and influence the design of the security architecture because the establishment and maintenance of trust relationships between devices need to be addressed with care. In addition, battery lifetime and cost constraints put severe limits on the security overhead these networks can tolerate, something that is of far

less concern with higher bandwidth networks. Most of these security architectural elements can be implemented at higher layers and, therefore, are outside the scope of this standard.

The cryptographic mechanism in this standard is based on symmetric-key cryptography and uses keys that are provided by higher layer processes. The establishment and maintenance of these keys are outside the scope of this standard. The mechanism assumes a secure implementation of cryptographic operations and secure and authentic storage of keying material.

The cryptographic mechanism provides particular combinations of the following security services:

— *Data confidentiality:* Assurance that transmitted information is only disclosed to parties for which it is intended.

— *Data authenticity:* Assurance of the source of transmitted information (and, hereby, that information was not modified in transit).

— *Replay protection:* Assurance that duplicate information is detected.

The actual frame protection provided can be adapted on a frame-by-frame basis and allows for varying levels of data authenticity (to minimize security overhead in transmitted frames where required) and for optional data confidentiality. When nontrivial protection is required, replay protection is always provided.

Cryptographic frame protection uses either a key shared between two peer devices (link key) or a key shared among a group of devices (group key), thus allowing some flexibility and application-specific tradeoffs between key storage and key maintenance costs versus the cryptographic protection provided. If a group key is used for peer-to-peer communication, protection is provided only against outsider devices and not against potential malicious devices in the key-sharing group.

For more detailed information on the cryptographic security mechanisms used for protected MAC frames following this standard, refer to Clause 7.

4.6 Concept of primitives

The services of a layer are the capabilities it offers to the user in the next higher layer or sublayer by building its functions on the services of the next lower layer. This concept is illustrated in Figure 7, showing the service hierarchy and the relationship of the two correspondent users and their associated layer (or sublayer) peer protocol entities.

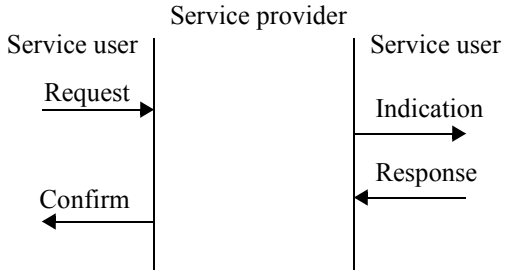


Figure 7—Service primitives

The services are specified by describing the information flow between the N-user and the N-layer. This information flow is modeled by discrete, instantaneous events, which characterize the provision of a service. Each event consists of passing a service primitive from one layer to the other through a layer SAP associated with an N-user. Service primitives convey the required information by providing a particular service. These

service primitives are an abstraction because they specify only the provided service rather than the means by which it is provided. This definition is independent of any other interface implementation.

A service is specified by describing the service primitives and parameters that characterize it.

A primitive can be one of four generic types:

— *Request:* The request primitive is used to request that a service is initiated.
— *Indication:* The indication primitive is used to indicate to the user an internal event.
— *Response:* The response primitive is used to complete a procedure previously invoked by an indication primitive.
— *Confirm:* The confirm primitive is used to convey the results of one or more associated previous service requests.

5. MAC protocol

5.1 MAC functional description

The MAC sublayer handles all access to the physical radio channel and is responsible for the following tasks:

— Generating network beacons if the device is a coordinator
— Synchronizing to network beacons
— Supporting PAN association and disassociation
— Supporting device security
— Employing the CSMA-CA mechanism for channel access
— Handling and maintaining the GTS mechanism
— Providing a reliable link between two peer MAC entities

There are two device types: a full-function device (FFD) and a reduced-function device (RFD). The FFD may operate in three modes serving as a personal area network (PAN) coordinator, a coordinator, or a device. An RFD shall only operate as a device.

Throughout this subclause, the receipt of a frame is defined as the successful receipt of the frame by the PHY and the successful verification of the frame check sequence (FCS) by the MAC sublayer, as described in 5.2.1.9.

Constants and PAN information base (PIB) attributes that are specified and maintained by the MAC sublayer or PHY layer are written in the text in italics. Constants have a general prefix of “a”, e.g., *aBaseSlotDuration*, and the MAC constants are listed in Table 51, while the PHY constants are listed in Table 70. MAC PIB attributes have a general prefix of “mac”, e.g., *macAckWaitDuration*, and are listed in Table 52, while the security attributes are listed in Table 60. PHY PIB attributes have a general prefix of “phy”, e.g., *phyCurrentChannel*, and are listed in Table 71.

The next higher layer accesses the services provided by the MAC through the MAC sublayer management entity (MLME) service access point (SAP), as described in 6.2, and the MAC common part sublayer (MCPS) SAP, as described in 6.3. The primitives for the MLME SAP are written in all capital letters prefixed with MLME, e.g., MLME-SCAN.confirm. The primitives for the MCPS SAP are written in all capital letters prefixed with MCPS, e.g., MCPS-DATA.request.

5.1.1 Channel access

This subclause describes the mechanisms for accessing the physical radio channel.

5.1.1.1 Superframe structure

A coordinator on a PAN can optionally bound its channel time using a superframe structure. A superframe is bounded by the transmission of a beacon frame and can have an active portion and an inactive portion. The coordinator may enter a low-power (sleep) mode during the inactive portion.

The structure of this superframe is described by the values of *macBeaconOrder* and *macSuperframeOrder*. The MAC PIB attribute *macBeaconOrder* describes the interval at which the coordinator shall transmit its beacon frames. The value of *macBeaconOrder* and the beacon interval, *BI*, are related as follows:

$$BI = aBaseSuperframeDuration \times 2^{macBeaconOrder}$$

for

$$0 \leq macBeaconOrder \leq 14$$

If *macBeaconOrder* = 15, the coordinator shall not transmit beacon frames except when requested to do so, such as on receipt of a beacon request command. The value of *macSuperframeOrder* shall be ignored if *macBeaconOrder* = 15.

The MAC PIB attribute *macSuperframeOrder* describes the length of the active portion of the superframe, which includes the beacon frame. The value of *macSuperframeOrder*, and the superframe duration, *SD*, are related as follows:

$$SD = aBaseSuperframeDuration \times 2^{macSuperframeOrder}$$

for

$$0 \leq macSuperframeOrder \leq macBeaconOrder \leq 14$$

If *macSuperframeOrder* = 15, the superframe shall not remain active after the beacon. If *macBeaconO*rder = 15, the superframe shall not exist (the value of *macSuperframeOrder* shall be ignored), and *macRxOnWhenIdle* shall define whether the receiver is enabled during periods of transceiver inactivity.

The active portion of each superframe shall be divided into *aNumSuperframeSlots* equally spaced slots of duration $2^{macSuperframeOrder} \times aBaseSlotDuration$ and is composed of three parts: a beacon, a CAP, and a CFP. The beacon shall be transmitted, without the use of CSMA, at the start of slot 0, and the CAP shall commence immediately following the beacon. The start of slot 0 is defined as the point at which the first symbol of the beacon PPDU is transmitted. The CFP, if present, follows immediately after the CAP and extends to the end of the active portion of the superframe. Any allocated GTSs shall be located within the CFP.

The MAC sublayer shall ensure that the integrity of the superframe timing is maintained, e.g., compensating for clock drift error.

PANs that wish to use the superframe structure (referred to as beacon-enabled PANs) shall set *macBeaconOrder* to a value between 0 and 14, both inclusive, and *macSuperframeOrder* to a value between 0 and the value of *macBeaconOrder*, both inclusive.

PANs that do not wish to use the superframe structure (referred to as nonbeacon-enabled PANs) shall set both *macBeaconOrder* and *macSuperframeOrder* to 15. In this case, a coordinator shall not transmit beacons, except upon receipt of a beacon request command; all transmissions, with the exception of acknowledgment frames and any data frame that quickly follows the acknowledgment of a data request command, as described in 5.1.6.3, shall use an unslotted CSMA-CA mechanism to access the channel. In addition, GTSs shall not be permitted.

An example of a superframe structure is shown in Figure 8. In this case, the beacon interval, *BI*, is twice as long as the active superframe duration, *SD*, and the CFP contains two GTSs.

5.1.1.1.1 Contention access period (CAP)

The CAP shall start immediately following the beacon and complete before the beginning of the CFP on a superframe slot boundary. If the CFP is zero length, the CAP shall complete at the end of the active portion of the superframe. The CAP shall be at least *aMinCAPLength*, unless additional space is needed to

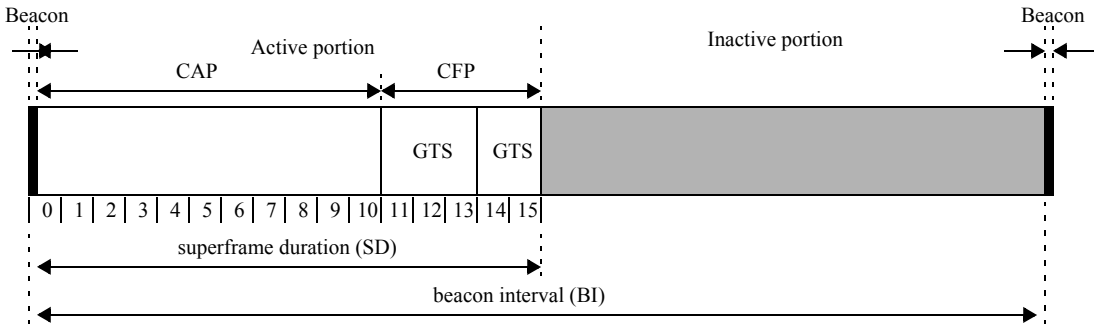


Figure 8—An example of the superframe structure

temporarily accommodate the increase in the beacon frame length needed to perform GTS maintenance, as described in 5.2.2.1.3, and shall shrink or grow dynamically to accommodate the size of the CFP.

All frames, except acknowledgment frames and any data frame that quickly follows the acknowledgment of a data request command, as described in 5.1.6.3, transmitted in the CAP shall use a slotted CSMA-CA mechanism to access the channel. A device transmitting within the CAP shall ensure that its transaction is complete (i.e., including the reception of any acknowledgment) one interframe spacing (IFS) period, as described in 5.1.1.3, before the end of the CAP. If this is not possible, the device shall defer its transmission until the CAP of the following superframe.

MAC command frames shall always be transmitted in the CAP.

5.1.1.1.2 Contention-free period (CFP)

The CFP shall start on a slot boundary immediately following the CAP, and it shall complete before the end of the active portion of the superframe. If any GTSs have been allocated by the PAN coordinator, they shall be located within the CFP and occupy contiguous slots. The CFP shall therefore grow or shrink depending on the total length of all of the combined GTSs.

No transmissions within the CFP shall use a CSMA-CA mechanism to access the channel. A device transmitting in the CFP shall ensure that its transmissions are complete one IFS period, as described in 5.1.1.3, before the end of its GTS.

5.1.1.2 Incoming and outgoing superframe timing

On a beacon-enabled PAN, a coordinator that is not the PAN coordinator shall maintain the timing of both the superframe in which its coordinator transmits a beacon (the incoming superframe) and the superframe in which it transmits its own beacon (the outgoing superframe). The relative timing of these superframes is defined by the StartTime parameter of the MLME-START.request primitive, as described in 6.2.12.1 and 5.1.2.4. The relationship between incoming and outgoing superframes is illustrated in Figure 9.

If a device receives a coordinator realignment command frame from its coordinator indicating that the coordinator will begin using a new superframe configuration, the device shall ensure that its own beacons do not overlap with the beacons transmitted by the coordinator. If the new superframe configuration causes the incoming and outgoing superframes to overlap, the device shall stop transmitting its beacons immediately and notify the next higher layer via the MLME-SYNC-LOSS.indication primitive, as described in 6.2.13.2, with the LossReason parameter set to SUPERFRAME_OVERLAP.

The beacon order and superframe order shall be equal for all superframes on a PAN. All devices shall interact with the PAN only during the active portion of a superframe.

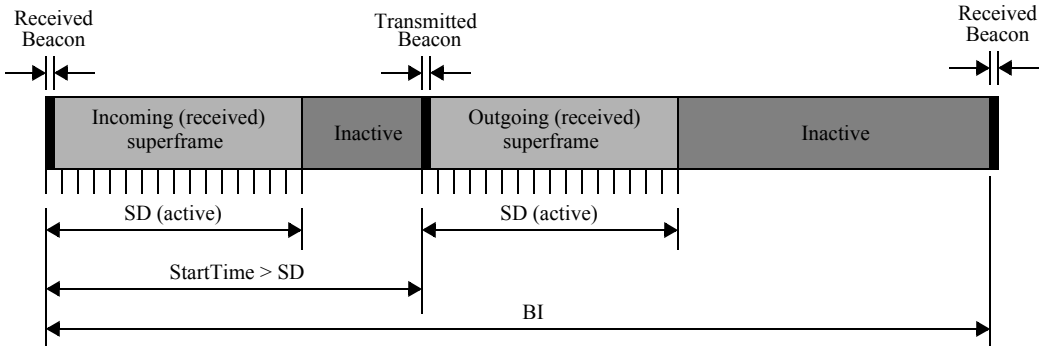


Figure 9—The relationship between incoming and outgoing beacons

5.1.1.3 Interframe spacing (IFS)

The MAC sublayer needs a finite amount of time to process data received by the PHY. To allow for this, two successive frames transmitted from a device shall be separated by at least an IFS period; if the first transmission requires an acknowledgment, the separation between the acknowledgment frame and the second transmission shall be at least an IFS period. The length of the IFS period is dependent on the size of the frame that has just been transmitted. Frames (i.e., MPDUs) of up to *aMaxSIFSFrameSize* shall be followed by a short interfame space (SIFS) period of a duration of at least *macSIFSPeriod*. Frames (i.e., MPDUs) with lengths greater than *aMaxSIFSFrameSize* shall be followed by a long interframe spacing (LIFS) period of a duration of at least *macLIFSPeriod*. These concepts are illustrated in Figure 10.

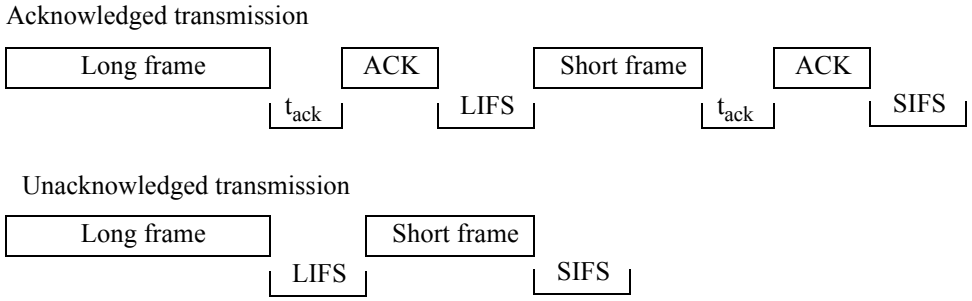


Figure 10—IFS

The CSMA-CA algorithm shall take this requirement into account for transmissions in the CAP. The timing of the transmission of an acknowledgment frame, t_{ack}, is defined in 5.1.6.3 and 5.1.6.4.2.

5.1.1.4 CSMA-CA algorithm

The CSMA-CA algorithm shall be used before the transmission of data or MAC command frames transmitted within the CAP, unless the frame can be quickly transmitted following the acknowledgment of a data request command, as defined in 5.1.6.3. The CSMA-CA algorithm shall not be used for the transmission of beacon frames in a beacon-enabled PAN, acknowledgment frames, or data frames transmitted in the CFP.

If periodic beacons are being used in the PAN, the MAC sublayer shall employ the slotted version of the CSMA-CA algorithm for transmissions in the CAP of the superframe. Conversely, if periodic beacons are not being used in the PAN or if a beacon could not be located in a beacon-enabled PAN, the MAC sublayer shall transmit using the unslotted version of the CSMA-CA algorithm. In both cases, the algorithm is

implemented using units of time called backoff periods, where one backoff period shall be equal to *aUnitBackoffPeriod*.

In slotted CSMA-CA, the backoff period boundaries of every device in the PAN shall be aligned with the superframe slot boundaries of the PAN coordinator; i.e., the start of the first backoff period of each device is aligned with the start of the beacon transmission. In slotted CSMA-CA, the MAC sublayer shall ensure that the PHY commences all of its transmissions on the boundary of a backoff period. In unslotted CSMA-CA, the backoff periods of one device are not related in time to the backoff periods of any other device in the PAN.

Each device shall maintain three variables for each transmission attempt: *NB*, *CW,* and *BE*. *NB* is the number of times the CSMA-CA algorithm was required to back off while attempting the current transmission; this value shall be initialized to zero before each new transmission attempt. *CW* is the contention window length, defining the number of backoff periods that need to be clear of channel activity before the transmission can commence. This value shall be initialized to CW_0 before each transmission attempt and reset to CW_0 each time the channel is assessed to be busy. For operation in the Japanese 950 MHz band, CW_0 shall be set to one; otherwise, CW_0 shall be set to two. The *CW* variable is only used for slotted CSMA-CA. *BE* is the backoff exponent, which is related to how many backoff periods a device shall wait before attempting to assess a channel. In unslotted systems, or slotted systems with the received battery life extension (BLE) field, as defined in Figure 41, set to zero, *BE* shall be initialized to the value of *macMinBE*. In slotted systems with the received BLE field set to one, this value shall be initialized to the lesser of two and the value of *macMinBE*. Note that if *macMinBE* is set to zero, collision avoidance will be disabled during the first iteration of this algorithm.

Although the receiver of the device is enabled during the CCA analysis portion of this algorithm, the device may discard any frames received during this time.

Figure 11 illustrates the steps of the CSMA-CA algorithm. If the algorithm ends in “Success,” the MAC is allowed to begin transmission of the frame. Otherwise, the algorithm terminates with a channel access failure.

In a slotted CSMA-CA system with the BLE field set to zero, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction can be transmitted before the end of the CAP. If the number of backoff periods is greater than the remaining number of backoff periods in the CAP, the MAC sublayer shall pause the backoff countdown at the end of the CAP and resume it at the start of the CAP in the next superframe. If the number of backoff periods is less than or equal to the remaining number of backoff periods in the CAP, the MAC sublayer shall apply its backoff delay and then evaluate whether it can proceed. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps [i.e., two CCA analyses, or a single continuous CCA analysis of at least *phyCCADuration* for the regulatory domains that require listen before talk (LBT) such as the 950 MHz band in Japan, as described in Annex H], the frame transmission, and any acknowledgment can be completed before the end of the CAP. If the MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay before evaluating whether it can proceed again.

In a slotted CSMA-CA system with the BLE field set to one, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction can be transmitted before the end of the CAP. The backoff countdown shall only occur during the first *macBattLifeExtPeriods* full backoff periods after the end of the IFS period following the beacon. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps [two CCA analyses, or a single continuous CCA analysis of at least *phyCCADuration* for the regulatory domains that require listen before talk (LBT) such as the 950 MHz band in Japan, as described in Annex H], the frame transmission, and any acknowledgment can be completed before the end of the CAP, and the frame transmission will start in one of the first *macBattLifeExtPeriods* full backoff periods after the IFS period following the beacon. If the

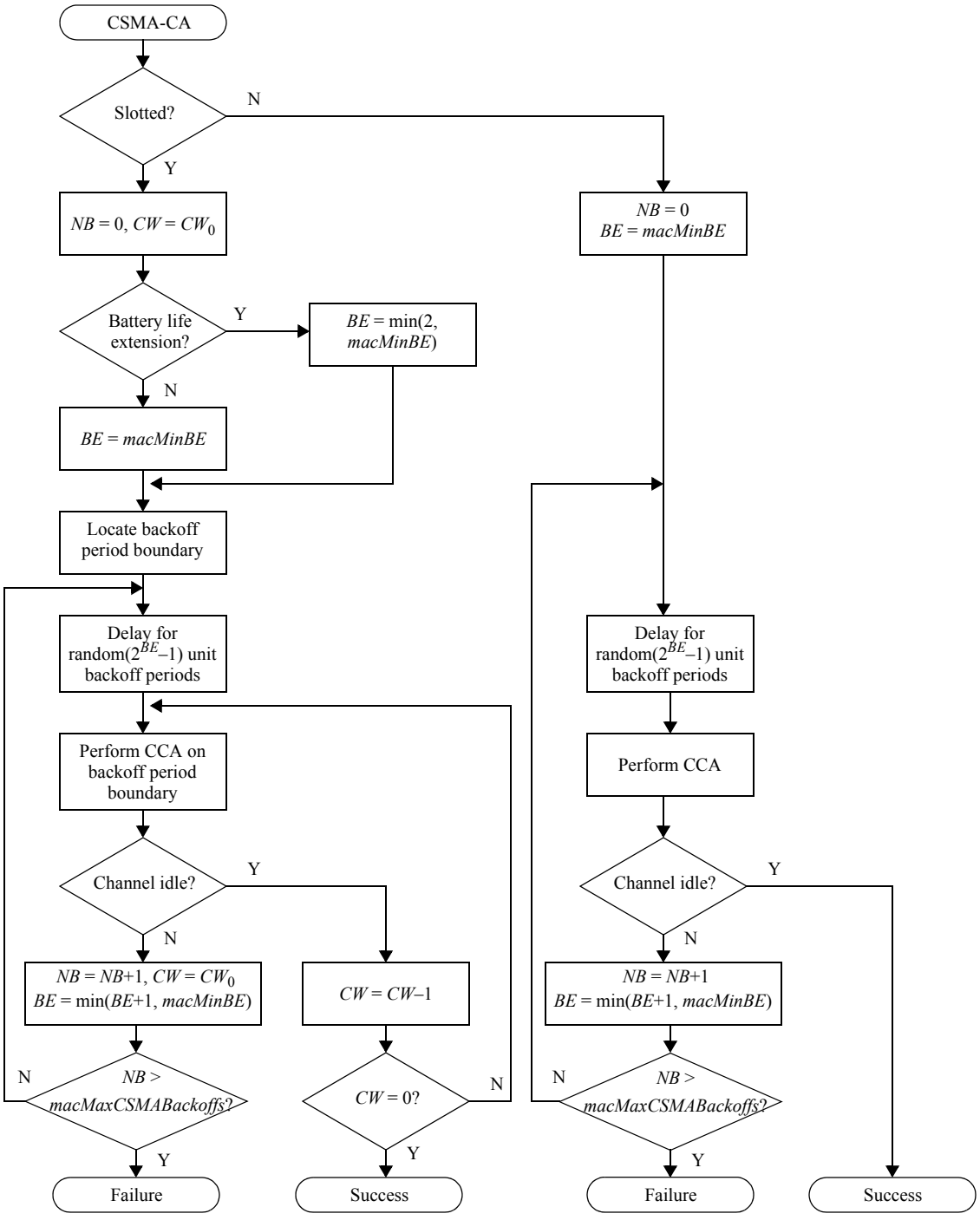


Figure 11—CSMA-CA algorithm

MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay before evaluating whether it can proceed again.

5.1.2 Starting and maintaining PANs

This subclause specifies the procedures for scanning through channels, identifying PAN identifier conflicts, and starting PANs.

5.1.2.1 Scanning through channels

All devices shall be capable of performing passive and orphan scans across a specified list of channels. In addition, an FFD shall be able to perform energy detection (ED) and active scans. The next higher layer should submit a scan request for a particular channel page containing a list of channels chosen only from the channels specified by *phyChannelsSupported* for that particular channel page.

A device is instructed to begin a channel scan through the MLME-SCAN.request primitive, as described in 6.2.10.1. Channels are scanned in order from the lowest channel number to the highest. For the duration of the scan, the device shall suspend beacon transmissions, if applicable, and shall only accept frames received over the PHY data service that are relevant to the scan being performed. For UWB and CSS PHYs, each preamble code appropriate to the specified channel is scanned. Upon the conclusion of the scan, the coordinator device in a beacon-enabled PAN shall recommence beacon transmissions. The results of the scan shall be returned via the MLME-SCAN.confirm primitive as described in 6.2.10.2.

5.1.2.1.1 ED channel scan

An ED scan allows a device to obtain a measure of the peak energy in each requested channel. This could be used by a prospective PAN coordinator to select a channel on which to operate prior to starting a new PAN. During an ED scan, the MAC sublayer shall discard all frames received over the PHY data service.

An ED scan over a specified set of channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an ED scan. For each channel, the MLME shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and then repeatedly perform an ED measurement for [*aBaseSuperframeDuration* × (2^n + 1)], where *n* is the value of the ScanDuration parameter in the MLME-SCAN.request primitive. The maximum ED measurement obtained during this period shall be noted before moving on to the next channel in the channel list. A device shall be able to store at least one channel ED measurement.

The ED scan shall terminate when either the number of channel ED measurements stored equals the implementation-specified maximum or energy has been measured on each of the specified channels.

5.1.2.1.2 Active and passive channel scan

An active or passive channel scan allows a device to locate any coordinator transmitting beacon frames within its radio communications range. An active scan uses the beacon request command to extract the beacon from a coordinator. In a passive scan, the beacon request command is not transmitted. A message sequence chart for active scan is illustrated in Figure 12 and for passive scan in Figure 13.

During an active or passive scan, the MAC sublayer shall discard all frames received over the PHY data service that are not beacon frames. If a beacon frame is received that contains the address of the scanning device in its list of pending addresses, the scanning device shall not attempt to extract the pending data.

Before commencing an active or passive scan, the MAC sublayer shall store the value of *macPANId* and then set it to 0xffff for the duration of the scan. This enables the receive filter to accept all beacons rather than just the beacons from its current PAN, as described in 5.1.6.2. On completion of the scan, the MAC sublayer shall restore the value of *macPANId* to the value stored before the scan began.

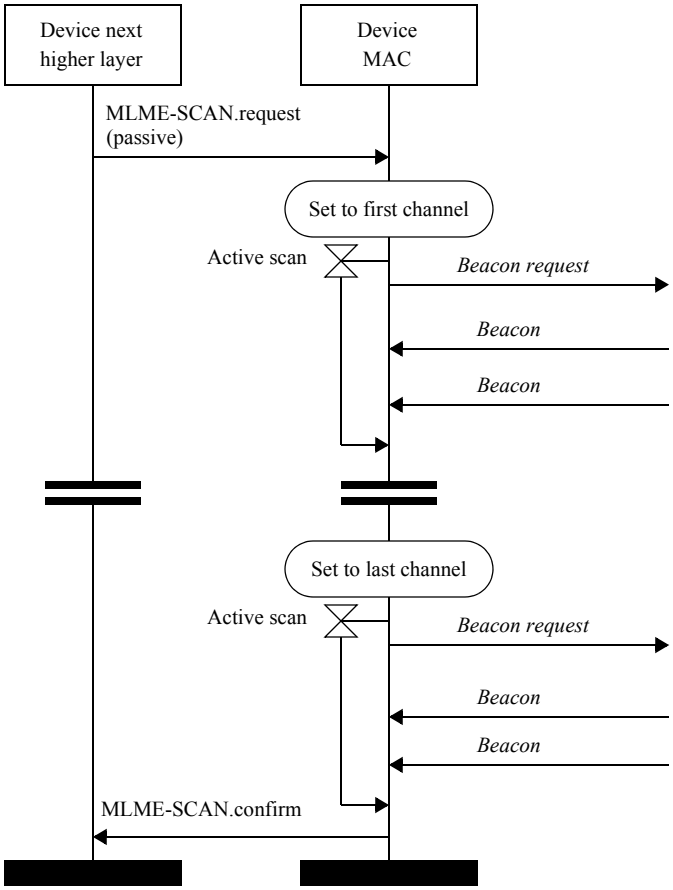


Figure 12—Active scan message sequence chart

An active or passive scan over a specified set of channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an active or passive scan. For each channel, the device shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly. For an active scan, the device shall send a beacon request command, as described in 5.3.7. For UWB and CSS PHYs, the scan process shall be repeated for each mandatory preamble code, setting the *phyCurrentCode* appropriately. Upon successful transmission of the beacon request command for an active scan or after switching to the channel for a passive scan, the device shall enable its receiver for at most [*aBaseSuperframeDuration* × (2^n + 1)], where *n* is the value of the ScanDuration parameter. During this time, the device shall reject all nonbeacon frames and record the information contained in all unique beacons in a PAN descriptor structure, as described in Table 17, including the channel information and, if required, the preamble code.

If a beacon frame is received when *macAutoRequest* is set to TRUE, the list of PAN descriptor structures shall be stored by the MAC sublayer until the scan is complete; at this time, the list shall be sent to the next higher layer in the PANDescriptorList parameter of the MLME-SCAN.confirm primitive. A device shall be able to store at least one PAN descriptor. A beacon frame shall be assumed to be unique if it contains both a PAN identifier and a source address that has not been seen before during the scan of the current channel.

If a beacon frame is received when *macAutoRequest* is set to FALSE, each recorded PAN descriptor is sent to the next higher layer in a separate MLME-BEACON-NOTIFY.indication primitive as described in 6.2.4.1. A received beacon frame containing one or more octets of payload shall also cause the PAN descriptor to be sent to the next higher layer via the MLME-BEACON-NOTIFY.indication primitive. Once

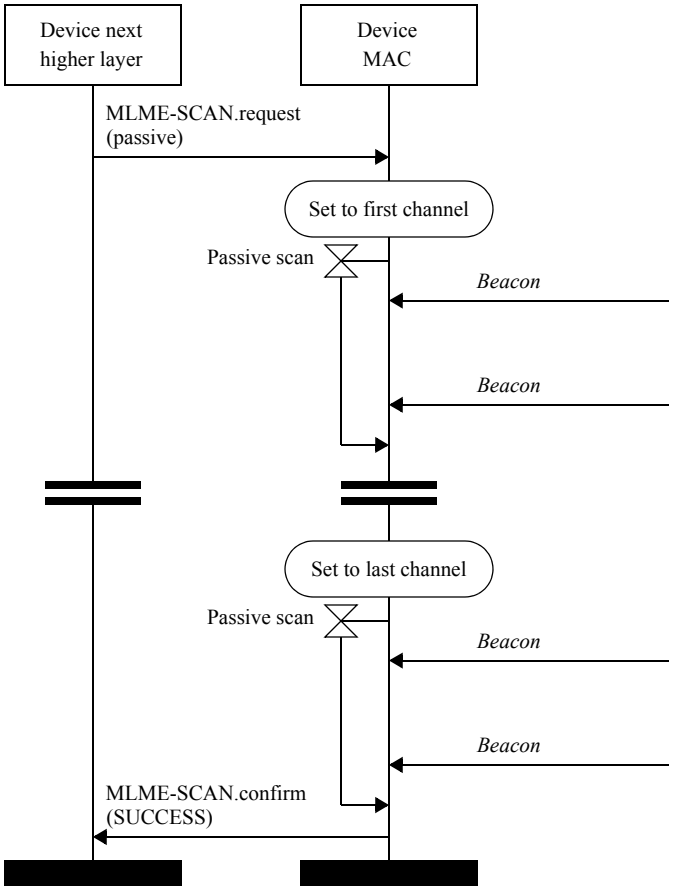


Figure 13—Passive scan message sequence chart

the scan with *macAutoRequest* set to FALSE is complete, the MLME-SCAN.confirm shall be issued to the next higher layer with a null PANDescriptorList.

For UWB and CSS PHYs, the beacon request is repeated for each preamble code.

If a protected beacon frame is received, i.e., the Security Enabled field is set to one, the device shall attempt to unsecure the beacon frame using the unsecuring process described in 7.2.3.

The security-related elements of the PAN descriptor corresponding to the beacon, as defined in Table 17, shall be set to the corresponding parameters returned by the unsecuring process. The SecurityStatus element of the PAN descriptor shall be set to SUCCESS if the status from the unsecuring process is SUCCESS and set to one of the other status codes indicating an error in the security processing otherwise.

The information from the unsecured frame shall be recorded in the PAN descriptor even if the status from the unsecuring process indicated an error.

If a coordinator of a beacon-enabled PAN receives the beacon request command, it shall ignore the command and continue transmitting its periodic beacons as usual. If a coordinator of a nonbeacon-enabled PAN receives this command, it shall transmit a single beacon frame using unslotted CSMA-CA.

If *macAutoRequest* is set to TRUE, the active scan on a particular channel shall terminate when the number of beacons found equals the implementation-specified limit or the channel has been scanned for the full time, as specified in 5.1.2.1.2. If *macAutoRequest* is set to FALSE, the active scan on a particular channel

shall terminate when the channel has been scanned for the full time. If a channel was not scanned for the full time, it shall be considered to be unscanned.

If *macAutoRequest* is set to TRUE, the entire scan procedure shall terminate when the number of PAN descriptors stored equals the implementation-specified maximum or every channel in the set of available channels has been scanned. If *macAutoRequest* is set to FALSE, the entire scan procedure shall only terminate when every channel in the set of available channels has been scanned.

5.1.2.1.3 Orphan channel scan

An orphan scan allows a device to attempt to relocate its coordinator following a loss of synchronization. During an orphan scan, the MAC sublayer shall discard all frames received over the PHY data service that are not coordinator realignment command frames.

An orphan scan over a specified set of channels is requested using the MLME-SCAN.request primitive with the ScanType parameter set to indicate an orphan scan. For each channel, the device shall first switch to the channel, by setting *phyCurrentChannel* and *phyCurrentPage* accordingly, and for UWB and CSS PHYs, setting the preamble code *phyCurrentCode* appropriately, and then send an orphan notification command, as described in 5.3.6. Upon successful transmission of the orphan notification command, the device shall enable its receiver for at most *macResponseWaitTime*. If the device successfully receives a coordinator realignment command, as described in 5.3.8, within this time, the device shall terminate the scan. For the UWB and CSS PHYs, if the coordinator realignment command is not received, the process shall be repeated for each preamble code until a realignment command is received or all preamble codes for the PHY have been used.

The orphan scan shall terminate when the device receives a coordinator realignment command or the specified set of channels has been scanned.

A example message sequence chart for orphan scan and realignment is shown in Figure 14.

If a coordinator receives the orphan notification command, the MLME shall send the MLME-ORPHAN.indication primitive, as described in 6.2.7.1, to the next higher layer. The next higher layer should then search its device list for the device indicated by the primitive. If the next higher layer finds a record of the device, it should send a coordinator realignment command to the orphaned device using the MLME-ORPHAN.response primitive, as described in 6.2.7.2, with the AssociatedMember parameter set to TRUE and the ShortAddress parameter set to the corresponding short address allocated to the orphaned device. The process of searching for the device and sending the coordinator realignment command shall occur within *macResponseWaitTime*. The coordinator realignment command shall contain its current PAN identifier, *macPANId*, its current channel and channel page, and the short address of the orphaned device. If the next higher layer of the coordinator finds no record of the device, it should send the MLME-ORPHAN.response primitive to the MLME with the AssociatedMember parameter set to FALSE.

Figure 15 illustrates the sequence of messages necessary for a coordinator to issue a notification of an orphaned device.

5.1.2.2 PAN identifier conflict resolution

In some instances a situation could occur in which two PANs exist in the same radio communications range with the same PAN identifier. If this conflict happens, the PAN coordinator and its devices shall perform the PAN identifier conflict resolution procedure.

This procedure is optional for an RFD.

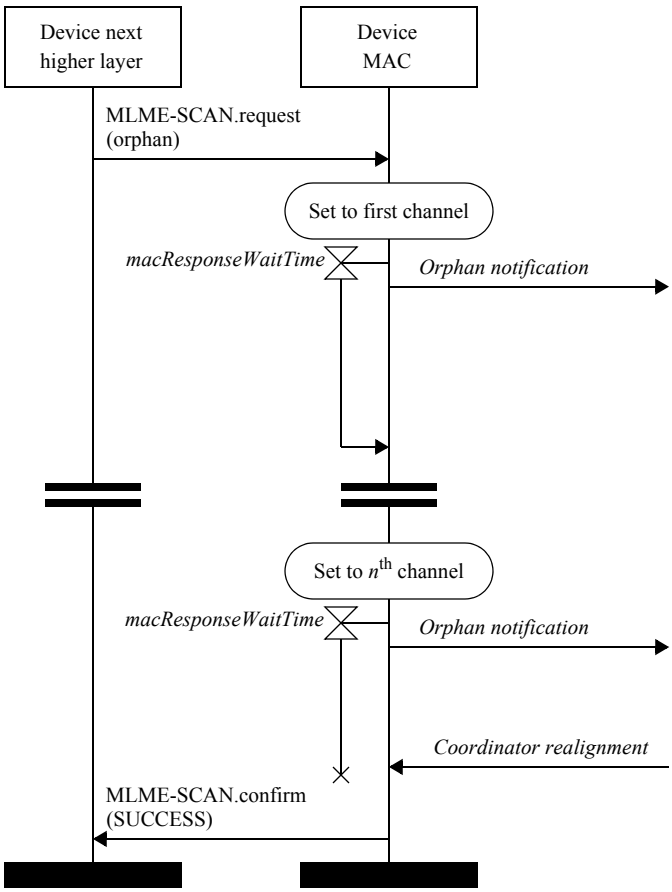


Figure 14—Orphaned device realignment message sequence chart

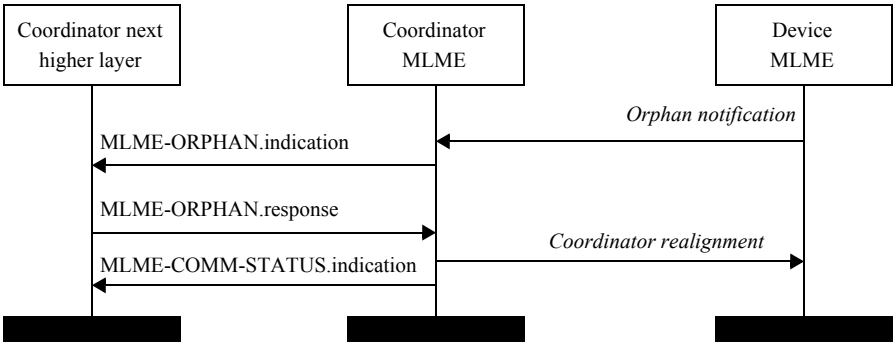


Figure 15—Message sequence chart for orphan notification

5.1.2.2.1 Detection

The PAN coordinator shall conclude that a PAN identifier conflict is present if either of the following apply:

— A beacon frame is received by the PAN coordinator with the PAN Coordinator field, as defined in 5.2.2.1.2, set to one and the PAN identifier equal to *macPANId*.

— A PAN ID conflict notification command, as defined in 5.3.5, is received by the PAN coordinator from a device associated with it on its PAN.

A device that is associated through the PAN coordinator (i.e., *macAssociatedPANCoord* is set to TRUE) shall conclude that a PAN identifier conflict is present if the following applies:

— A beacon frame is received by the device with the PAN Coordinator field set to one, the PAN identifier equal to *macPANId*, and an address that is equal to neither *macCoordShortAddress* nor *macCoordExtendedAddress*.

A device that is associated through a coordinator that is not the PAN coordinator shall not be capable of detecting a PAN identifier conflict.

5.1.2.2.2 Resolution

On the detection of a PAN identifier conflict by a device, it shall generate the PAN ID conflict notification command, as defined in 5.3.5, and send it to its PAN coordinator. Because the PAN ID conflict notification command has the AR field set to request an acknowledgment, the PAN coordinator shall confirm its receipt by sending an acknowledgment frame. Once the device has received the acknowledgment frame from the PAN coordinator, the MLME shall issue an MLME-SYNC-LOSS.indication primitive, as described in 6.2.13.2, with the LossReason parameter set to PAN_ID_CONFLICT. If the device does not receive an acknowledgment frame, the MLME shall not inform the next higher layer of the PAN identifier conflict.

On the detection of a PAN identifier conflict by the PAN coordinator, the MLME shall issue an MLME-SYNC-LOSS.indication to the next higher layer with the LossReason parameter set to PAN_ID_CONFLICT. The next higher layer of the PAN coordinator may then perform an active scan and, using the information from the scan, select a new PAN identifier. The algorithm for selecting a suitable PAN identifier is outside the scope of this standard. If the next higher layer does select a new PAN identifier, it may then issue an MLME-START.request with the CoordRealignment parameter set to TRUE in order to realign the PAN, as described in 5.1.2.3.

5.1.2.3 Starting and realigning a PAN

This subclause specifies procedures for the PAN coordinator starting a PAN, coordinators realigning a PAN, and devices being realigned on a PAN.

5.1.2.3.1 Starting a PAN

A PAN should be started by an FFD only after having first performed a MAC sublayer reset, by issuing the MLME-RESET.request primitive, as described in 6.2.8.1, with the SetDefaultPIB parameter set to TRUE, an active channel scan, and a suitable PAN identifier selection. The algorithm for selecting a suitable PAN identifier from the list of PAN descriptors returned from the active channel scan procedure is outside the scope of this standard. In addition, an FFD should set *macShortAddress* to a value less than 0xffff.

An FFD is instructed to begin operating a PAN through the use of the MLME-START.request primitive, as defined in 6.2.12.1, with the PANCoordinator parameter set to TRUE and the CoordRealigment parameter set to FALSE. On receipt of this primitive, the MAC sublayer shall update the superframe configuration and channel parameters as specified in 5.1.2.3.4. After completing this, the MAC sublayer shall issue the MLME-START.confirm primitive, as described in 6.2.12.2, with a status of SUCCESS and begin operating as the PAN coordinator.

A message sequence chart for starting a PAN is illustrated in Figure 16.

5.1.2.3.2 Realigning a PAN

If a coordinator receives the MLME-START.request primitive, as defined in 6.2.12.1, with the CoordRealignment parameter set to TRUE, the coordinator shall attempt to transmit a coordinator

realignment command containing the new parameters for PANId, ChannelNumber, and, if present, ChannelPage.

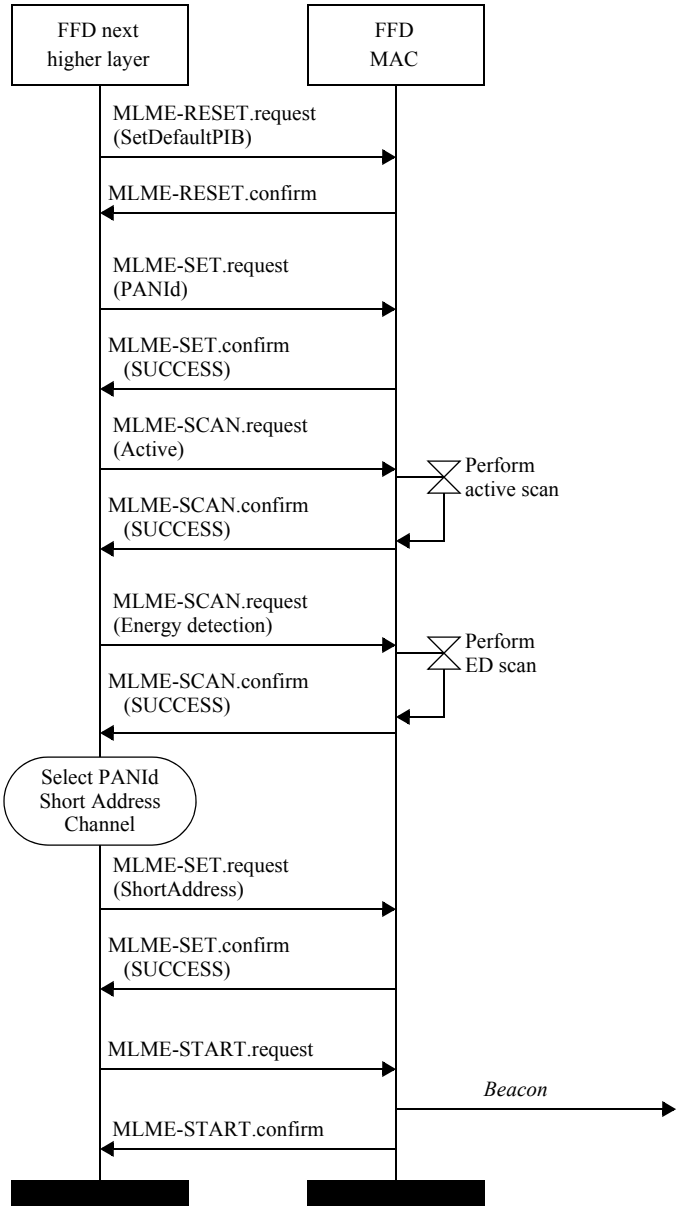


Figure 16—PAN start message sequence chart—PAN coordinator

When the coordinator is already transmitting beacons and the CoordRealignment parameter is set to TRUE, the next scheduled beacon shall be transmitted on the current channel using the current superframe configuration, with the Frame Pending field set to one. Immediately following the transmission of the beacon, the coordinator realignment command shall also be transmitted on the current channel using CSMA-CA.

When the coordinator is not already transmitting beacons and the CoordRealignment parameter is set to TRUE, the coordinator realignment command shall be transmitted immediately on the current channel using CSMA-CA.

If the transmission of the coordinator realignment command fails due to a channel access failure, the MLME shall notify the next higher layer by issuing the MLME-START.confirm primitive with a status of CHANNEL_ACCESS_FAILURE. The next higher layer may then choose to issue the MLME-START.request primitive again.

Upon successful transmission of the coordinator realignment command, the new superframe configuration and channel parameters shall be put into operation as described in 5.1.2.3.4 at the subsequent scheduled beacon, or immediately if the coordinator is not already transmitting beacons, and the MAC sublayer shall issue the MLME-START.confirm primitive with a status of SUCCESS.

5.1.2.3.3 Realignment in a PAN

If a device has received the coordinator realignment command, as defined in 5.3.8, from the coordinator through which it is associated and the MLME was not carrying out an orphan scan, the MLME shall issue the MLME-SYNC-LOSS.indication primitive with the LossReason parameter set to REALIGNMENT and the PANId, ChannelNumber, ChannelPage, and the security-related parameters set to the respective fields in the coordinator realignment command. The next higher layer of a coordinator may then issue an MLME-START.request primitive with the CoordRealignment parameter set to TRUE. The next higher layer of a device that is not a coordinator may instead change the superframe configuration or channel parameters through use of the MLME-SET.request primitive, as described in 6.2.11.1.

5.1.2.3.4 Updating superframe configuration and channel PIB attributes

To update the superframe configuration and channel attributes, the MLME shall assign values from the MLME-START.request primitive parameters to the appropriate PIB attributes. The MLME shall set *macBeaconOrder* to the value of the BeaconOrder parameter. If *macBeaconOrder* is equal to 15, the MLME will also set *macSuperframeOrder* to 15. In this case, this primitive configures a nonbeacon-enabled PAN. If *macBeaconOrder* is less than 15, the MAC sublayer will set *macSuperframeOrder* to the value of the SuperframeOrder parameter. The MAC sublayer shall also update *macPANID* with the value of the PANId parameter and update *phyCurrentPage* and *phyCurrentChannel* with the values of the ChannelPage and ChannelNumber parameters, respectively.

5.1.2.4 Beacon generation

A device shall be permitted to transmit beacon frames only if *macShortAddress* is not equal to 0xffff.

An FFD shall use the MLME-START.request primitive to begin transmitting beacons only if the BeaconOrder parameter is less than 15. The FFD may begin beacon transmission either as the PAN coordinator of a new PAN or as a device on a previously established PAN, depending upon the setting of the PANCoordinator parameter, as defined in 6.2.12.1. The FFD shall begin beacon transmission on a previously established PAN only once it has successfully associated with that PAN.

If the FFD is the PAN coordinator (i.e., the PANCoordinator parameter is set to TRUE), the MAC sublayer shall ignore the StartTime parameter and begin beacon transmissions immediately. Setting the StartTime parameter to zero shall also cause the MAC sublayer to begin beacon transmissions immediately. If the FFD is not the PAN coordinator and the StartTime parameter is nonzero, the time to begin beacon transmissions shall be calculated using the following method. The StartTime parameter, which is rounded to a backoff period boundary, shall be added to the time, obtained from the local clock, when the MAC sublayer receives the beacon of the coordinator through which it is associated. The MAC sublayer shall then begin beacon transmissions when the current time, obtained from the local clock, equals the calculated time. In order for the beacon transmission time to be calculated by the MAC sublayer, the MAC sublayer shall first track the beacon of the coordinator through which it is associated. If the MLME-START.request primitive is issued with a nonzero StartTime parameter and the MAC sublayer is not currently tracking the beacon of its

coordinator, the MLME shall not begin beacon transmissions but shall instead issue the MLME-START.confirm primitive with a status of TRACKING_OFF.

If a device misses between one and (*aMaxLostBeacons*–1) consecutive beacon frames from its coordinator, the device shall continue to transmit its own beacons based on both *macBeaconOrder*, as defined in 5.1.2.3.4, and its local clock. If the device then receives a beacon frame from its coordinator and, therefore, does not lose synchronization, the device shall resume transmitting its own beacons based on the StartTime parameter and the incoming beacon. If a device does lose synchronization with its coordinator, the MLME of the device shall issue the MLME-SYNC-LOSS.indication primitive to the next higher layer and immediately stop transmitting its own beacons. The next higher layer may, at any time following the reception of the MLME-SYNC-LOSS.indication primitive, resume beacon transmissions by issuing a new MLME-START.request primitive.

On receipt of the MLME-START.request primitive, the MAC sublayer shall set the PAN identifier in *macPANId* and use this value in the Source PAN Identifier field of the beacon frame. The address used in the Source Address field of the beacon frame shall contain the value of *macExtendedAddress* if *macShortAddress* is equal to 0xfffe or *macShortAddress* otherwise.

The time of transmission of the most recent beacon shall be recorded in *macBeaconTxTime* and shall be computed so that its value is taken at the same symbol boundary in each beacon frame, the location of which is implementation specific. The symbol boundary, which is specified by the *macSyncSymbolOffset* attribute, is the same as that used in the timestamp of the incoming beacon frame, as described in 5.1.4.1.

All beacon frames, as defined in 5.2.2.1, shall be transmitted at the beginning of each superframe at an interval equal to *aBase-SuperframeDuration* × 2^n, where *n* is the value of *macBeaconOrder*.

For devices operating in beacon-enabled mode in the Japanese 950 MHz band, a coordinator may precede beacon transmission with listen before talk (LBT) without random backoff. The MAC shall ensure that the beacon is transmitted at the beginning of the superframe with accurate timing.

Beacon transmissions shall be given priority over all other transmit and receive operations.

5.1.2.5 Device discovery

The PAN coordinator or a coordinator indicates its presence on a PAN to other devices by transmitting beacon frames. This allows the other devices to perform device discovery.

A coordinator that is not the PAN coordinator shall begin transmitting beacon frames only when it has successfully associated with a PAN. The transmission of beacon frames by the device is initiated through the use of the MLME-START.request primitive with the PANCoordinator parameter set to FALSE. On receipt of this primitive, the MLME shall begin transmitting beacons based on the StartTime parameter, as described in 5.1.2.4, using the identifier of the PAN with which the device has associated, *macPANId*, and its extended address, *macExtendedAddress*, if *macShortAddress* is equal to 0xfffe, or its short address, *macShortAddress*, otherwise. A beacon frame shall be transmitted at a rate of one beacon frame every *aBaseSuperframeDuration* × 2^n, where *n* is the value of *macBeaconOrder*.

5.1.3 Association and disassociation

This subclause specifies the procedures for association and disassociation.

5.1.3.1 Association

The next higher layer shall attempt to associate only after having first performed a MAC sublayer reset, by issuing the MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE, and then

having completed either an active or a passive channel scan, as defined in 5.1.2.1.2. The results of the channel scan would have then been used to choose a suitable PAN. The algorithm for selecting a suitable PAN with which to associate from the list of PAN descriptors returned from the channel scan procedure is outside the scope of this standard.

Following the selection of a PAN with which to associate, the next higher layers shall request through the MLME-ASSOCIATE.request primitive, as described in 6.2.2.1, that the MLME configures the following PHY and MAC PIB attributes to the values necessary for association:

— *phyCurrentChannel* shall be set equal to the ChannelNumber parameter of the MLME-ASSOCIATE.request primitive.
— *phyCurrentPage* shall be set equal to the ChannelPage parameter of the MLME-ASSOCIATE.request primitive.
— *macPANId* shall be set equal to the CoordPANId parameter of the MLME-ASSOCIATE.request primitive.
— *macCoordExtendedAddress* or *macCoordShortAddress*, depending on which is known from the beacon frame from the coordinator through which it wishes to associate, shall be set equal to the CoordAddress parameter of the MLME-ASSOCIATE.request primitive.

A coordinator shall allow association only if *macAssociationPermit* is set to TRUE. Similarly, a device should attempt to associate only with a PAN through a coordinator that is currently allowing association, as indicated in the results of the scanning procedure. If a coordinator with *macAssociationPermit* set to FALSE receives an association request command from a device, the command shall be ignored.

A device that is instructed to associate with a PAN, through the MLME-ASSOCIATE.request primitive, shall try to associate only with an existing PAN and shall not attempt to start its own PAN.

The MAC sublayer of an unassociated device shall initiate the association procedure by sending an association request command, as described in 5.3.1, to the coordinator of an existing PAN; if the association request command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer.

The acknowledgment to an association request command does not mean that the device has associated. The next higher layer of the coordinator needs time to determine whether the current resources available on the PAN are sufficient to allow another device to associate. The next higher layer should make this decision within *macResponseWaitTime*. If the next higher layer of the coordinator finds that the device was previously associated on its PAN, all previously obtained device-specific information should be replaced. If sufficient resources are available, the next higher layer should allocate a short address to the device, and the MAC sublayer shall generate an association response command, as described in 5.3.2, containing the new address and a status indicating a successful association. If sufficient resources are not available, the next higher layer of the coordinator should inform the MAC sublayer, and the MLME shall generate an association response command containing a status indicating a failure, as defined in Table 6. The association response command shall be sent to the device requesting association using indirect transmission; i.e., the association response command frame shall be added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using the method described in 5.1.6.3.

If the Allocate Address field of the Capability Information field, as described in 5.3.1.2, of the association request command is set to one, the next higher layer of the coordinator shall allocate a address with a range depending on the addressing mode supported by the coordinator, as described in Table 1. If the Allocate Address field of the association request command is set to zero, the short address shall be equal to 0xfffe. A short address of 0xfffe is a special case that indicates that the device has associated but has not been allocated a short address by the coordinator. In this case, the device shall use only its extended address to operate on the network.

On receipt of the acknowledgment to the association request command, the device shall wait for at most *macResponseWaitTime* for the coordinator to make its association decision; the PIB attribute *macResponseWaitTime* is a network-topology-dependent parameter and may be set to match the specific requirements of the network that a device is trying to join. If the device is tracking the beacon, it shall attempt to extract the association response command from the coordinator whenever it is indicated in the beacon frame. If the device is not tracking the beacon, it shall attempt to extract the association response command from the coordinator after *macResponseWaitTime*. If the device does not extract an association response command frame from the coordinator within *macResponseWaitTime*, the MLME shall issue the MLME-ASSOCIATE.confirm primitive, as described in 6.2.2.4, with a status of NO_DATA, and the association attempt shall be deemed a failure. In this case, the next higher layer shall terminate any tracking of the beacon. This is achieved by issuing the MLME-SYNC.request primitive, as described in 6.2.13.1, with the TrackBeacon parameter set to FALSE.

If the Association Status field of the association response command indicates that the association was successful, the device shall store the address contained in the Short Address field of the command in *macShortAddress*; communication on the PAN using this short address shall depend on its range, as described in Table 1. If the original beacon selected for association following a scan contained the short address of the coordinator, the extended address of the coordinator, contained in the MHR of the association response command frame, shall be stored in *macCoordExtendedAddress.*

Table 1—Usage of the short address

Value of *macShortAddress*	Description
0x0000–0xfffd	If a source address is included, the device shall use short source addressing mode for beacon and data frames and the appropriate source addressing mode specified in 5.3 for MAC command frames.
0xfffe	If a source address is included, the device shall use extended source addressing mode for beacon and data frames and the appropriate source addressing mode specified in 5.3 for MAC command frames.
0xffff	The device is not associated and, therefore, shall not perform any data frame communication. The device shall use the appropriate source addressing mode specified in 5.3 for MAC command frames.

If the value of the Association Status field of the command is not “Association successful,” if there were a communication failure during the association process due to a missed acknowledgment, or if the association response command frame were not received, the device shall set *macPANId* to the default value (0xffff).

A message sequence chart for association is illustrated in Figure 17.

Figure 18 illustrates a sequence of messages that may be used by a device that is not tracking the beacon of the coordinator to successfully associate with a PAN.

5.1.3.2 Disassociation

The disassociation procedure is initiated by the next higher layer by issuing the MLME-DISASSOCIATE.request primitive, as described in 6.2.3.1, to the MLME.

When a coordinator wants one of its associated devices to leave the PAN, the MLME of the coordinator shall send the disassociation notification command in the manner specified by the TxIndirect parameter of the MLME-DISASSOCIATE.request primitive previously sent by the next higher layer. If TxIndirect is TRUE, the MLME of the coordinator shall send the disassociation notification command to the device using indirect transmission; i.e., the disassociation notification command frame shall be added to the list of pending transactions stored on the coordinator and extracted at the discretion of the device concerned using

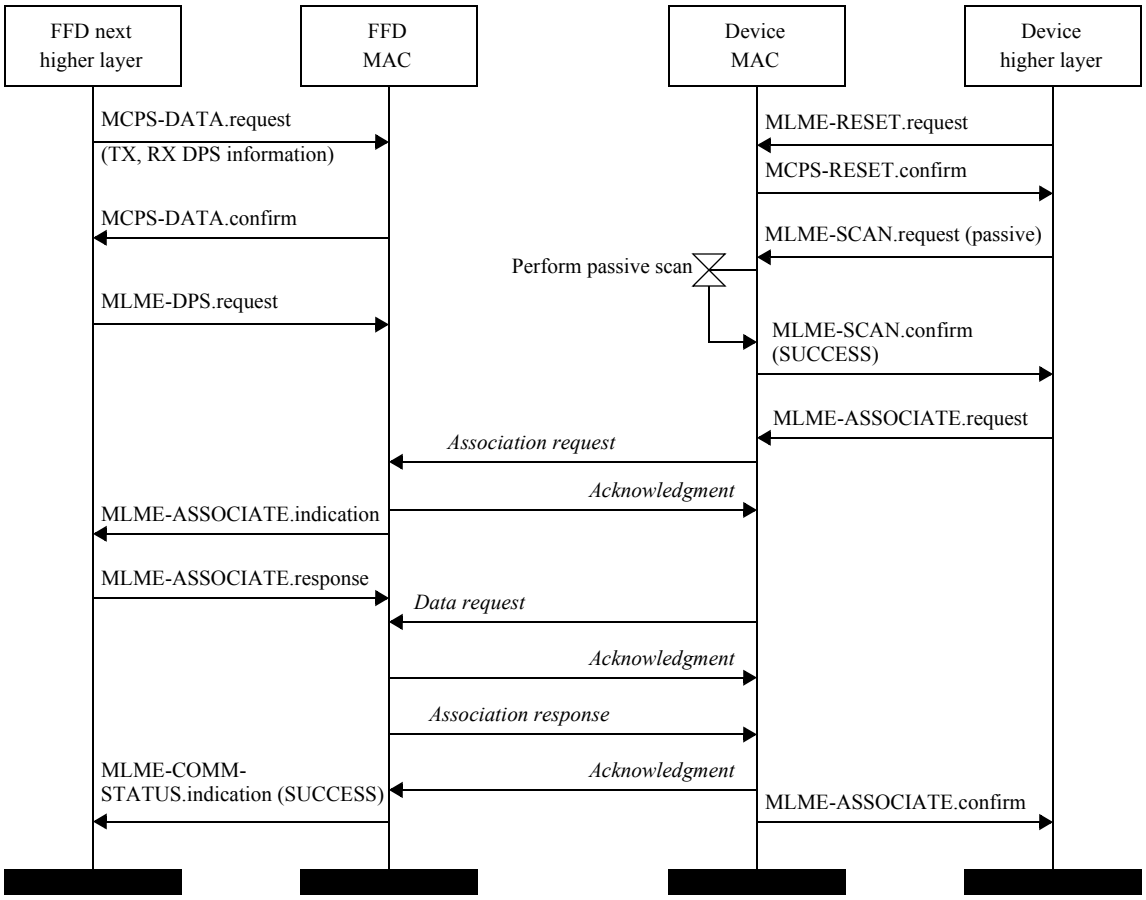


Figure 17—Association message sequence chart

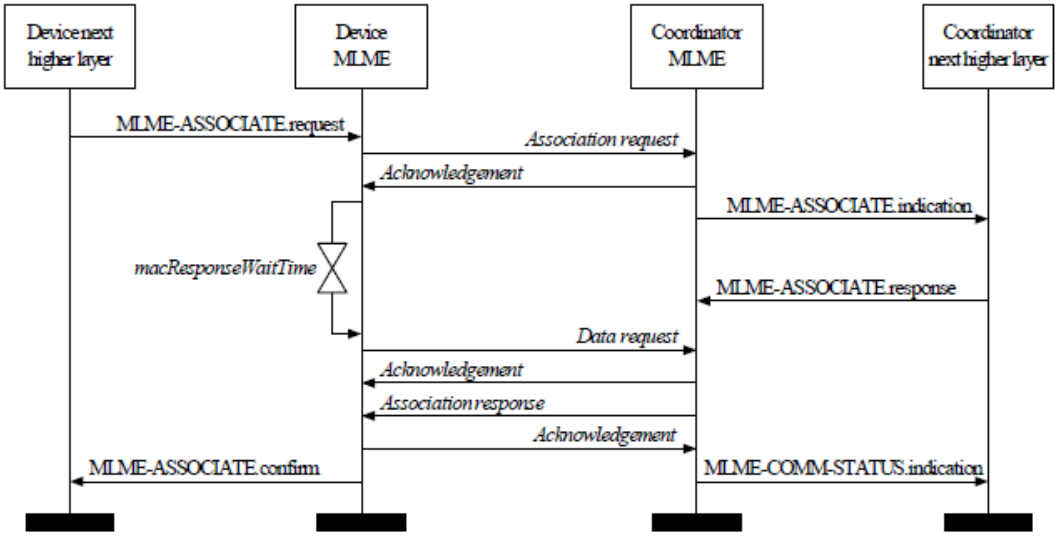


Figure 18—Message sequence chart for association

the method described in 5.1.6.3. If the command frame is not successfully extracted by the device, the coordinator should consider the device disassociated. Otherwise, the MLME shall send the disassociation

notification command to the device directly. In this case, if the disassociation notification command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer.

If the direct or indirect transmission fails, the coordinator should consider the device disassociated.

If an associated device wants to leave the PAN, the MLME of the device shall send a disassociation notification command to its coordinator. If the disassociation notification command cannot be sent due to a channel access failure, the MAC sublayer shall notify the next higher layer. If the acknowledgment to disassociation request is not received, the device should consider itself disassociated.

If the source address contained in the disassociation notification command is equal to *macCoord-ExtendedAddress*, the device should consider itself disassociated. If the command is received by a coordinator and the source is not equal to *macCoordExtendedAddress*, it shall verify that the source address corresponds to one of its associated devices; if so, the coordinator should consider the device disassociated. If none of these conditions are satisfied, the disassociation notification command shall be ignored.

An associated device shall disassociate itself by removing all references to the PAN; the MLME shall set *macPANId*, *macShortAddress*, *macAssociatedPANCoord*, *macCoordShortAddress*, and *macCoordExtended-Address* to the default values. The next higher layer of a coordinator should disassociate a device by removing all references to that device.

The next higher layer of the requesting device shall be notified of the result of the disassociation procedure through the MLME-DISASSOCIATE.confirm primitive, as described in 6.2.3.3.

Figure 19 illustrates the sequence of messages for a device to disassociate itself from the PAN.

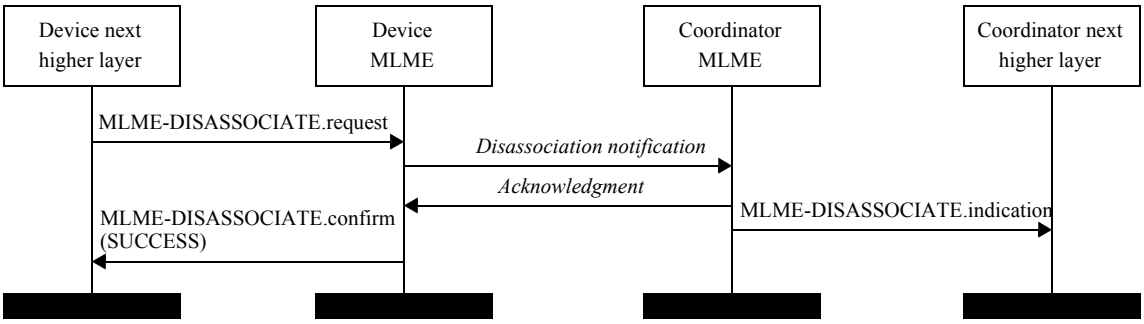


Figure 19—Message sequence chart for disassociation initiated by a device

Figure 20 illustrates the sequence necessary for a coordinator in a beacon-enabled PAN to successfully disassociate a device from its PAN using indirect transmission.

5.1.4 Synchronization

This subclause specifies the procedures for coordinators to generate beacon frames and for devices to synchronize with a coordinator. For PANs supporting beacons, synchronization is performed by receiving and decoding the beacon frames. For PANs not supporting beacons, synchronization is performed by polling the coordinator for data.

5.1.4.1 Synchronization with beacons

All devices operating on a beacon-enabled PAN (i.e., *macBeaconOrder* < 15) shall be able to acquire beacon sychronization in order to detect any pending messages or to track the beacon. Devices shall be permitted to acquire beacon synchronization only with beacons containing the PAN identifier specified in

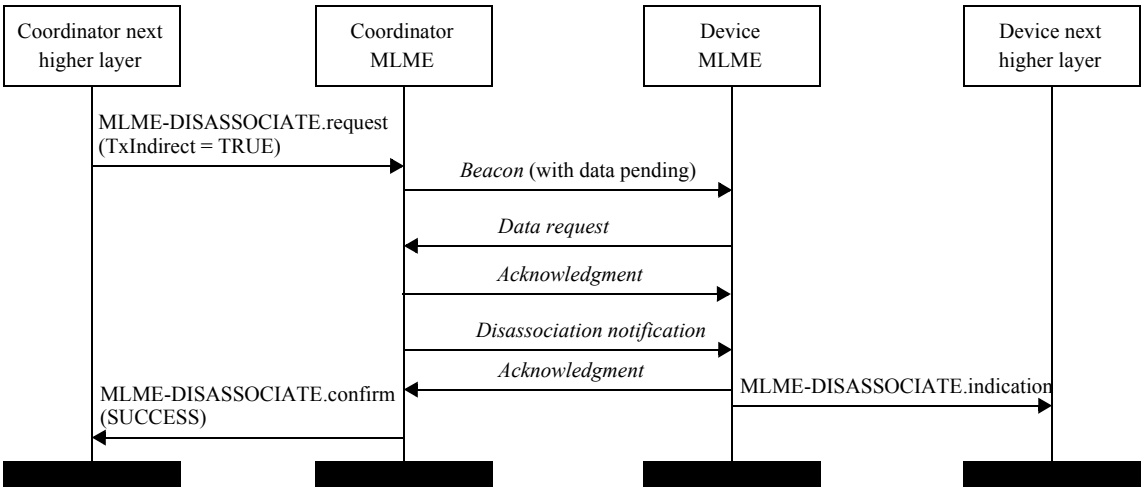


Figure 20—Message sequence chart for disassociation initiated by a coordinator, using indirect transmission, in a beacon-enabled PAN

macPANId. If *macPANId* specifies the broadcast PAN identifier, a device shall not attempt to acquire beacon synchronization.

A device is instructed to attempt to acquire the beacon through the MLME-SYNC.request primitive. If tracking is specified in the MLME-SYNC.request primitive, the device shall attempt to acquire the beacon and keep track of it by regular and timely activation of its receiver. If tracking is not specified, the device shall either attempt to acquire the beacon only once or terminate the tracking after the next beacon if tracking was enabled through a previous request.

To acquire beacon synchronization, a device shall enable its receiver and search for at most [*aBaseSuperframeDuration* × (2^n + 1)], where *n* is the value of *macBeaconOrder*. If a beacon frame containing the current PAN identifier of the device is not received, the MLME shall repeat this search. Once the number of missed beacons reaches *aMaxLostBeacons*, the MLME shall notify the next higher layer by issuing the MLME-SYNC-LOSS.indication primitive with a loss reason of BEACON_LOSS.

The MLME shall timestamp each received beacon frame at the same symbol boundary within each frame, the location of which is described by the *macSyncSymbolOffset* attribute. The symbol boundary shall be the same as that used in the timestamp of the outgoing beacon frame, stored in *macBeaconTxTime*. The timestamp value shall be that of the local clock of the device at the time of the symbol boundary. The timestamp is intended to be a relative time measurement that may or may not be made absolute, at the discretion of the implementer.

If a protected beacon frame is received (i.e., the Security Enabled field is set to one), the device shall attempt to unsecure the beacon frame using the unsecuring process described in 7.2.3.

If the status from the unsecuring process is not SUCCESS, the MLME shall issue an MLME-COMM-STATUS.indication primitive, as described in 6.2.4.2, with the status parameter set to the status from the unsecuring process, indicating the error.

The security-related elements of the PAN descriptor corresponding to the beacon, as defined in Table 17, shall be set to the corresponding parameters returned by the unsecuring process. The SecurityStatus element of the PAN descriptor shall be set to SUCCESS if the status from the unsecuring process is SUCCESS and set to one of the other status codes indicating an error in the security processing otherwise.

If a beacon frame is received, the MLME shall discard the beacon frame if the Source Address and the Source PAN Identifier fields of the MHR of the beacon frame do not match the coordinator source address (*macCoordShortAddress* or *macCoordExtendedAddress*, depending on the addressing mode) and the PAN identifier of the device (*macPANId*).

If a valid beacon frame is received and *macAutoRequest* is set to FALSE, the MLME shall indicate the beacon parameters to the next higher layer by issuing the MLME-BEACON-NOTIFY.indication primitive. If a beacon frame is received and *macAutoRequest* is set to TRUE, the MLME shall first issue the MLME-BEACON-NOTIFY.indication primitive if the beacon contains any payload. The MLME shall then compare its address with those addresses in the Address List field of the beacon frame. If the Address List field contains the short address or extended address of the device and the source PAN identifier matches *macPANId*, the MLME shall follow the procedure for extracting pending data from the coordinator, as described in 5.1.6.3.

If beacon tracking is activated, the MLME shall enable its receiver at a time prior to the next expected beacon frame transmission, i.e., just before the known start of the next superframe. If the number of consecutive beacons missed by the MLME reaches *aMaxLostBeacons*, the MLME shall respond with the MLME-SYNC-LOSS.indication primitive with a loss reason of BEACON_LOST.

In Figure 21 the next higher layer issues a synchronization request with TrackBeacon set to FALSE. The MLME then searches for a beacon and, if found, determines whether the coordinator has any data pending for the device. If so, the data are requested as described in 5.1.6.3.

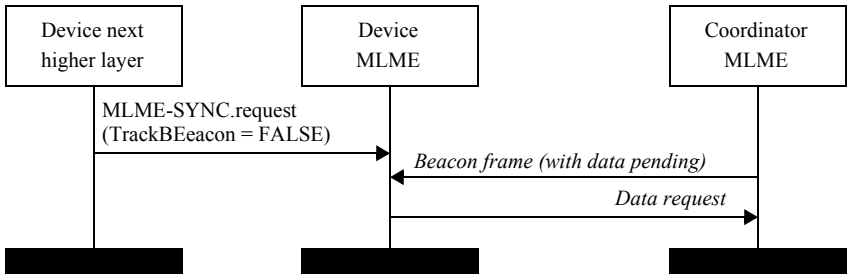


Figure 21—Synchronizing to a coordinator in a beacon-enabled PAN without tracking beacons

In Figure 22, the next higher layer issues a synchronization request with TrackBeacon set to TRUE. The MLME then searches for a beacon and, if found, attempts to keep track of it using a timer that expires just before the expected time of the next beacon.

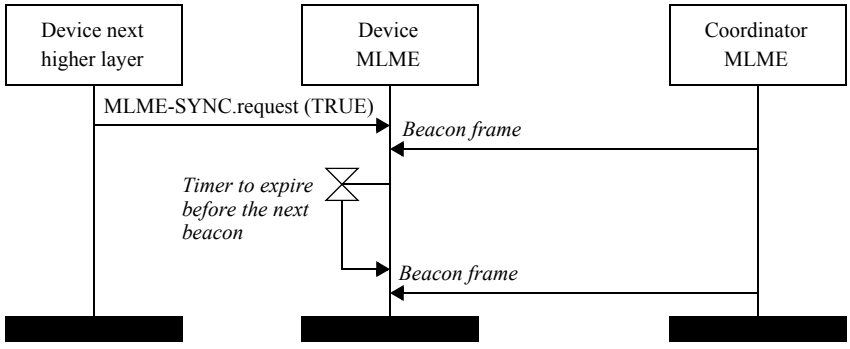


Figure 22—Synchronizing to a coordinator in a beacon-enabled PAN while tracking beacons

5.1.4.2 Synchronization without beacons

All devices operating on a nonbeacon-enabled PAN (*macBeaconOrder* = 15) shall be able to poll the coordinator for data at the discretion of the next higher layer.

A device is instructed to poll the coordinator when the MLME receives the MLME-POLL.request primitive, as described in 6.2.14.1. On receipt of this primitive, the MLME shall follow the procedure for extracting pending data from the coordinator, as described in 5.1.6.3.

5.1.4.3 Orphaned device realignment

If the next higher layer receives repeated communications failures following its requests to transmit data, it may conclude that it has been orphaned. A single communications failure occurs when a device transaction fails to reach the coordinator; i.e., an acknowledgment is not received after *macMaxFrameRetries* attempts at sending the data. If the next higher layer concludes that it has been orphaned, it may instruct the MLME to either perform the orphaned device realignment procedure or reset the MAC sublayer and then perform the association procedure.

If the decision has been made by the next higher layer to perform the orphaned device realignment procedure, it will have issued an MLME-SCAN.request with the ScanType parameter set to orphan scan and the ScanChannel parameter containing the list of channels to be scanned. Upon receiving this primitive, the MAC sublayer shall begin an orphan scan, as described in 5.1.2.1.3.

If the orphan scan is successful (i.e., its PAN has been located), the device shall update its MAC PIB with the PAN information contained in the coordinator realignment command, as described in 5.3.8.

5.1.5 Transaction handling

Because this standard favors very low cost devices that, in general, will be battery powered, transactions can be instigated from the devices themselves rather than from the coordinator. In other words, either the coordinator needs to indicate in its beacon when messages are pending for devices or the devices themselves need to poll the coordinator to determine whether they have any messages pending. Such transfers are called indirect transmissions.

The coordinator shall begin handling a transaction on receipt of an indirect transmission request either via the MCPS-DATA.request primitive or via a request from the MLME to send a MAC command instigated by a primitive from the next higher layer, such as the MLME-ASSOCIATE.response primitive, as described in 6.2.2.3. On completion of the transaction, the MAC sublayer shall indicate a status value to the next higher layer. If a request primitive instigated the indirect transmission, the corresponding confirm primitive shall be used to convey the appropriate status value. Conversely, if a response primitive instigated the indirect transmission, the MLME-COMM-STATUS.indication primitive shall be used to convey the appropriate status value. The MLME-COMM-STATUS.indication primitive can be related to its corresponding response primitive by examining the Destination Address field.

The information contained in the indirect transmission request forms a transaction, and the coordinator shall be capable of storing at least one transaction. On receipt of an indirect transmission request, if there is no capacity to store another transaction, the MAC sublayer shall indicate to the next higher layer a status of TRANSACTION_OVERFLOW in the appropriate corresponding primitive.

If the coordinator is capable of storing more than one transaction, it shall ensure that all the transactions for the same device are sent in the order in which they arrived at the MAC sublayer. For each transaction sent, if another exists for the same device, the MAC sublayer shall set its Frame Pending field to one, indicating the additional pending data.

Each transaction shall persist in the coordinator for at most *macTransactionPersistenceTime*. If the transaction is not successfully extracted by the appropriate device within this time, the transaction information shall be discarded and the MAC sublayer shall indicate to the next higher layer a status of TRANSACTION_EXPIRED in the appropriate corresponding primitive. In order to be successfully extracted, an acknowledgment shall be received if one was requested.

If the transaction was successful, the transaction information shall be discarded, and the MAC sublayer shall indicate to the next higher layer a status of SUCCESS in the appropriate corresponding primitive.

If the coordinator transmits beacons, it shall list the addresses of the devices to which each transaction is associated in the Address List field and indicate the number of addresses in the Pending Address Specification field of the beacon frame. If the coordinator is able to store more than seven pending transactions, it shall indicate them in its beacon on a first-come-first-served basis, ensuring that the beacon frame contains at most seven addresses. For transactions requiring a GTS, the PAN coordinator shall not add the address of the recipient to its list of pending addresses in the beacon frame. Instead it shall transmit the transaction in the GTS allocated for the device, as described in 5.1.7.3.

On a beacon-enabled PAN, if there is a transaction pending for the broadcast address, the Frame Pending field in the beacon frame shall be set to one, and the pending message shall be transmitted immediately following the beacon using the CSMA-CA algorithm. If there is a second message pending for the broadcast address, its transmission shall be delayed until the following superframe. Only one broadcast message shall be allowed to be sent indirectly per superframe.

On a beacon-enabled PAN, a device that receives a beacon containing its address in the list of pending addresses shall attempt to extract the data from the coordinator. On a nonbeacon-enabled PAN, a device shall attempt to extract the data from the coordinator on receipt of the MLME-POLL.request primitive. The procedure for extracting pending data from the coordinator is described in 5.1.6.3. If a device receives a beacon with the Frame Pending field set to one, it shall leave its receiver enabled for up to *macMaxFrameTotalWaitTime* to receive the broadcast data frame from the coordinator.

5.1.6 Transmission, reception, and acknowledgment

This subclause describes the fundamental procedures for transmission, reception, and acknowledgment.

5.1.6.1 Transmission

Each device shall store its current DSN value in the MAC PIB attribute *macDSN* and initialize it to a random value; the algorithm for choosing a random number is outside the scope of this standard. Each time a data or a MAC command frame is generated, the MAC sublayer shall copy the value of *macDSN* into the Sequence Number field of the MHR of the outgoing frame and then increment it by one. Each device shall generate exactly one DSN regardless of the number of unique devices with which it wishes to communicate. The value of *macDSN* shall be permitted to roll over.

Each coordinator shall store its current BSN value in the MAC PIB attribute *macBSN* and initialize it to a random value; the algorithm for choosing a random number is outside the scope of this standard. Each time a beacon frame is generated, the MAC sublayer shall copy the value of *macBSN* into the Sequence Number field of the MHR of the outgoing frame and then increment it by one. The value of *macBSN* shall be permitted to roll over.

The Source Address field, if present, shall contain the address of the device sending the frame. When a device has associated and has been allocated a short address (i.e., *macShortAddress* is not equal to 0xfffe or 0xffff), it shall use that address in preference to its extended address (i.e., *macExtendedAddress*) wherever possible. When a device has not yet associated to a PAN, it shall use its extended address in all communications requiring the Source Address field. If the Source Address field is not present, the originator

of the frame shall be assumed to be the PAN coordinator, and the Destination Address field shall contain the address of the recipient.

The Destination Address field, if present, shall contain the address of the intended recipient of the frame, which may be either a short address or an extended address. If the Destination Address field is not present, the recipient of the frame shall be assumed to be the PAN coordinator, and the Source Address field shall contain the address of the originator.

If both destination and source addressing information is present, the MAC sublayer shall compare the destination and source PAN identifiers. If the PAN identifiers are identical, the PAN ID Compression field shall be set to one, and the source PAN identifier shall be omitted from the transmitted frame. If the PAN identifiers are different, the PAN ID Compression field shall be set to zero, and both Destination PAN Identifier and Source PAN Identifier fields shall be included in the transmitted frame. If only either the destination or the source addressing information is present, the PAN ID Compression field shall be set to zero, and the PAN identifier field of the single address shall be included in the transmitted frame.

If the frame is to be transmitted on a beacon-enabled PAN, the transmitting device shall attempt to find the beacon before transmitting. If the beacon is not being tracked, as described in 5.1.4.1, and hence the device does not know where the beacon will appear, it shall enable its receiver and search for at most [*aBaseSuperframeDuration* × $(2^n + 1)$], where *n* is the value of *macBeaconOrder,* in order to find the beacon. If the beacon is not found after this time, the device shall transmit the frame following the successful application of the unslotted version of the CSMA-CA algorithm, as described in 5.1.1.4. Once the beacon has been found, either after a search or due to its being tracked, the frame shall be transmitted in the appropriate portion of the superframe. Transmissions in the CAP shall follow a successful application of the slotted version of the CSMA-CA algorithm, as described in 5.1.1.4, and transmissions in a GTS shall not use CSMA-CA.

If the frame is to be transmitted on a nonbeacon-enabled PAN, the frame shall be transmitted following the successful application of the unslotted version of the CSMA-CA algorithm, as described in 5.1.1.4.

For either a beacon-enabled PAN or a nonbeacon-enabled PAN, if the transmission is direct and originates due to a primitive issued by the next higher layer and the CSMA-CA algorithm fails, the next higher layer shall be notified. If the transmission is indirect and the CSMA-CA algorithm fails, the frame shall remain in the transaction queue until it is requested again and successfully transmitted or until the transaction expires.

The device shall process the frame using the outgoing frame security procedure described in 7.2.1.

If the status from the outgoing frame security procedure is not SUCCESS, the MLME shall issue the corresponding confirm or MLME-COMM-STATUS.indication primitive with the status parameter set to the status from the outgoing frame security procedure, indicating the error, and shall not transmit the frame.

If the status from the outgoing frame security procedure is SUCCESS, the MAC sublayer shall transmit the frame.

5.1.6.2 Reception and rejection

Each device may choose whether the MAC sublayer is to enable its receiver during idle periods. During these idle periods, the MAC sublayer shall still service transceiver task requests from the next higher layer. A transceiver task shall be defined as a transmission request with acknowledgment reception, if required, or a reception request. On completion of each transceiver task, the MAC sublayer shall request that the PHY enables or disables its receiver, depending on the values of *macBeaconOrder* and *macRxOnWhenIdle.* If *macBeaconOrder* is less than 15, the value of *macRxOnWhenIdle* shall be considered relevant only during idle periods of the CAP of the incoming superframe. If *macBeaconOrder* is equal to 15, the value of *macRxOnWhenIdle* shall be considered relevant at all times.

Due to the nature of radio communications, a device with its receiver enabled will be able to receive and decode transmissions from all devices complying with this standard that are currently operating on the same channel and are in its radio communications range, along with interference from other sources. The MAC sublayer shall, therefore, be able to filter incoming frames and present only the frames that are of interest to the upper layers.

For the first level of filtering, the MAC sublayer shall discard all received frames that do not contain a correct value in their FCS field in the MFR, as described in 5.2.1.9. The FCS field shall be verified on reception by recalculating the purported FCS over the MHR and MAC payload of the received frame and by subsequently comparing this value with the received FCS field. The FCS field of the received frame shall be considered to be correct if these values are the same and incorrect otherwise.

The second level of filtering shall be dependent on whether the MAC sublayer is currently operating in promiscuous mode. In promiscuous mode, the MAC sublayer shall pass all frames received after the first filter directly to the upper layers without applying any more filtering or processing. The MAC sublayer shall be in promiscuous mode if *macPromiscuousMode* is set to TRUE.

If the MAC sublayer is not in promiscuous mode (i.e., *macPromiscuousMode* is set to FALSE), it shall accept only frames that satisfy all of the following third-level filtering requirements:

— The Frame Type field shall not contain a reserved frame type.
— The Frame Version field shall not contain a reserved value.
— If a destination PAN identifier is included in the frame, it shall match *macPANId* or shall be the broadcast PAN identifier.
— If a short destination address is included in the frame, it shall match either *macShortAddress* or the broadcast address. Otherwise, if an extended destination address is included in the frame, it shall match *macExtendedAddress*.
— If the frame type indicates that the frame is a beacon frame, the source PAN identifier shall match *macPANId* unless *macPANId* is equal to the broadcast PAN identifier, in which case the beacon frame shall be accepted regardless of the source PAN identifier.
— If only source addressing fields are included in a data or MAC command frame, the frame shall be accepted only if the device is the PAN coordinator and the source PAN identifier matches *macPANId*.

If any of the third-level filtering requirements are not satisfied, the MAC sublayer shall discard the incoming frame without processing it further. If all of the third-level filtering requirements are satisfied, the frame shall be considered valid and processed further. For valid frames that are not broadcast, if the Frame Type field indicates a data or MAC command frame and the AR field is set to request an acknowledgment, the MAC sublayer shall send an acknowledgment frame. Prior to the transmission of the acknowledgment frame, the sequence number included in the received data or MAC command frame shall be copied into the Sequence Number field of the acknowledgment frame. This step will allow the transaction originator to know that it has received the appropriate acknowledgment frame.

If the PAN ID Compression field is set to one and both destination and source addressing information is included in the frame, the MAC sublayer shall assume that the omitted Source PAN Identifier field is identical to the Destination PAN Identifier field.

The device shall process the frame using the incoming frame security procedure described in 7.2.3.

If the status from the incoming frame security procedure is not SUCCESS, the MLME shall issue the corresponding confirm or MLME-COMM-STATUS.indication primitive with the status parameter set to the

status from the incoming frame security procedure, indicating the error, and with the security-related parameters set to the corresponding parameters returned by the unsecuring process.

If the valid frame is a data frame, the MAC sublayer shall pass the frame to the next higher layer. This is achieved by issuing the MCPS-DATA.indication primitive containing the frame information. The security-related parameters of the MCPS-DATA.indication primitive shall be set to the corresponding parameters returned by the unsecuring process.

If the valid frame is a MAC command or beacon frame, it shall be processed by the MAC sublayer accordingly, and a corresponding confirm or indication primitive may be sent to the next higher layer. The security-related parameters of the corresponding confirm or indication primitive shall be set to the corresponding parameters returned by the unsecuring process.

5.1.6.3 Extracting pending data from a coordinator

A device on a beacon-enabled PAN can determine whether any frames are pending for it by examining the contents of the received beacon frame, as described in 5.1.4.1. If the address of the device is contained in the Address List field of the beacon frame and *macAutoRequest* is TRUE, the MLME of the device shall send a data request command, as described in 5.3.4, to the coordinator during the CAP with the AR field set to request an acknowledgment; the only exception to this is if the beacon frame is received while performing an active or passive scan, as described in 5.1.2.1. There are two other cases for which the MLME shall send a data request command to the coordinator. The first case is when the MLME receives the MLME-POLL.request primitive. In the second case, a device may send a data request command *macResponseWaitTime* after the acknowledgment to a request command frame, such as during the association procedure. If the data request is intended for the PAN coordinator, the destination address information may be omitted.

If the data request command originated from an MLME-POLL.request primitive, the MLME shall perform the security process on the data request command based on the SecurityLevel, KeyIdMode, KeySource, and KeyIndex parameters of the MLME-POLL.request primitive, according to 7.2.1. Otherwise, the MLME shall perform the security process on the data request command based on the *macAutoRequestSecurityLevel*, *macAutoRequestKeyIdMode*, *macAutoRequestKeySource,* and *macAutoRequestKeyIndex* PIB attributes, according to 7.2.1.

On successfully receiving a data request command, the coordinator shall send an acknowledgment frame, thus confirming its receipt. If the coordinator has enough time to determine whether the device has a frame pending before sending the acknowledgment frame, as described in 5.1.6.4.2, it shall set the Frame Pending field of the acknowledgment frame accordingly to indicate whether a frame is actually pending for the device. If this is not possible, the coordinator shall set the Frame Pending field of the acknowledgment frame to one.

On receipt of the acknowledgment frame with the Frame Pending field set to zero, the device shall conclude that there are no data pending at the coordinator.

On receipt of the acknowledgment frame with the Frame Pending field set to one, a device shall enable its receiver for at most *macMaxFrameTotalWaitTime* to receive the corresponding data frame from the coordinator. If there is an actual data frame pending within the coordinator for the requesting device, the coordinator shall send the frame to the device using one of the mechanisms described in this subclause. If there is no data frame pending for the requesting device, the coordinator shall send a data frame without requesting acknowledgment to the device containing a zero length payload, indicating that no data are present, using one of the mechanisms described in this subclause.

The data frame following the acknowledgment of the data request command shall be transmitted using one of the following mechanisms:

— Without using CSMA-CA, if the MAC sublayer can commence transmission of the data frame between *macSIFSPeriod* and (*macSIFSPeriod* + *aUnitBackoffPeriod*), on a backoff period boundary, and there is time remaining in the CAP for the message, appropriate IFS, and acknowledgment. If a requested acknowledgment frame is not received following this data frame, the process shall begin anew following the receipt of a new data request command.

— Using CSMA-CA, otherwise.

If the requesting device does not receive a data frame from the coordinator within *macMaxFrameTotalWaitTime* or if the requesting device receives a data frame from the coordinator with a zero length payload, it shall conclude that there are no data pending at the coordinator. If the requesting device does receive a data frame from the coordinator, it shall send an acknowledgment frame, if requested, thus confirming receipt.

If the Frame Pending field of the data frame received from the coordinator is set to one, the device still has more data pending with the coordinator. In this case it may extract the data by sending a new data request command to the coordinator.

In Figure 23 a poll request is issued to the MLME, which then sends a data request command to the coordinator. The corresponding acknowledgment has the Frame Pending (FP) field set to zero and the MLME issues the poll request confirmation immediately.

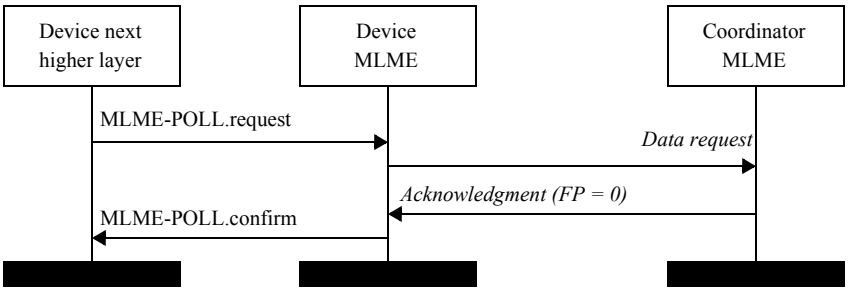


Figure 23—Message sequence chart for requesting data from the coordinator when the coordinator does not have data pending

In Figure 24 a poll request is issued to the MLME, which then sends a data request command to the coordinator. The corresponding acknowledgment has the Frame Pending field set to one and the MLME enables the receiver in anticipation of the data frame from the coordinator. On receipt of this data frame, the MLME issues a poll request confirmation followed by a data indication containing the data of the received frame.

5.1.6.4 Use of acknowledgments and retransmissions

A data or MAC command frame shall be sent with the AR field set appropriately for the frame. A beacon or acknowledgment frame shall always be sent with the AR field set to indicate no acknowledgment requested. Similarly, any frame that is broadcast shall be sent with its AR field set to indicate no acknowledgment requested.

5.1.6.4.1 No acknowledgment

A frame transmitted with its AR field set to indicate no acknowledgment requested, as defined in 5.2.1.1.4, shall not be acknowledged by its intended recipient. The originating device shall assume that the transmission of the frame was successful.

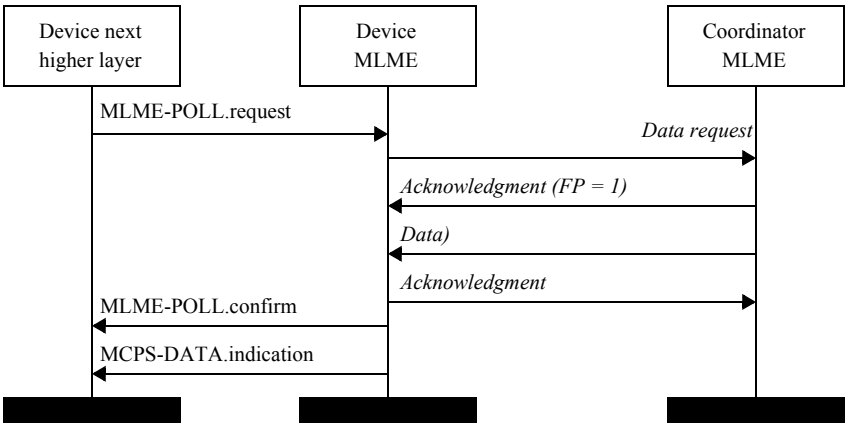


Figure 24—Message sequence chart for requesting data from the coordinator when the coordinator has data pending

The message sequence chart in Figure 25 shows the scenario for transmitting a single frame of data from an originator to a recipient without requiring an acknowledgment.

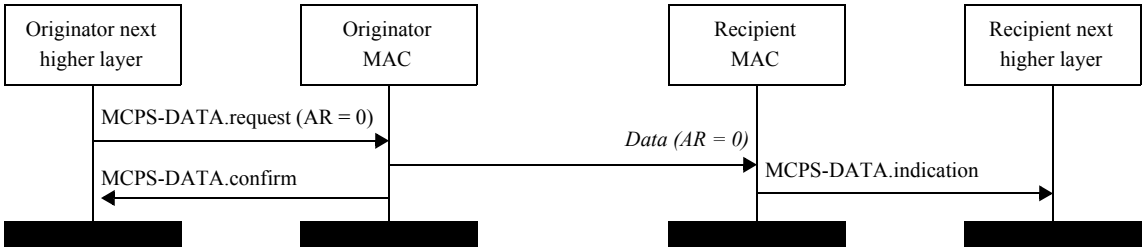


Figure 25—Successful data transmission without an acknowledgment

5.1.6.4.2 Acknowledgment

A frame transmitted with the AR field set to request an acknowledgment, as defined in 5.2.1.1.4, shall be acknowledged by the recipient. If the intended recipient correctly receives the frame, it shall generate and send an acknowledgment frame containing the same DSN from the data or MAC command frame that is being acknowledged.

The transmission of an acknowledgment frame in a nonbeacon-enabled PAN or in the CFP shall commence *macSIFSPeriod* after the reception of the last symbol of the data or MAC command frame. The transmission of an acknowledgment frame in the CAP shall commence either *macSIFSPeriod* after the reception of the last symbol of the data or MAC command frame or at a backoff period boundary. In the latter case, the transmission of an acknowledgment frame shall commence between *macSIFSPeriod* and (*macSIFSPeriod* + *aUnitBackoffPeriod*) after the reception of the last symbol of the data or MAC command frame.

The message sequence chart in Figure 26 shows the scenario for transmitting a single frame of data from an originator to a recipient with an acknowledgment requested.

5.1.6.4.3 Retransmissions

A device that sends a frame with the AR field set to indicate no acknowledgment requested may assume that the transmission was successfully received and shall not perform the retransmission procedure.

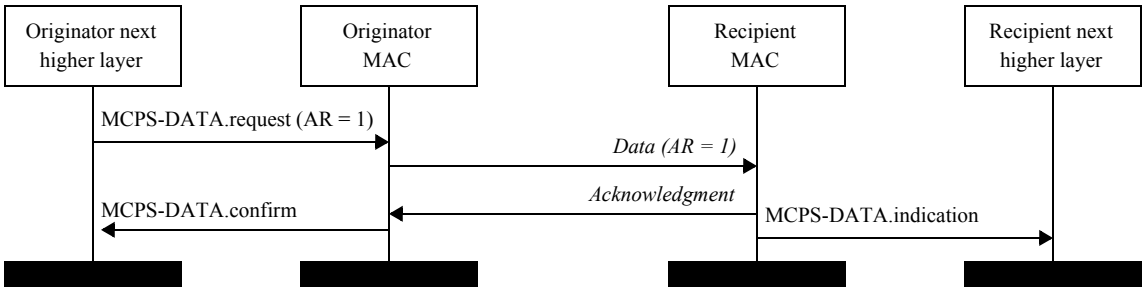


Figure 26—Successful data transmission with an acknowledgment

A device that sends a data or MAC command frame with its AR field set to acknowledgment requested shall wait for at most *macAckWaitDuration* for the corresponding acknowledgment frame to be received. If an acknowledgment frame is received within *macAckWaitDuration* and contains the same DSN as the original transmission, the transmission is considered successful, and no further action regarding retransmission shall be taken by the device. If an acknowledgment is not received within *macAckWaitDuration* or an acknowledgment is received containing a DSN that was not the same as the original transmission, the device shall conclude that the single transmission attempt has failed.

If a single transmission attempt has failed and the transmission was indirect, the coordinator shall not retransmit the data or MAC command frame. Instead, the frame shall remain in the transaction queue of the coordinator and can only be extracted following the reception of a new data request command. If a new data request command is received, the originating device shall transmit the frame using the same DSN as was used in the original transmission.

If a single transmission attempt has failed and the transmission was direct, the device shall repeat the process of transmitting the data or MAC command frame and waiting for the acknowledgment, up to a maximum of *macMaxFrameRetries* times. The retransmitted frame shall contain the same DSN as was used in the original transmission. Each retransmission shall only be attempted if it can be completed within the same portion of the superframe, i.e., the CAP or a GTS in which the original transmission was attempted. If this timing is not possible, the retransmission shall be deferred until the same portion in the next superframe. If an acknowledgment is still not received after *macMaxFrameRetries* retransmissions, the MAC sublayer shall assume the transmission has failed and notify the next higher layer of the failure.

When a frame with the Security Enabled field set to one is retransmitted, the frame shall be retransmitted without changes and without passing through the outgoing frame security procedure, as defined in 7.2.1.

5.1.6.5 Promiscuous mode

A device may activate promiscuous mode by setting *macPromiscuousMode*. If the MLME is requested to set *macPromiscuousMode* to TRUE, the MLME shall then request that the PHY enable its receiver.

When in promiscuous mode, the MAC sublayer shall process received frames according to 5.1.6.2 and pass all frames correctly received to the next higher layer using the MCPS-DATA.indication primitive. The source and destination addressing mode parameters shall each be set to 0x00, the MSDU parameter shall contain the MHR concatenated with the MAC payload, as illustrated in Figure 35, and the msduLength parameter shall contain the total number of octets in the MHR concatenated with the MAC payload. The mpduLinkQuality parameter shall be valid.

If the MLME is requested to set *macPromiscuousMode* to FALSE, the MLME shall request that the PHY set its receiver to the state specified by *macRxOnWhenIdle*.

5.1.6.6 Transmission scenarios

Due to the imperfect nature of the radio medium, a transmitted frame does not always reach its intended destination. There are three different transmission scenarios:

— *Successful data transmission.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a timer that will expire after *macAckWaitDuration*. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator, and passes the data frame to the next higher layer. The originator MAC sublayer receives the acknowledgment from the recipient before its timer expires and then disables and resets the timer. The data transfer is now complete, and the originator MAC sublayer issues a success confirmation to the next higher layer. This sequence of messages for successful data transmission is illustrated in Figure 27.

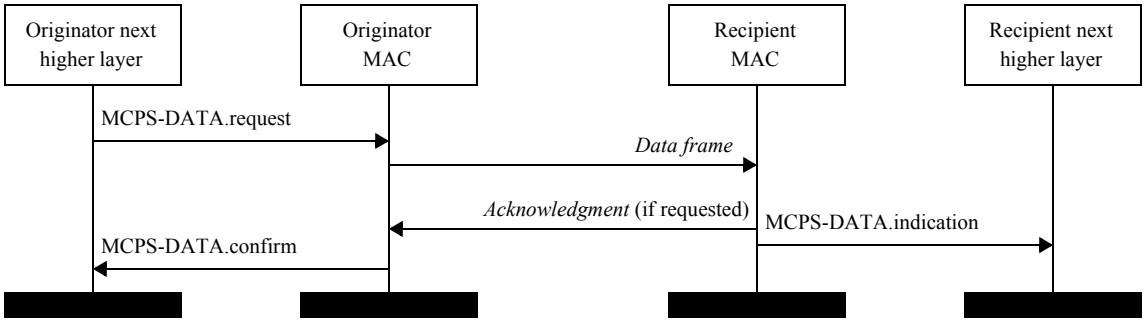


Figure 27—Successful data transmission sequence

— *Lost data frame.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a timer that will expire after *macAckWaitDuration*. The recipient MAC sublayer does not receive the data frame and so does not respond with an acknowledgment. The timer of the originator MAC sublayer expires before an acknowledgment is received; therefore, the data transfer has failed. If the transmission was direct, the originator retransmits the data, and this entire sequence may be repeated up to a maximum of *macMaxFrameRetries* times; if a data transfer attempt fails a total of (1 + *macMaxFrameRetries*) times, the originator MAC sublayer will issue a failure confirmation to the next higher layer. If the transmission was indirect, the data frame will remain in the transaction queue until either another request for the data is received and correctly acknowledged or until *macTransactionPersistenceTime* is reached. If *macTransactionPersistenceTime* is reached, the transaction information will be discarded, and the MAC sublayer will issue a failure confirmation to the next higher layer. The sequence of messages for a lost data frame is illustrated in Figure 28.

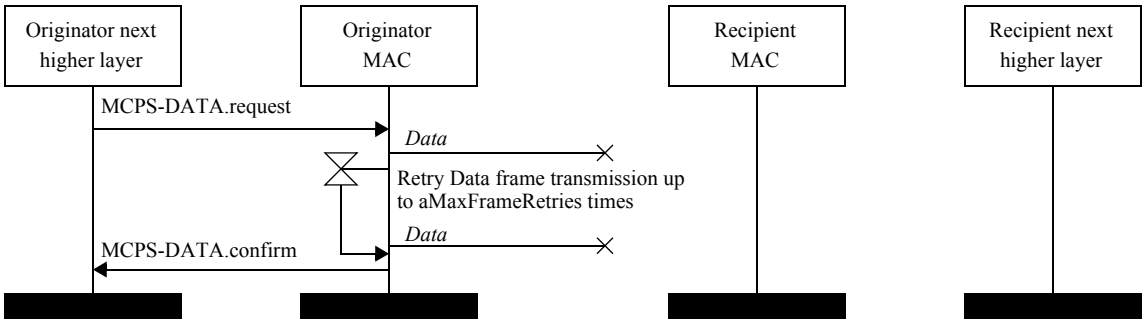


Figure 28—Lost data frame message sequence

— *Lost acknowledgment frame.* The originator MAC sublayer transmits the data frame to the recipient via the PHY data service. In waiting for an acknowledgment, the originator MAC sublayer starts a

timer that will expire after *macAckWaitDuration*. The recipient MAC sublayer receives the data frame, sends an acknowledgment back to the originator, and passes the data frame to the next higher layer. The originator MAC sublayer does not receive the acknowledgment frame, and its timer expires. Therefore, the data transfer has failed. If the transmission was direct, the originator retransmits the data, and this entire sequence may be repeated up to a maximum of *macMaxFrameRetries* times. If a data transfer attempt fails a total of (1 + *macMaxFrameRetries*) times, the originator MAC sublayer will issue a failure confirmation to the next higher layer. If the transmission was indirect, the data frame will remain in the transaction queue either until another request for the data is received and correctly acknowledged or until *macTransactionPersistenceTime* is reached. If *macTransactionPersistenceTime* is reached, the transaction information will be discarded, and the MAC sublayer will issue a failure confirmation to the next higher layer. The message sequence for a lost acknowledgment frame is illustrated in Figure 29.

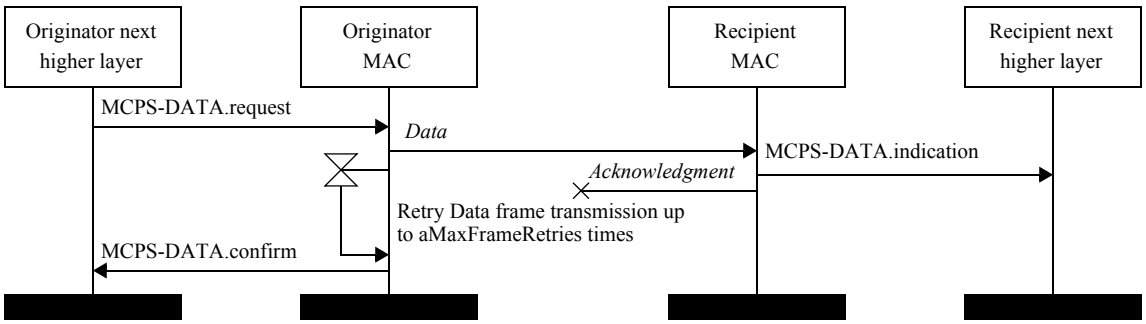


Figure 29—Lost acknowledgment message sequence

5.1.7 GTS allocation and management

A GTS allows a device to operate on the channel within a portion of the superframe that is dedicated (on the PAN) exclusively to that device. The use of GTSs is optional.

A GTS shall be allocated only by the PAN coordinator, and it shall be used only for communications between the PAN coordinator and a device associated with the PAN through the PAN coordinator. A single GTS may extend over one or more superframe slots. The PAN coordinator may allocate up to seven GTSs at the same time, provided there is sufficient capacity in the superframe.

A GTS shall be allocated before use, with the PAN coordinator deciding whether to allocate a GTS based on the requirements of the GTS request and the current available capacity in the superframe. GTSs shall be allocated on a first-come-first-served basis, and all GTSs shall be placed contiguously at the end of the superframe and after the CAP. Each GTS shall be deallocated when the GTS is no longer required, and a GTS can be deallocated at any time at the discretion of the PAN coordinator or by the device that originally requested the GTS. A device that has been allocated a GTS may also operate in the CAP.

A data frame transmitted in an allocated GTS shall use only short addressing.

The management of GTSs shall be undertaken by the PAN coordinator only. To facilitate GTS management, the PAN coordinator shall be able to store all the information necessary to manage seven GTSs. For each GTS, the PAN coordinator shall be able to store its starting slot, length, direction, and associated device address.

The GTS direction, which is relative to the data flow from the device that owns the GTS, is specified as either transmit or receive. The device address and direction shall, therefore, uniquely identify each GTS. Each device may request one transmit GTS and/or one receive GTS. For each allocated GTS, the device shall be able to store its starting slot, length, and direction. If a device has been allocated a receive GTS, it

shall enable its receiver for the entirety of the GTS. In the same way, the PAN coordinator shall enable its receiver for the entirety of the GTS if a device has been allocated a transmit GTS. If a data frame is received during a receive GTS and an acknowledgment is requested, the device shall transmit the acknowledgment frame as usual. Similarly, a device shall be able to receive an acknowledgment frame during a transmit GTS.

A device shall attempt to allocate and use a GTS only if it is currently tracking the beacons. The MLME is instructed to track beacons by issuing the MLME-SYNC.request primitive with the TrackBeacon parameter set to TRUE. If a device loses synchronization with the PAN coordinator, all its GTS allocations shall be lost.

5.1.7.1 CAP maintenance

The PAN coordinator shall preserve the minimum CAP length of *aMinCAPLength* and take preventative action if the minimum CAP is not satisfied. However, an exception shall be allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance. If preventative action becomes necessary, the action chosen is left up to the implementation but may include one or more of the following:

— Limiting the number of pending addresses included in the beacon.
— Not including a payload field in the beacon frame.
— Deallocating one or more of the GTSs.

5.1.7.2 GTS allocation

A device is instructed to request the allocation of a new GTS through the MLME-GTS.request primitive, as described in 6.2.6.1, with GTS characteristics set according to the requirements of the intended application.

To request the allocation of a new GTS, the MLME shall send the GTS request command, as described in 5.3.9, to the PAN coordinator. The Characteristics Type field of the GTS Characteristics field of the request shall be set to one (GTS allocation), and the length and direction fields shall be set according to the desired characteristics of the required GTS.

On receipt of a GTS request command indicating a GTS allocation request, the PAN coordinator shall first check if there is available capacity in the current superframe, based on the remaining length of the CAP and the desired length of the requested GTS. The superframe shall have available capacity if the maximum number of GTSs has not been reached and allocating a GTS of the desired length would not reduce the length of the CAP to less than *aMinCAPLength*. GTSs shall be allocated on a first-come-first-served basis by the PAN coordinator provided there is sufficient bandwidth available. The PAN coordinator shall make this decision within *aGTSDescPersistenceTime* superframes.

On receipt of the acknowledgment to the GTS request command, the device shall continue to track beacons and wait for at most *aGTSDescPersistenceTime* superframes. If no GTS descriptor for the device appears in the beacon within this time, the MLME of the device shall notify the next higher layer of the failure. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive, as described in 6.2.6.2, with a status of NO_DATA.

When the PAN coordinator determines whether capacity is available for the requested GTS, it shall generate a GTS descriptor with the requested specifications and the short address of the requesting device. If the GTS was allocated successfully, the PAN coordinator shall set the start slot in the GTS descriptor to the superframe slot at which the GTS begins and the length in the GTS descriptor to the length of the GTS. In addition, the PAN coordinator shall notify the next higher layer of the new GTS. This notification is achieved when the MLME of the PAN coordinator issues the MLME-GTS.indication primitive, as described in 6.2.6.3, with the characteristics of the allocated GTS. If there was not sufficient capacity to allocate the

requested GTS, the start slot shall be set to zero and the length to the largest GTS length that can currently be supported. The PAN coordinator shall then include this GTS descriptor in its beacon and update the GTS Specification field of the beacon frame accordingly. The PAN coordinator shall also update the Final CAP Slot field of the Superframe Specification field of the beacon frame, indicating the final superframe slot utilized by the decreased CAP. The GTS descriptor shall remain in the beacon frame for *aGTSDescPersistenceTime* superframes, after which it shall be removed automatically. The PAN coordinator shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length due to the inclusion of the GTS descriptor.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress*, the device shall process the descriptor. The MLME of the device shall then notify the next higher layer of whether the GTS allocation request was successful. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive with a status of SUCCESS (if the start slot in the GTS descriptor was greater than zero) or DENIED (if the start slot was equal to zero or if the length did not match the requested length).

Figure 30 illustrates the message flow for the case in which the device requests the GTS allocation.

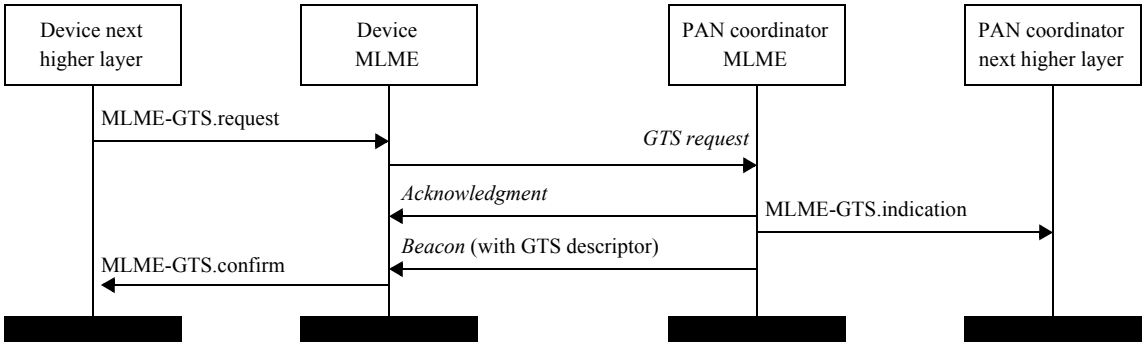


Figure 30—Message sequence chart for GTS allocation initiated by a device

5.1.7.3 GTS usage

When the MAC sublayer of a device that is not the PAN coordinator receives an MCPS-DATA.request primitive, as described in 6.3.1, with the TxOptions parameter indicating a GTS transmission, it shall determine whether it has a valid transmit GTS. If a valid GTS is found, the MAC sublayer shall transmit the data during the GTS, i.e., between its starting slot and its starting slot plus its length. At this time, the MAC sublayer shall transmit the MPDU immediately without using CSMA-CA, provided the requested transaction can be completed before the end of the GTS. If the requested transaction cannot be completed before the end of the current GTS, the MAC sublayer shall defer the transmission until the specified GTS in the next superframe.

If the device has any receive GTSs, the MAC sublayer of the device shall ensure that the receiver is enabled at a time prior to the start of the GTS and for the duration of the GTS, as indicated by its starting slot and its length.

When the MAC sublayer of the PAN coordinator receives an MCPS-DATA.request primitive with the TxOptions parameter indicating a GTS transmission, it shall determine whether it has a valid receive GTS corresponding to the device with the requested destination address. If a valid GTS is found, the PAN coordinator shall defer the transmission until the start of the receive GTS. In this case, the address of the device with the message requiring a GTS transmission shall not be added to the list of pending addresses in the beacon frame, as described in 5.1.5. At the start of the receive GTS, the MAC sublayer shall transmit the data without using CSMA-CA, provided the requested transaction can be completed before the end of the

GTS. If the requested transaction cannot be completed before the end of the current GTS, the MAC sublayer shall defer the transmission until the specified GTS in the next superframe.

For all allocated transmit GTSs (relative to the device), the MAC sublayer of the PAN coordinator shall ensure that its receiver is enabled at a time prior to the start and for the duration of each GTS.

Before commencing transmission in a GTS, each device shall ensure that the data transmission, the acknowledgment, if requested, and the IFS, suitable to the size of the data frame, can be completed before the end of the GTS.

If a device misses the beacon at the beginning of a superframe, it shall not use its GTSs until it receives a subsequent beacon correctly. If a loss of synchronization occurs due to the loss of the beacon, the device shall consider all of its GTSs deallocated.

5.1.7.4 GTS deallocation

A device is instructed to request the deallocation of an existing GTS through the MLME-GTS.request primitive, as described in 6.2.6.1, using the characteristics of the GTS it wishes to deallocate. From this point onward, the GTS to be deallocated shall not be used by the device, and its stored characteristics shall be reset.

To request the deallocation of an existing GTS, the MLME shall send the GTS request command, as described in 5.3.9, to the PAN coordinator. The Characteristics Type field of the GTS Characteristics field of the request shall be set to zero (i.e., GTS deallocation), and the length and direction fields shall be set according to the characteristics of the GTS to deallocate. On receipt of the acknowledgment to the GTS request command, the MLME shall notify the next higher layer of the deallocation. This notification is achieved when the MLME issues the MLME-GTS.confirm primitive, as described in 6.2.6.2, with a status of SUCCESS and a GTSCharacteristics parameter with its Characteristics Type field set to zero. If the GTS request command is not received correctly by the PAN coordinator, it shall determine that the device has stopped using its GTS by the procedure described in 5.1.7.6.

On receipt of a GTS request command with the Characteristics Type field of the GTS Characteristics field set to zero (GTS deallocation), the PAN coordinator shall attempt to deallocate the GTS. If the GTS characteristics contained in the GTS request command do not match the characteristics of a known GTS, the PAN coordinator shall ignore the request. If the GTS characteristics contained in the GTS request command match the characteristics of a known GTS, the MLME of the PAN coordinator shall deallocate the specified GTS and notify the next higher layer of the change. This notification is achieved when the MLME issues the MLME-GTS.indication primitive, as described in 6.2.6.3, with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a Characteristics Type field set to zero. The PAN coordinator shall also update the Final CAP Slot field of the Superframe Specification field of the beacon frame, indicating the final superframe slot utilized by the increased CAP. It shall not add a descriptor to the beacon frame to describe the deallocation.

GTS deallocation may be initiated by the PAN coordinator due to a deallocation request from the next higher layer, the expiration of the GTS, as described in 5.1.7.6, or maintenance required to maintain the minimum CAP length, *aMinCAPLength*, as described in 5.1.7.1.

The next higher layer of the PAN coordinator initiates a GTS deallocation using an MLME-GTS.request primitive with the GTS Characteristics field of the request set to indicate a GTS deallocation and the length and direction fields set according to the characteristics of the GTS to deallocate. The MLME shall then respond with an MLME-GTS.confirm primitive with a status of SUCCESS and the GTSCharacteristics parameter with a Characteristics Type field set to zero.

When a GTS deallocation is initiated by the PAN coordinator either due to the GTS expiring or due to CAP maintenance, the MLME shall notify the next higher layer of the change using the MLME-GTS.indication primitive with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a Characteristics Type field set to zero.

In the case of any deallocation initiated by PAN coordinator, the PAN coordinator shall deallocate the GTS and add a GTS descriptor into its beacon frame corresponding to the deallocated GTS, but with its starting slot set to zero. The descriptor shall remain in the beacon frame for *aGTSDescPersistenceTime* superframes. The PAN coordinator shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length due to the inclusion of the GTS descriptor.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress* and a start slot equal to zero, the device shall immediately stop using the GTS. The MLME of the device shall then notify the next higher layer of the deallocation using the MLME-GTS.indication primitive with a GTSCharacteristics parameter containing the characteristics of the deallocated GTS and a Characteristics Type field set to zero.

Figure 31 depicts the message flow for the cases in which a GTS deallocation is initiated by a device.

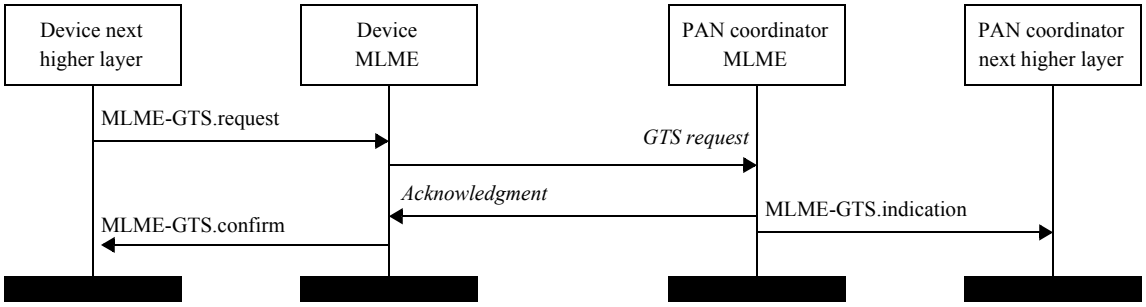


Figure 31—Message sequence chart for GTS deallocation initiated by a device

Figure 32 depicts the message flow for the cases in which a GTS deallocation is initiated by the PAN coordinator.

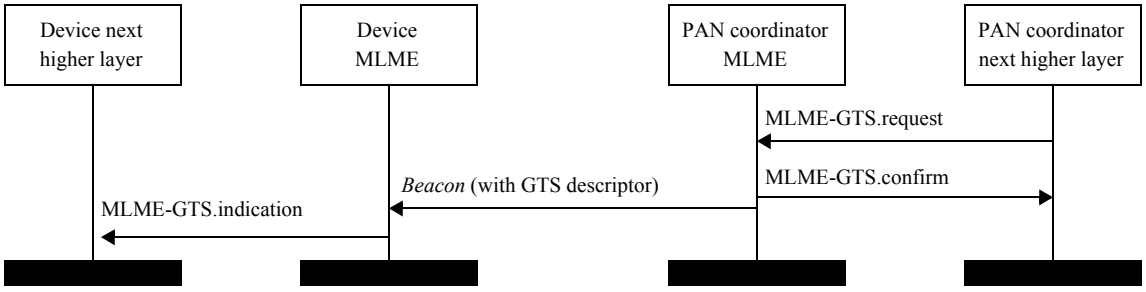


Figure 32—Message sequence chart for GTS deallocation initiated by the PAN coordinator

5.1.7.5 GTS reallocation

The deallocation of a GTS may result in the superframe becoming fragmented. For example, Figure 33 shows three stages of a superframe with allocated GTSs. In stage 1, three GTSs are allocated starting at slots 14, 10, and 8, respectively. If GTS 2 is now deallocated (stage 2), there will be a gap in the superframe during which nothing can happen. To solve this, GTS 3 will have to be shifted to fill the gap, thus increasing the size of the CAP (stage 3).

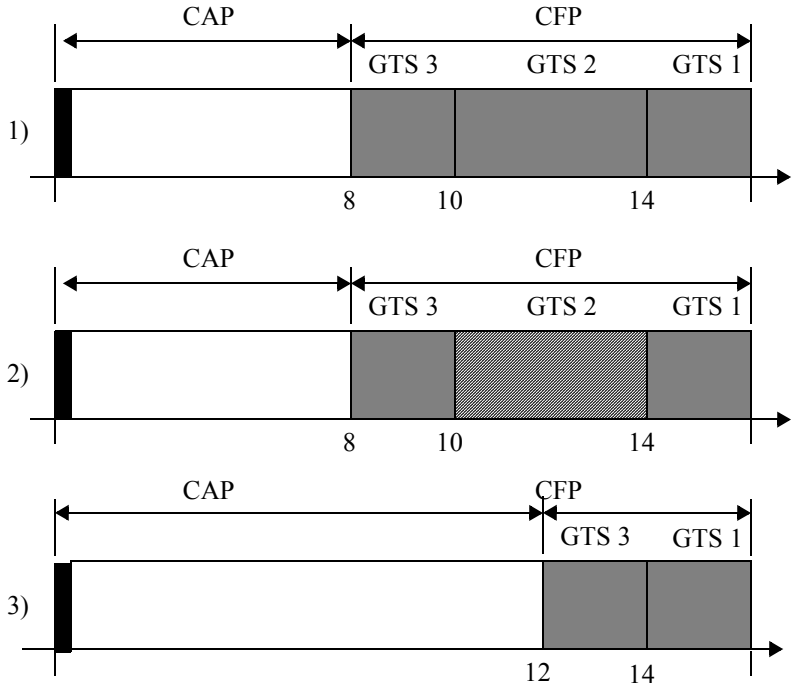


Figure 33—CFP defragmentation on GTS deallocations

The PAN coordinator shall ensure that any gaps occurring in the CFP, appearing due to the deallocation of a GTS, are removed to maximize the length of the CAP.

When a GTS is deallocated by the PAN coordinator, it shall add a GTS descriptor into its beacon frame indicating that the GTS has been deallocated. If the deallocation is initiated by a device, the PAN coordinator shall not add a GTS descriptor into its beacon frame to indicate the deallocation. For each device with an allocated GTS having a starting slot lower than the GTS being deallocated, the PAN coordinator shall update the GTS with the new starting slot and add a GTS descriptor to its beacon corresponding to this adjusted GTS. The new starting slot is computed so that no space is left between this GTS and either the end of the CFP, if the GTS appears at the end of the CFP, or the start of the next GTS in the CFP.

In situations where multiple reallocations occur at the same time, the PAN coordinator may choose to perform the reallocation in stages. The PAN coordinator shall keep each GTS descriptor in its beacon for *aGTSDescPersistenceTime* superframes.

On receipt of a beacon frame containing a GTS descriptor corresponding to *macShortAddress* and a direction and length corresponding to one of its GTSs, the device shall adjust the starting slot of the GTS corresponding to the GTS descriptor and start using it immediately.

In cases where it is necessary for the PAN coordinator to include a GTS descriptor in its beacon, it shall be allowed to reduce its CAP below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length. After *aGTSDescPersistenceTime* superframes, the PAN coordinator shall remove the GTS descriptor from the beacon.

5.1.7.6 GTS expiration

The MLME of the PAN coordinator shall attempt to detect when a device has stopped using a GTS using the following rules:

— For a transmit GTS, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if a data frame is not received from the device in the GTS at least every $2 \times n$ superframes, where n is defined below.

— For receive GTSs, the MLME of the PAN coordinator shall assume that a device is no longer using its GTS if an acknowledgment frame is not received from the device at least every $2 \times n$ superframes, where n is defined below. If the data frames sent in the GTS do not require acknowledgment frames, the MLME of the PAN coordinator will not be able to detect whether a device is using its receive GTS. However, the PAN coordinator is capable of deallocating the GTS at any time.

The value of n is defined as follows:

$$n = 2^{(8-macBeaconOrder)} \qquad 0 \le macBeaconOrder \le 8$$

$$n = 1 \qquad 9 \le macBeaconOrder \le 14$$

5.1.8 Ranging

Ranging is an optional feature. The fundamental measurements for ranging are achieved using a data-acknowledgment frame sequence. Ranging capabilities are enabled in a ranging-capable device (RDEV) with the MCPS-DATA.request primitive. Whenever ranging is enabled in an RDEV, the RDEV delivers timestamp reports to the next higher layer as a result of events at the device antenna.

5.1.8.1 Set-up activities before a ranging exchange

The mandatory part of ranging is limited to the generation of timestamp reports during the period that ranging is enabled in an RDEV. It is possible that an RDEV will consume more power when ranging is enabled; therefore, a natural default for an application would be to have ranging disabled. Prior to a two-way ranging exchange, both RDEVs involved in the exchange shall already have ranging enabled. Furthermore, if the optional DPS capability is to be used, there shall have been some sort of coordination of preambles prior to the two-way ranging exchange. How this coordination and enabling actually is accomplished is beyond the scope of this standard. It may be perfectly acceptable to accomplish the coordination and enabling with a clock and a look-up table that says what a device should do at a particular time. Because coordination generally involves communication and because the PHYs are designed to achieve communication, it is natural to suggest that the PHY be used for coordination.

5.1.8.2 Finish-up activities after a ranging exchange

At the end of a two-way exchange, each device is in possession of a timestamp report. To accomplish anything useful, both of those timestamp reports shall eventually come to be at the same node where computations are performed. How this movement of timestamp reports is accomplished is beyond the scope of this standard. Timestamp reports are just data. Because movement of data involves communication and because the PHYs are designed to achieve communication, it is natural to suggest that the PHY be used for the final consolidation of timestamp reports.

The application is responsible for enabling the ranging mode in the RDEV before a ranging exchange. After a ranging exchange, the application is again responsible for disabling the ranging mode in the RDEV. If the application fails to disable the ranging mode in the RDEV, there will be no algorithmic harm. Ranging mode is fully compatible with other uses of the RDEV, and the only result of leaving ranging enabled when it is not really being used is that the RDEV will generate useless timestamp reports while potentially consuming significant power.

5.1.8.3 Managing DPS

Figure 34 shows a suggested message sequence for two-way ranging. The messages represented in the two top dotted boxes are simply suggestions showing how the communications capability of the RDEV can be used to accomplish the ranging setup activities. The messages in the bottom dotted box are suggestions showing how the communications capability of the RDEV can be used to accomplish the ranging finish-up activities.

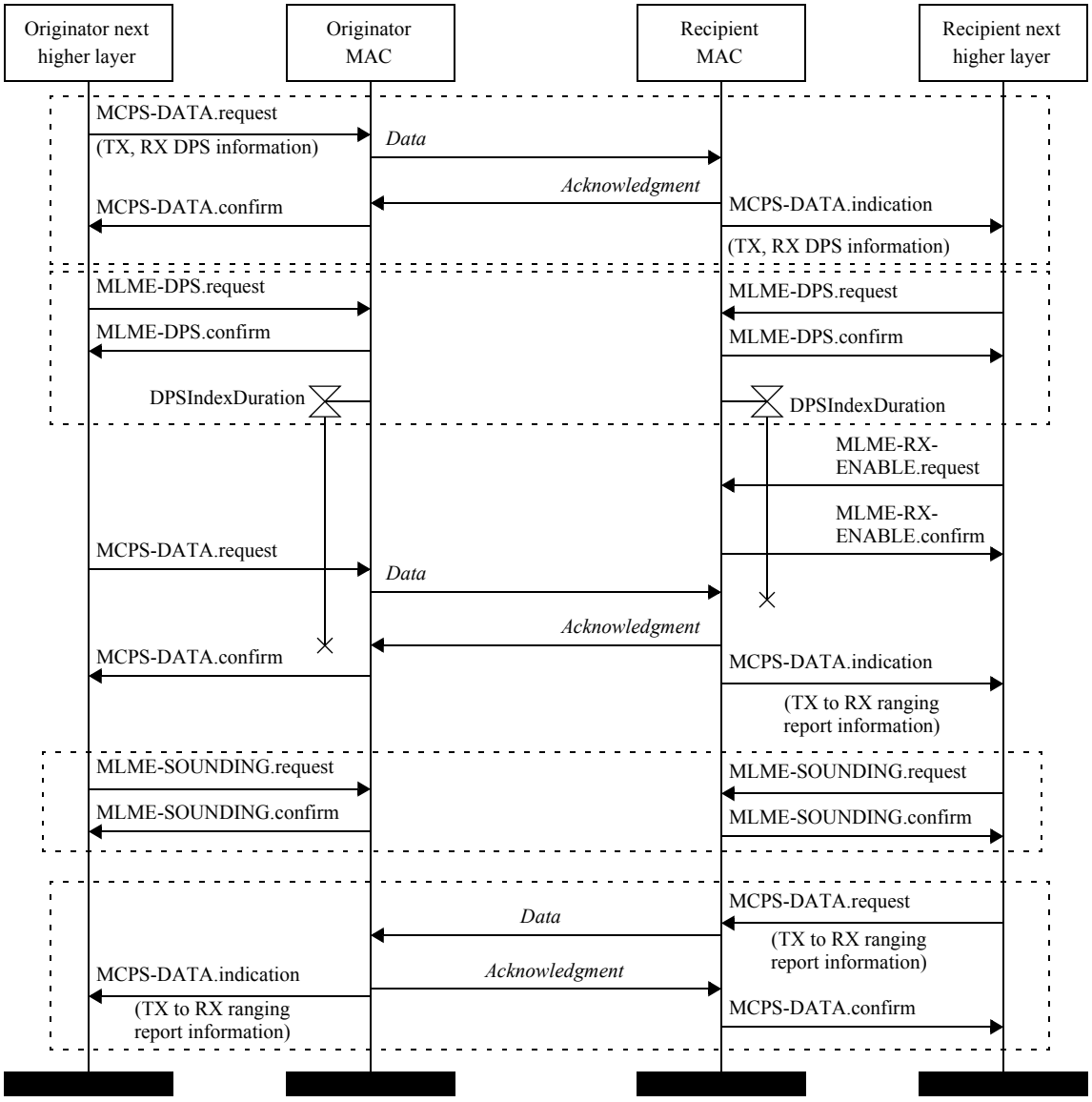


Figure 34—A message sequence for ranging

The top dotted box in Figure 34 illustrates the use of a data exchange to effect the coordination of the preambles to be used for a two-way ranging exchange. The coordination of preambles is needed only when using the optional DPS capability of the PHY. If optional DPS is not used, the communication sequence in the top box can be thought of as arranging for the recipient RDEV to become aware that a ranging exchange is desired and that the recipient next higher layer should enable ranging in the recipient PHY. The middle dotted box in Figure 34 illustrates the use of the MLME-DPS.request, as described in 6.2.15.1, and the

MLME-DPS.confirm, as described in 6.2.15.2. Use of these primitives is unique to the optional DPS mode of ranging.

Upon the assertion of the MLME-DPS.confirm primitives, as illustrated in Figure 34, both of the PHYs have switched from the normal length preamble symbols to long preamble symbols. This is desirable behavior intended to help hide the PHYs' transmissions from malicious nodes and protect the PHYs from transmissions by malicious nodes. A side effect of this mode is that neither PHYs can communicate with the rest of the network. To prevent the PHYs from becoming lost as a result of this optional behavior, the MAC sublayers on both sides of the link shall initiate timers after sending the frame (for the originator) or receiving the frame (for the recipient). If the timer duration is exceeded before the MAC sublayer receives the MCPS-DATA.comfirm (for the originator) or the MCPS-DATA.indication primitive (for the recipient), then the MAC sublayer shall initiate a MLME-DPS.indication to the next higher layer as described in 6.2.15.3. Not shown in Figure 34, one responsibility of the application, if the optional DPS capability is used, is to initiate the MLME-DPS.request primitive on both sides of the ranging link at the completion of the ranging exchange. Most typically, this MLME-DPS.request primitive would be part of the finish-up activities and would have both TxDPSIndex and RxDPSIndex set to zero to return the PHYs to using *phyCurrentCode* from the PIB. Also not shown in Figure 34, another responsibility of the application is to initiate a MLME-DPS.request primitive in response to an MLME-DPS.indication. Most typically, this MLME-DPS.request primitive would have both TxDPSIndex and RxDPSIndex set to zero and return the PHY to using *phyCurrentCode* from the PIB.

5.1.8.4 The ranging exchange

The essential core of the ranging exchange is shown in Figure 34 starting just below the middle dotted box. The application is responsible for initiating the MLME-RX-ENABLE.request primitive, as described in 6.2.9.1, with RangingRxControl equal to RANGING_ON. Once the RDEV has received the MLME-RX-ENABLE.request primitive with RangingRxControl equal to RANGING_ON, all future RFRAMEs received by the RDEV shall generate timestamp reports until ranging is disabled.

At the initiator, the application is responsible for initiating a MCPS-DATA.request primitive with Ranging equal to ALL_RANGING. Upon receipt of a MCPS-DATA.request primitive with Ranging equal to ALL_RANGING, RDEV shall generate timestamp reports for all RFRAMEs after the transmit frame is transmitted. The timestamp reports will continue until ranging is disabled. The Tx-to-Rx turnaround enabling the originator to receive the acknowledgment frame is necessary and is not shown in Figure 34. This turnaround is the normal turnaround that is done for any exchange expecting an acknowledgment. The turnaround happens without any action required by the originator next higher layer. Timestamp reports are generated to the next higher layer independent of the state of the acknowledge request bit in the MAC header of received RFRAMEs.

As shown in Figure 34, the first timestamp report to the originator next higher layer shall come back as elements of the MCPS-DATA.confirm. The first timestamp report to the recipient next higher layer shall come back as elements of the MCPS-DATA.indication primitive. All subsequent timestamp reports on either side of the link shall come back as elements of MCPS-DATA.indication primitives. The potential additional MCPS-DATA.indication primitives that would be due to unexpected stray RFRAMEs are not shown in Figure 34 for simplicity. The timestamp reports due to any strays shall continue until ranging is disabled. The reporting of timestamps for a stream of "strays" is the behavior that enables the RDEV to be used as an infrastructure RDEVs in one-way ranging applications.

In all of these timestamp reports, the timestamp itself shall consist of the 12 octets defined for the timestamp report in 14.7.1, 14.7.2, and 14.7.3. Use of nonzero timestamp reports is limited to RDEVs. Only devices that have *phyRanging* set to TRUE shall return a nonzero timestamp report to a next higher layer.

5.2 MAC frame formats

This subclause specifies the format of the MAC frame (MPDU).

The frames in the MAC sublayer are described as a sequence of fields in a specific order. All frame formats in this subclause are depicted in the order in which they are transmitted by the PHY, from left to right, where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to $k - 1$ (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent to the PHY in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

Unless otherwise specified in this Clause, all reserved bits shall be set to zero upon transmission and may be ignored upon receipt.

A device's extended address shall be a 64-bit universal address, as defined by the IEEE Registration Authority.[5]

5.2.1 General MAC frame format

The general MAC frame shall be formatted as illustrated in Figure 35.

<table>
<tr><th>Octets: 2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>0/5/6/10/14</th><th>variable</th><th>2</th></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="5">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Figure 35—General MAC frame format

The fields of the MHR appear in a fixed order; however, the addressing fields may not be included in all frames.

5.2.1.1 Frame Control field

The Frame Control field contains information defining the frame type, addressing fields, and other control flags. The Frame Control field shall be formatted as illustrated in Figure 36.

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figure 36—Format of the Frame Control field

[5]Interested applicants should contact the IEEE Registration Authority, http://standards.ieee.org/develop/regauth/.

5.2.1.1.1 Frame Type field

The Frame Type field shall be set as defined in Table 2.

Table 2—Values of the Frame Type field

Frame type value b_2 b_1 b_0	Description
000	Beacon
001	Data
010	Acknowledgment
011	MAC command
100–111	Reserved

5.2.1.1.2 Security Enabled field

The Security Enabled field shall be set to one if the frame is protected by the MAC sublayer and shall be set to zero otherwise. The Auxiliary Security Header field of the MHR shall be present only if the Security Enabled field is set to one.

5.2.1.1.3 Frame Pending field

The Frame Pending field shall be set to one if the device sending the frame has more data for the recipient, as described in 5.1.6.3. This field shall be set to zero otherwise.

The Frame Pending field shall be used only in beacon frames or frames transmitted either during the CAP by devices operating on a beacon-enabled PAN or at any time by devices operating on a nonbeacon-enabled PAN.

At all other times, it shall be set to zero on transmission and ignored on reception.

5.2.1.1.4 Acknowledgment Request (AR) field

The AR field specifies whether an acknowledgment is required from the recipient device on receipt of a data or MAC command frame. If this field is set to one, the recipient device shall send an acknowledgment frame only if, upon reception, the frame passes the filtering described in 5.1.6.2. If this field is set to zero, the recipient device shall not send an acknowledgment frame.

5.2.1.1.5 PAN ID Compression field

The PAN ID Compression field specifies whether the MAC frame is to be sent containing only one of the PAN identifier fields when both source and destination addresses are present. If this field is set to one and both the source and destination addresses are present, the frame shall contain only the Destination PAN Identifier field, and the Source PAN Identifier field shall be assumed equal to that of the destination. If this field is set to zero, then the PAN Identifier field shall be present if and only if the corresponding address is present.

5.2.1.1.6 Destination Addressing Mode field

The Destination Addressing Mode field shall be set to one of the values listed in Table 3.

If this field is equal to zero and the Frame Type field does not specify that this frame is an acknowledgment or beacon frame, the Source Addressing Mode field shall be nonzero, implying that the frame is directed to the PAN coordinator with the PAN identifier as specified in the Source PAN Identifier field.

Table 3—Possible values of the Destination Addressing Mode and Source Addressing Mode fields

Addressing mode value b_1 b_0	Description
00	PAN identifier and address fields are not present.
01	Reserved.
10	Address field contains a short address (16 bit).
11	Address field contains an extended address (64 bit).

5.2.1.1.7 Frame Version field

The Frame Version field specifies the version number corresponding to the frame.

This field shall be set to 0x00 to indicate a frame compatible with IEEE Std 802.15.4-2003 and 0x01 to indicate an IEEE 802.15.4 frame. All other field values are reserved. Details on frame compatibility are described in 5.2.3.

5.2.1.1.8 Source Addressing Mode field

The Source Addressing Mode field shall be set to one of the values listed in Table 3.

If this field is equal to zero and the Frame Type field does not specify that this frame is an acknowledgment frame, the Destination Addressing Mode field shall be nonzero, implying that the frame has originated from the PAN coordinator with the PAN identifier as specified in the Destination PAN Identifier field.

5.2.1.2 Sequence Number field

The Sequence Number field specifies the sequence identifier for the frame.

For a beacon frame, the Sequence Number field shall specify a BSN. For a data, acknowledgment, or MAC command frame, the Sequence Number field shall specify a DSN that is used to match an acknowledgment frame to the data or MAC command frame.

5.2.1.3 Destination PAN Identifier field

The Destination PAN Identifier field, when present, specifies the unique PAN identifier of the intended recipient of the frame. A value of 0xffff in this field shall represent the broadcast PAN identifier, which shall be accepted as a valid PAN identifier by all devices currently listening to the channel.

This field shall be included in the MAC frame only if the Destination Addressing Mode field is nonzero.

5.2.1.4 Destination Address field

The Destination Address field, when present, specifies the address of the intended recipient of the frame. A value of 0xffff in this field shall represent the broadcast short address, which shall be accepted as a valid address by all devices currently listening to the channel.

This field shall be included in the MAC frame only if the Destination Addressing Mode field is nonzero.

5.2.1.5 Source PAN Identifier field

The Source PAN Identifier field, when present, specifies the unique PAN identifier of the originator of the frame. This field shall be included in the MAC frame only if the Source Addressing Mode field is nonzero and the PAN ID Compression field is equal to zero.

The PAN identifier of a device is initially determined during association on a PAN but may change following a PAN identifier conflict resolution, as described in 5.1.2.2.

5.2.1.6 Source Address field

The Source Address field, when present, specifies the address of the originator of the frame. This field shall be included in the MAC frame only if the Source Addressing Mode field is nonzero.

5.2.1.7 Auxiliary Security Header field

The Auxiliary Security Header field specifies information required for security processing. This field shall be present only if the Security Enabled field is set to one. The formatting of the Auxiliary Security Header field is described in 7.4.

5.2.1.8 Frame Payload field

The Frame Payload field contains information specific to individual frame types. If the Security Enabled field is set to one, the frame payload the frame may be cryptographically protected, as described in Clause 7.

5.2.1.9 FCS field

The FCS field contains a 16-bit ITU-T CRC. The FCS is calculated over the MHR and MAC payload parts of the frame.

The FCS shall be calculated using the following standard generator polynomial of degree 16:

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

The FCS shall be calculated for transmission using the following algorithm:

— Let $M(x) = b_0x^{k-1} + b_1x^{k-2} + \ldots + b_{k-2}x + b_{k-1}$ be the polynomial representing the sequence of bits for which the checksum is to be computed.
— Multiply $M(x)$ by x^{16}, giving the polynomial $x^{16} \times M(x)$.
— Divide $x^{16} \times M(x)$ modulo 2 by the generator polynomial, $G_{16}(x)$, to obtain the remainder polynomial, $R(x) = r_0x^{15} + r_1x^{14} + \ldots + r_{14}x + r_{15}$.
— The FCS field is given by the coefficients of the remainder polynomial, *R(x)*.

Here, binary polynomials are represented as bit strings, in highest polynomial degree first order.

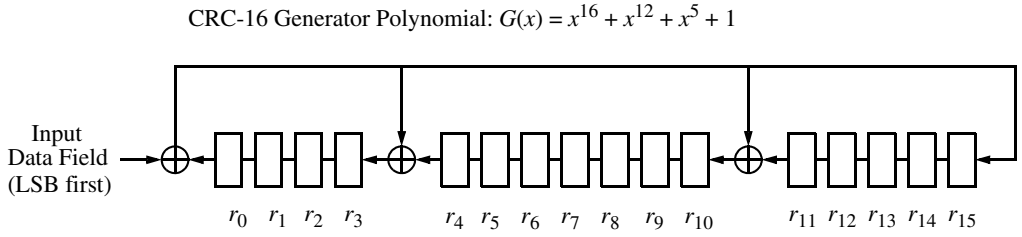
As an example, consider an acknowledgment frame with no payload and the following 3 byte MHR:

0100 0000 0000 0000 0101 0110 [leftmost bit (b_0) transmitted first in time]
b_0.....b_{23}

The FCS for this case would be the following:

0010 0111 1001 1110 [leftmost bit (r_0) transmitted first in time]
r_0.....r_{15}

A typical implementation is depicted in Figure 37.



1. Initialize the remainder register (r_0 through r_{15}) to zero.
2. Shift MHR and payload into the divider in the order of transmission (LSB first).
3. After the last bit of the data field is shifted into the divider, the remainder register contains the FCS.
4. The FCS is appended to the data field so that r_0 is transmitted first.

Figure 37—Typical FCS implementation

5.2.2 Format of individual frame types

5.2.2.1 Beacon frame format

The beacon frame shall be formatted as illustrated in Figure 38.

Octets: 2	**1**	**4/10**	**0/5/6/10/14**	**2**	**variable**	**variable**	**variable**	**2**
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS fields	Pending address fields	Beacon Payload	FCS
MHR				MAC Payload				MFR

Figure 38—Beacon frame format

The GTS fields shall be formatted as illustrated in Figure 39, and the pending address fields shall be formatted as illustrated in Figure 40.

Octets: 1	0/1	variable
GTS Specification	GTS Directions	GTS List

Figure 39—Format of the GTS information fields

Octets: 1	variable
Pending Address Specification	Address List

Figure 40—Format of the pending address information fields

5.2.2.1.1 Beacon frame MHR fields

The Frame Type field shall contain the value that indicates a beacon frame, as shown in Table 2, and the Source Addressing Mode field shall be set to indicate the beacon source addressing mode, as defined in 5.1.2.4. The Security Enabled field shall be set to one if security is enabled and the Frame Version field is not zero. If a broadcast data or command frame is pending, the Frame Pending field shall be set to one. All other fields in the Frame Control field shall be set to zero and ignored on reception.

The Sequence Number field shall contain the current value of *macBSN*.

The addressing fields shall comprise only the source address fields. The Source PAN Identifier and Source Address fields shall contain the PAN identifier and address, respectively, of the device transmitting the beacon.

The Auxiliary Security Header field, if present, shall contain the information required for security processing of the beacon frame, as specified in 5.2.1.7.

5.2.2.1.2 Superframe Specification field

The Superframe Specification field shall be formatted as illustrated in Figure 41.

Bits: 0–3	4–7	8–11	12	13	14	15
Beacon Order	Superframe Order	Final CAP Slot	Battery Life Extension (BLE)	Reserved	PAN Coordinator	Association Permit

Figure 41—Format of the Superframe Specification field

The Beacon Order field shall specify the transmission interval of the beacon. The relationship between the beacon order and the beacon interval is explained in 5.1.1.1.

The Superframe Order field shall specify the length of time during which the superframe is active (i.e., receiver enabled), including the beacon frame transmission time. The relationship between the superframe order and the superframe duration is explained in 5.1.1.1.

The Final CAP Slot field specifies the final superframe slot utilized by the CAP. The duration of the CAP, as implied by this field, shall be greater than or equal to the value specified by *aMinCAPLength*. However, an

exception is allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance, as described in 5.2.2.1.3.

The Battery Life Extension (BLE) field shall be set to one if frames transmitted to the beaconing device during its CAP are required to start in or before *macBattLifeExtPeriods* full backoff periods after the IFS period following the beacon. Otherwise, the BLE field shall be set to zero.

The PAN Coordinator field shall be set to one if the beacon frame is being transmitted by the PAN coordinator. Otherwise, the PAN Coordinator field shall be set to zero.

The Association Permit field shall be set to one if *macAssociationPermit* is set to TRUE (i.e., the coordinator is accepting association to the PAN). The association permit bit shall be set to zero if the coordinator is currently not accepting association requests on its network.

5.2.2.1.3 GTS Specification field

The GTS Specification field shall be formatted as illustrated in Figure 42.

Bits: 0–2	3–6	7
GTS Descriptor Count	Reserved	GTS Permit

Figure 42—Format of the GTS Specification field

The GTS Descriptor Count field specifies the number of 3-octet GTS descriptors contained in the GTS List field of the beacon frame. If the value of this field is greater than zero, the size of the CAP shall be allowed to dip below *aMinCAPLength* to accommodate the temporary increase in the beacon frame length caused by the inclusion of the field. If the value of this field is zero, the GTS Directions field and GTS List field of the beacon frame are not present.

The GTS Permit field shall be set to one if *macGTSPermit* is equal to TRUE (i.e., the PAN coordinator is accepting GTS requests). Otherwise, the GTS Permit field shall be set to zero.

5.2.2.1.4 GTS Directions field

The GTS Directions field shall be formatted as illustrated in Figure 43.

Bits: 0–6	7
GTS Directions Mask	Reserved

Figure 43—Format of the GTS Directions field

The GTS Directions Mask field is a mask identifying the directions of the GTSs in the superframe. The lowest bit in the mask corresponds to the direction of the first GTS contained in the GTS List field of the beacon frame, with the remainder appearing in the order that they appear in the list. Each bit shall be set to one if the GTS is a receive-only GTS or to zero if the GTS is a transmit-only GTS. GTS direction is defined relative to the direction of the data frame transmission by the device.

5.2.2.1.5 GTS List field

The size of the GTS List field is defined by the values specified in the GTS Specification field of the beacon frame and contains the list of GTS descriptors that represents the GTSs that are being maintained. The maximum number of GTS descriptors shall be limited to seven.

Each GTS descriptor shall be formatted as illustrated in Figure 44.

Bits: 0–15	16–19	20–23
Device Short Address	GTS Starting Slot	GTS Length

Figure 44—Format of the GTS descriptor

The Device Short Address field shall contain the short address of the device for which the GTS descriptor is intended.

The GTS Starting Slot field contains the superframe slot at which the GTS is to begin.

The GTS Length field contains the number of contiguous superframe slots over which the GTS is active.

5.2.2.1.6 Pending Address Specification field

The Pending Address Specification field shall be formatted as illustrated in Figure 45.

Bits: 0–2	3	4-6	7
Number of Short Addresses Pending	Reserved	Number of Extended Addresses Pending	Reserved

Figure 45—Format of the Pending Address Specification field

The Number of Short Addresses Pending field indicates the number of short addresses contained in the Address List field of the beacon frame.

The Number of Extended Addresses Pending field indicates the number of extended addresses contained in the Address List field of the beacon frame.

5.2.2.1.7 Address List field

The size of the Address List field is determined by the values specified in the Pending Address Specification field of the beacon frame and contains the list of addresses of the devices that currently have messages pending with the coordinator. The address list shall not contain the broadcast short address.

The maximum number of addresses pending shall be limited to seven and may comprise both short and extended addresses. All pending short addresses shall appear first in the list followed by any extended addresses. If the coordinator is able to store more than seven transactions, it shall indicate them in its beacon on a first-come-first-served basis, ensuring that the beacon frame contains at most seven addresses.

5.2.2.1.8 Beacon Payload field

The Beacon Payload field is an optional sequence of up to *aMaxBeaconPayloadLength* specified to be transmitted in the beacon frame by the next higher layer. The set of octets contained in *macBeaconPayload* shall be copied into this field.

5.2.2.2 Data frame format

The data frame shall be formatted as illustrated in Figure 46.

Octets: 2	**1**	**variable**	**0/5/6/10/14**	**variable**	**2**
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MAC Payload	MFR

Figure 46—Data frame format

5.2.2.2.1 Data frame MHR fields

The Frame Type field shall contain the value that indicates a data frame, as shown in Table 2. The Security Enabled field shall be set to one if security is enabled and the Frame Version field is not zero. All other fields in the Frame Control field shall be set appropriately according to the intended use of the data frame.

The Sequence Number field shall contain the current value of *macDSN*.

The addressing fields shall comprise the destination address fields and/or the source address fields, dependent on the settings in the Frame Control field.

The Auxiliary Security Header field, if present, shall contain the information required for security processing of the data frame, as specified in 5.2.1.7.

5.2.2.2.2 Data Payload field

The payload of a data frame shall contain the sequence of octets that the next higher layer has requested the MAC sublayer to transmit.

5.2.2.3 Acknowledgment frame format

The acknowledgment frame shall be formatted as illustrated in Figure 47.

Octets: 2	**1**	**2**
Frame Control	Sequence Number	FCS
MHR		MFR

Figure 47—Acknowledgment frame format

The Frame Type field shall contain the value that indicates an acknowledgment frame, as shown in Table 2. If the acknowledgment frame is being sent in response to a received data request command, the device sending the acknowledgment frame shall determine whether it has data pending for the recipient. If the device can determine this before sending the acknowledgment frame, as described in 5.1.6.4.2, it shall set the Frame Pending field according to whether there is pending data. Otherwise, the Frame Pending field shall be set to one. If the acknowledgment frame is being sent in response to either a data frame or another type of MAC command frame, the device shall set the Frame Pending field to zero. All other fields in the Frame Control field shall be set to zero and ignored on reception.

The Sequence Number field shall contain the value of the sequence number received in the frame for which the acknowledgment is to be sent.

5.2.2.4 MAC command frame format

The MAC command frame shall be formatted as illustrated in Figure 48.

Octets: 2	**1**	**variable**	**0/5/6/10/14**	**1**	**variable**	**2**
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MAC Payload		MFR

Figure 48—MAC command frame format

5.2.2.4.1 MAC command frame MHR fields

The Frame Type field shall contain the value that indicates a MAC command frame, as shown in Table 2. If the frame is to be secured, the Security Enabled field shall be set to one and the frame secured according to the process described in 7.2.1. Otherwise the Security Enabled field shall be set to zero. All other fields in the Frame Control field shall be set appropriately according to the intended use of the MAC command frame.

The Sequence Number field shall contain the current value of *macDSN*.

The addressing fields shall comprise the destination address fields and/or the source address fields, dependent on the settings in the Frame Control field.

The Auxiliary Security Header field, if present, shall contain the information required for security processing of the MAC command frame, as specified in 5.2.1.7.

5.2.2.4.2 Command Frame Identifier field

The Command Frame Identifier field identifies the MAC command being used. Valid values of the Command Frame Identifier field are defined in Table 5.

5.2.2.4.3 Command Payload field

The Command Payload field contains the MAC command itself. The formats of the individual commands are described in 5.3.

5.2.3 Frame compatibility

All unsecured frames specified in this standard are compatible with unsecured frames compliant with IEEE Std 802.15.4-2003, with two exceptions: a coordinator realignment command frame with the Channel Page field present, as defined in 5.3.8, and any frame with a MAC Payload field larger than *aMaxMACSafePayloadSize*.

Compatibility for secured frames is shown in Table 4, which identifies the security operating modes for IEEE Std 802.15.4-2003 and this standard.

Table 4—Frame compatibility between IEEE Std 802.15.4-2003 and this standard

Frame Control field bit assignments		Functionality
Security enabled b_3	Frame version b_{13} b_{12}	
0	00	No security. Frames are compatible between IEEE Std 802.15.4-2003 and this standard.
0	01	No security. Frames are not compatible between IEEE Std 802.15.4-2003 and this standard.
1	00	Secured frame formatted according to IEEE Std 802.15.4-2003. This type of frame is not supported in this standard.
1	01	Secured frame formatted according to this standard.

5.3 MAC command frames

The command frames defined by the MAC sublayer are listed in Table 5. An FFD shall be capable of transmitting and receiving all command frame types, with the exception of the GTS request command, while the requirements for an RFD are indicated by an "X" in the table. MAC commands shall only be transmitted in the CAP for beacon-enabled PANs or at any time for nonbeacon-enabled PANs.

MAC command reception shall abide by the procedure described in 5.1.6.2.

5.3.1 Association request command

The association request command allows a device to request association with a PAN through the PAN coordinator or a coordinator.

This command shall only be sent by an unassociated device that wishes to associate with a PAN. A device shall only associate with a PAN through the PAN coordinator or a coordinator allowing association, as determined through the scan procedure.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The association request command shall be formatted as illustrated in Figure 49.

Table 5—MAC command frames

Command frame identifier	Command name	RFD		Subclause
		Tx	Rx	
0x01	Association request	X		5.3.1
0x02	Association response		X	5.3.2
0x03	Disassociation notification	X	X	5.3.3
0x04	Data request	X		5.3.4
0x05	PAN ID conflict notification	X		5.3.5
0x06	Orphan notification	X		5.3.6
0x07	Beacon request			5.3.7
0x08	Coordinator realignment		X	5.3.8
0x09	GTS request			5.3.9
0x0a–0xff	Reserved			—

Octets: variable	1	1
MHR fields	Command Frame Identifier	Capability Information

Figure 49—Association request command format

5.3.1.1 MHR fields

The Source Addressing Mode field shall be set to indicate extended addressing. The Destination Addressing Mode field shall be set to the same mode as indicated in the beacon frame to which the association request command refers.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

The Destination PAN Identifier field shall contain the identifier of the PAN to which to associate. The Destination Address field shall contain the address from the beacon frame that was transmitted by the coordinator to which the association request command is being sent. The Source PAN Identifier field shall contain the broadcast PAN identifier. The Source Address field shall contain the value of *macExtendedAddress*.

5.3.1.2 Capability Information field

The Capability Information field shall be formatted as illustrated in Figure 50.

The Device Type field shall be set to one if the device is an FFD. Otherwise, the Device Type field shall be set to zero to indicate an RFD.

The Power Source field shall be set to one if the device is receiving power from the alternating current mains. Otherwise, the Power Source field shall be set to zero.

Bits: 0	1	2	3	4–5	6	7
Reserved	Device Type	Power Source	Receiver On When Idle	Reserved	Security Capability	Allocate Address

Figure 50—Capability Information field format

The Receiver On When Idle field shall be set to one if the device does not disable its receiver to conserve power during idle periods. Otherwise, the Receiver On When Idle field shall be set to zero.

The Security Capability field shall be set to one if the device is capable of sending and receiving cryptographically protected MAC frames as specified in 7.2; it shall be set to zero otherwise.

The Allocate Address field shall be set to one if the device wishes the coordinator to allocate a short address as a result of the association procedure. Otherwise, it shall be set to zero.

5.3.2 Association response command

The association response command allows the PAN coordinator or a coordinator to communicate the results of an association attempt back to the device requesting association.

This command shall only be sent by the PAN coordinator or coordinator to a device that is currently trying to associate.

All devices shall be capable of receiving this command, although an RFD is not required to be capable of transmitting it.

The association response command shall be formatted as illustrated in Figure 51.

Octets: variable	1	2	1
MHR fields	Command Frame Identifier	Short Address	Association Status

Figure 51—Association response command format

5.3.2.1 MHR fields

The Destination Addressing Mode and Source Addressing Mode fields shall each be set to indicate extended addressing.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

The PAN ID Compression field shall be set to one. In accordance with this value of the PAN ID Compression field, the Destination PAN Identifier field shall contain the value of *macPANId*, while the Source PAN Identifier field shall be omitted. The Destination Address field shall contain the extended address of the device requesting association. The Source Address field shall contain the value of *macExtendedAddress*.

5.3.2.2 Short Address field

If the coordinator was not able to associate the device to its PAN, the Short Address field shall be set to 0xffff, and the Association Status field shall contain the reason for the failure. If the coordinator was able to associate the device to its PAN, this field shall contain the short address that the device may use in its communications on the PAN until it is disassociated.

A Short Address field value equal to 0xfffe shall indicate that the device has been successfully associated with a PAN but has not been allocated a short address. In this case, the device shall communicate on the PAN using only its extended address.

5.3.2.3 Association Status field

Valid values of the Association Status field are defined in Table 6.

Table 6—Valid values of the Association Status field

Association status	Description
0x00	Association successful.
0x01	PAN at capacity.
0x02	PAN access denied.
0x03–0x7f	Reserved.
0x80–0xff	Reserved for MAC primitive enumeration values.

5.3.3 Disassociation notification command

The PAN coordinator, a coordinator, or an associated device may send the disassociate notification command.

All devices shall implement this command.

The disassociation notification command shall be formatted as illustrated in Figure 52.

Octets: variable	1	1
MHR fields	Command Frame Identifier	Disassociation Reason

Figure 52—Disassociation notification command format

5.3.3.1 MHR fields

The Destination Addressing Mode field shall be set according to the addressing mode specified by the corresponding primitive. The Source Addressing Mode field shall be set to indicate extended addressing.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

The PAN ID Compression field shall be set to one. In accordance with this value of the PAN ID Compression field, the Destination PAN Identifier field shall contain the value of *macPANId*, while the Source PAN Identifier field shall be omitted. If the coordinator wants an associated device to leave the PAN, then the Destination Address field shall contain the address of the device being removed from the PAN. If an associated device wants to leave the PAN, then the Destination Address field shall contain the value of either *macCoordShortAddress*, if the Destination Addressing Mode field is equal to two, or *macCoordExtendedAddress*, if the Destination Addressing Mode field is equal to three. The Source Address field shall contain the value of *macExtendedAddress*.

5.3.3.2 Disassociation Reason field

Valid values of the Disassociation Reason field are defined in Table 7.

Table 7—Valid disassociation reason codes

Disassociate reason	Description
0x00	Reserved.
0x01	The coordinator wishes the device to leave the PAN.
0x02	The device wishes to leave the PAN.
0x03–0x7f	Reserved.
0x80–0xff	Reserved for MAC primitive enumeration values.

5.3.4 Data request command

The data request command is sent by a device to request data from the PAN coordinator or a coordinator.

There are three cases for which this command is sent. On a beacon-enabled PAN, this command shall be sent by a device when *macAutoRequest* is equal to TRUE and a beacon frame indicating that data are pending for that device is received from its coordinator. The coordinator indicates pending data in its beacon frame by adding the address of the recipient of the data to the Address List field. This command shall also be sent when instructed to do so by the next higher layer on reception of the MLME-POLL.request primitive. In addition, a device may send this command to the coordinator *macResponseWaitTime* after the acknowledgment to an association request command.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The data request command shall be formatted as illustrated in Figure 53.

Octets: variable	1
MHR fields	Command Frame Identifier

Figure 53—Data request command format

If the data request command is being sent in response to the receipt of a beacon frame indicating that data are pending for that device, the Destination Addressing Mode field may be set to indicate that destination addressing information is not present if the beacon frame indicated in its Superframe Specification field, as defined in 5.2.2.1.2, that it originated from the PAN coordinator, as defined in 5.2.1.1.6, or set otherwise according to the coordinator to which the data request command is directed. If the destination addressing information is to be included, the Destination Addressing Mode field shall be set according to the value of *macCoordShortAddress*. If *macCoordShortAddress* is equal to 0xfffe, the Destination Addressing Mode field shall be set to indicate extended addressing, and the Destination Address field shall contain the value of *macCoordExtendedAddress*. Otherwise, the Destination Addressing Mode field shall be set to indicate short addressing and the Destination Address field shall contain the value of *macCoordShortAddress*.

If the data request command is being sent in response to the receipt of a beacon frame indicating that data are pending for that device, the Source Addressing Mode field shall be set according to the addressing mode used for the pending address. If the Source Addressing Mode field is set to indicate short addressing, the Source Address field shall contain the value of *macShortAddress*. Otherwise, the Source Addressing Mode field shall be set to indicate extended addressing and the Source Address field shall contain the value of *macExtendedAddress*.

If the data request command is triggered by the reception of an MLME-POLL.request primitive from the next higher layer, then the destination addressing information shall be the same as that contained in the primitive. The Source Addressing Mode field shall be set according to the value of *macShortAddress*. If *macShortAddress* is less than 0xfffe, short addressing shall be used. Extended addressing shall be used otherwise.

If the data request command is being sent following the acknowledgment to an association request command frame, the Destination Addressing Mode field shall be set according to the coordinator to which the data request command is directed. If *macCoordShortAddress* is equal to 0xfffe, extended addressing shall be used. Short addressing shall be used otherwise. The Source Addressing Mode field shall be set to use extended addressing.

If the Destination Addressing Mode field is set to indicate that destination addressing information is not present, the PAN ID Compression field shall be set to zero and the source PAN identifier shall contain the value of *macPANId*. Otherwise, the PAN ID Compression field shall be set to one. In this case and in accordance with the PAN ID Compression field, the Destination PAN Identifier field shall contain the value of *macPANId*, while the Source PAN Identifier field shall be omitted.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

5.3.5 PAN ID conflict notification command

The PAN ID conflict notification command is sent by a device to the PAN coordinator when a PAN identifier conflict is detected.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The PAN ID conflict notification command shall be formatted as illustrated in Figure 54.

Octets: variable	**1**
MHR fields	Command Frame Identifier

Figure 54—PAN ID conflict notification command format

The Destination Addressing Mode and Source Addressing Mode fields shall both be set to indicate extended addressing.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

The PAN ID Compression field shall be set to one. In accordance with this value of the PAN ID Compression field, the Destination PAN Identifier field shall contain the value of *macPANId*, while the Source PAN Identifier field shall be omitted. The Destination Address field shall contain the value of *macCoordExtendedAddress*. The Source Address field shall contain the value of *macExtendedAddress*.

5.3.6 Orphan notification command

The orphan notification command is used by an associated device that has lost synchronization with its coordinator.

All devices shall be capable of transmitting this command, although an RFD is not required to be capable of receiving it.

The orphan notification command shall be formatted as illustrated in Figure 55.

Octets: 15	**1**
MHR fields	Command Frame Identifier

Figure 55—Orphan notification command format

The Source Addressing Mode field shall be set to indicate extended addressing. The Destination Addressing Mode field shall be set to indicate short addressing.

The Frame Pending field and AR field shall be set to zero and ignored upon reception.

The PAN ID Compression field shall be set to one. In accordance with this value of the PAN ID Compression field, the Destination PAN Identifier field shall contain the value of the broadcast PAN identifier, while the Source PAN Identifier field shall be omitted. The Destination Address field shall contain the broadcast short address. The Source Address field shall contain the value of *macExtendedAddress*.

5.3.7 Beacon request command

The beacon request command is used by a device to locate all coordinators within its radio communications range during an active scan.

This command is optional for an RFD.

The beacon request command shall be formatted as illustrated in Figure 56.

The Destination Addressing Mode field shall be set to indicate short addressing, and the Source Addressing Mode field shall be set to indicate that the source addressing information is not present.

The Frame Pending field shall be set to zero and ignored upon reception. The AR field and Security Enabled field shall also be set to zero.

Octets: 7	1
MHR fields	Command Frame Identifier

Figure 56—Beacon request command format

The Destination PAN Identifier field shall contain the broadcast PAN identifier. The Destination Address field shall contain the broadcast short address.

5.3.8 Coordinator realignment command

The coordinator realignment command is sent by the PAN coordinator or a coordinator either following the reception of an orphan notification command from a device that is recognized to be on its PAN or when any of its PAN configuration attributes change due to the receipt of an MLME-START.request primitive.

If this command is sent following the reception of an orphan notification command, it is sent directly to the orphaned device. If this command is sent when any PAN configuration attributes (i.e., PAN identifier, short address, channel, or channel page) change, it is broadcast to the PAN.

All devices shall be capable of receiving this command, although an RFD is not required to be capable of transmitting it.

The coordinator realignment command shall be formatted as illustrated in Figure 57.

Octets: variable	1	2	2	1	2	0/1
MHR fields	Command Frame Identifier	PAN Identifier	Coordinator Short Address	Channel Number	Short Address	Channel page

Figure 57—Coordinator realignment command format

5.3.8.1 MHR fields

The Destination Addressing Mode field shall be set to indicate extended addressing if the command is directed to an orphaned device or set to indicate short addressing if it is to be broadcast to the PAN. The Source Addressing Mode field shall be set to indicate extended addressing.

The Frame Pending field shall be set to zero and ignored upon reception.

The AR field shall be set to one if the command is directed to an orphaned device or set to zero if the command is to be broadcast to the PAN.

The Frame Version field shall be set to 0x01 if the Channel Page field is present. Otherwise it shall be set as specified in 5.2.3.

The Destination PAN Identifier field shall contain the broadcast PAN identifier. The Destination Address field shall contain the extended address of the orphaned device if the command is directed to an orphaned device. Otherwise, the Destination Address field shall contain the broadcast short address. The Source PAN Identifier field shall contain the value of *macPANId*, and the Source Address field shall contain the value of *macExtendedAddress*.

5.3.8.2 PAN Identifier field

The PAN Identifier field shall contain the PAN identifier that the coordinator intends to use for all future communications.

5.3.8.3 Coordinator Short Address field

The Coordinator Short Address field shall contain the value of *macShortAddress*.

5.3.8.4 Channel Number field

The Channel Number field shall contain the channel number that the coordinator intends to use for all future communications.

5.3.8.5 Short Address field

If the coordinator realignment command is broadcast to the PAN, the Short Address field shall be set to 0xffff and ignored on reception.

If the coordinator realignment command is sent directly to an orphaned device, this field shall contain the short address that the orphaned device shall use to operate on the PAN. If the orphaned device does not have a short address, because it always uses its extended address, this field shall contain the value 0xfffe.

5.3.8.6 Channel Page field

The Channel Page field, if present, shall contain the channel page that the coordinator intends to use for all future communications. This field may be omitted if the new channel page is the same as the previous channel page.

5.3.9 GTS request command

The GTS request command is used by an associated device that is requesting the allocation of a new GTS or the deallocation of an existing GTS from the PAN coordinator. Only devices that have a short address less than 0xfffe shall send this command.

This command is optional.

The GTS request command shall be formatted as illustrated in Figure 58.

Octets: 7	**1**	**1**
MHR fields	Command Frame Identifier	GTS Characteristics

Figure 58—GTS request command format

5.3.9.1 MHR fields

The Destination Addressing Mode field shall be set to indicate that destination addressing information is not present, and the Source Addressing Mode field shall be set to indicate short addressing.

The Frame Pending field shall be set to zero and ignored upon reception, and the AR field shall be set to one.

The Source PAN Identifier field shall contain the value of *macPANId*, and the Source Address field shall contain the value of *macShortAddress*.

5.3.9.2 GTS Characteristics field

The GTS Characteristics field shall be formatted as illustrated in Figure 59.

Bits: 0–3	4	5	6–7
GTS Length	GTS Direction	Characteristics Type	Reserved

Figure 59—GTS Characteristics field format

The GTS Length field shall contain the number of superframe slots being requested for the GTS.

The GTS Direction field shall be set to one if the GTS is to be a receive-only GTS. Conversely, this field shall be set to zero if the GTS is to be a transmit-only GTS. GTS direction is defined relative to the direction of data frame transmissions by the device.

The Characteristics Type field shall be set to one if the characteristics refer to a GTS allocation or zero if the characteristics refer to a GTS deallocation.

6. MAC services

6.1 Overview

The MAC sublayer provides an interface between the next higher layer and the PHY. The MAC sublayer conceptually includes a management entity called the MLME. This entity provides the service interfaces through which layer managementons may be invoked. The MLME is also responsible for maintaining a database of managed objects pertaining to the MAC sublayer. This database is referred to as the MAC sublayer PIB.

Figure 60 depicts the components and interfaces of the MAC sublayer.

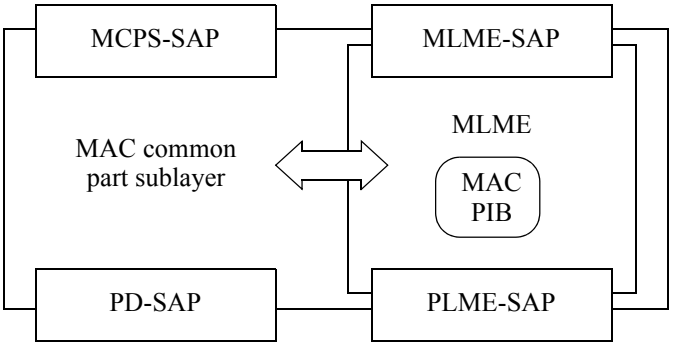


Figure 60—MAC sublayer reference model

The MAC sublayer provides two services, accessed through two SAPs:

— The MAC data service, accessed through the MAC common part sublayer (MCPS) data SAP (MCPS-SAP)

— The MAC management service, accessed through the MLME-SAP

These two services provide the interface between the next higher layer and the PHY. In addition to these external interfaces, an implicit interface also exists between the MLME and the MCPS that allows the MLME to use the MAC data service.

6.2 MAC management service

The MLME-SAP allows the transport of management commands between the next higher layer and the MLME. Table 8 summarizes the primitives supported by the MLME through the MLME-SAP interface. Primitives marked with a diamond (♦) are optional for an RFD. Primitives marked with an asterisk (*) are optional for both device types (i.e., RFD and FFD). The primitives are discussed in the subclauses referenced in the table.

Table 8—Summary of the primitives accessed through the MLME-SAP

Name	Request	Indication	Response	Confirm
MLME-ASSOCIATE	6.2.2.1	6.2.2.2♦	6.2.2.3♦	6.2.2.4
MLME-DISASSOCIATE	6.2.3.1	6.2.3.2		6.2.3.3

Table 8—Summary of the primitives accessed through the MLME-SAP *(continued)*

Name	Request	Indication	Response	Confirm
MLME-BEACON-NOTIFY		6.2.4.1		
MLME-COMM-STATUS		6.2.4.2		
MLME-GET	6.2.5.1			6.2.5.2
MLME-GTS	6.2.6.1*	6.2.6.3*		6.2.6.2*
MLME-ORPHAN		6.2.7.1♦	6.2.7.2♦	
MLME-RESET	6.2.8.1			6.2.8.2
MLME-RX-ENABLE	6.2.9.1*			6.2.9.2*
MLME-SCAN	6.2.10.1			6.2.10.2
MLME-SET	6.2.11.1			6.2.11.2
MLME-START	6.2.12.1♦			6.2.12.2♦
MLME-SYNC	6.2.13.1*			
MLME-SYNC-LOSS		6.2.13.2		
MLME-POLL	6.2.14.1			6.2.14.2
MLME-DPS	6.2.15.1*	6.2.15.3*		6.2.15.2*
MLME-SOUNDING	6.2.16.1*			6.2.16.1*
MLME-CALIBRATE	6.2.17.1*			6.2.17.2*

6.2.1 Common requirements for MLME primitives

If any error occurs during the outgoing frame security procedure, as described in 7.2.1, for a request primitive, the MLME will discard the frame and issue the corresponding confirm primitive with the error status returned by the outgoing frame security procedure.

If any error occurs during the outgoing frame security procedure, as described in 7.2.1, for a response primitive, the MLME will discard the frame and issue the MLME-COMM-STATUS.indication primitive with the error status returned by the outgoing frame security procedure.

If any parameter in request primitive is not supported or is out of range, the MAC sublayer will issue the corresponding confirm primitive with a status of INVALID_PARAMETER.

If any parameter in response primitive is not supported or is out of range, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of INVALID_PARAMETER.

If the MLME is unable to send the frame required by a request primitive due to a CSMA-CA algorithm failure, the MLME will issue the corresponding confirm primitive with a status parameter value of CHANNEL_ACCESS_FAILURE.

If the MLME successfully transmits the frame required by a request primitive, but the expected acknowledgment is not received, the MLME will issue the corresponding confirm primitive with a status parameter value of NO_ACK.

6.2.2 Association primitives

These primitives are used when a device becomes associated with a PAN.

6.2.2.1 MLME-ASSOCIATE.request

The MLME-ASSOCIATE.request primitive is used by a device to request an association with a coordinator.

The semantics of this primitive are:

```
MLME-ASSOCIATE.request          (
                                ChannelNumber,
                                ChannelPage,
                                CoordAddrMode,
                                CoordPANId,
                                CoordAddress,
                                CapabilityInformation,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

The primitive parameters are defined in Table 9.

Table 9—MLME-ASSOCIATE.request parameters

Name	Type	Valid range	Description
ChannelNumber	Integer	Any valid channel number	The channel number on which to attempt association.
ChannelPage	Integer	Any valid channel page	The channel page on which to attempt association.
CoordAddrMode	Enumeration	SHORT_ADDRESS, EXTENDED_ADDRESS	The coordinator addressing mode for this primitive and subsequent MPDU.
CoordPANId	Integer	0x0000–0xffff	The identifier of the PAN with which to associate.
CoordAddress	Device address	As specified by the CoordAddrMode parameter	The address of the coordinator with which to associate.
CapabilityInformation	Bitmap	As defined in 5.3.1.2	Specifies the operational capabilities of the associating device.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

On receipt of the MLME-ASSOCIATE.request primitive, the MLME of an unassociated device first updates the appropriate PHY and MAC PIB attributes, as described in 5.1.3.1, and then generates an association request command, as defined in 5.3.1.

The SecurityLevel parameter specifies the level of security to be applied to the association request command frame. Typically, the association request command should not be implemented using security. However, if the device requesting association shares a key with the coordinator, then security may be specified.

6.2.2.2 MLME-ASSOCIATE.indication

The MLME-ASSOCIATE.indication primitive is used to indicate the reception of an association request command.

The semantics of this primitive are:

```
MLME-ASSOCIATE.indication (
                          DeviceAddress,
                          CapabilityInformation,
                          SecurityLevel,
                          KeyIdMode,
                          KeySource,
                          KeyIndex
                          )
```

The primitive parameters are defined in Table 10.

Table 10—MLME-ASSOCIATE.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended IEEE address	The address of the device requesting association.
CapabilityInformation	Bitmap	As defined in 5.3.1.2	The operational capabilities of the device requesting association.
SecurityLevel	Integer	As defined in Table 48	As defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	As defined in Table 48.
KeySource	Set of octets	As defined in Table 48	As defined in Table 48.
KeyIndex	Integer	As defined in Table 48	As defined in Table 48.

When the next higher layer of a coordinator receives the MLME-ASSOCIATE.indication primitive, the coordinator determines whether to accept or reject the unassociated device using an algorithm outside the scope of this standard.

6.2.2.3 MLME-ASSOCIATE.response

The MLME-ASSOCIATE.response primitive is used to initiate a response to an MLME-ASSOCIATE.indication primitive.

The semantics of this primitive are:

```
MLME-ASSOCIATE.response        (
                               DeviceAddress,
                               AssocShortAddress,
                               status,
                               SecurityLevel,
                               KeyIdMode,
                               KeySource,
                               KeyIndex
                               )
```

The primitive parameters are defined in Table 11.

Table 11—MLME-ASSOCIATE.response parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended 64 bit IEEE address	The address of the device requesting association.
AssocShortAddress	Integer	0x0000–0xffff	The short device address allocated by the coordinator on successful association. This parameter is set to 0xffff if the association was unsuccessful.
status	Enumeration	As defined in 5.3.2.3	The status of the association attempt.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

When the MLME of a coordinator receives the MLME-ASSOCIATE.response primitive, it generates an association response command, as described in 5.3.2, and attempts to send it to the device requesting association, as described in 5.1.3.1.

6.2.2.4 MLME-ASSOCIATE.confirm

The MLME-ASSOCIATE.confirm primitive is used to inform the next higher layer of the initiating device whether its request to associate was successful or unsuccessful.

The semantics of this primitive are:

```
MLME-ASSOCIATE.confirm      (
                            AssocShortAddress,
                            status,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

The primitive parameters are defined in Table 12.

Table 12—MLME-ASSOCIATE.confirm parameters

Name	Type	Valid range	Description
AssocShortAddress	Integer	0x0000–0xffff	The short device address allocated by the coordinator on successful association. This parameter will be equal to 0xffff if the association attempt was unsuccessful.
status	Enumeration	The value of the Status field of the association response command, as defined in 5.3.2.3, SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, SECURITY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY, INVALID_PARAMETER	The status of the association attempt.
SecurityLevel	Integer	As defined in Table 46 or Table 48	If the primitive were generated following failed outgoing processing of an association request command, then it is as defined in Table 46. If the primitive were generated following receipt of an association response command, then it is as defined in Table 48.
KeyIdMode	Integer	As defined in Table 46 or Table 48	If the primitive were generated following failed outgoing processing of an association request command, then it is as defined in Table 46. If the primitive were generated following receipt of an association response command, then it is as defined in Table 48.

Table 12—MLME-ASSOCIATE.confirm parameters *(continued)*

Name	Type	Valid range	Description
KeySource	Set of octets	As defined in Table 46 or Table 48	If the primitive were generated following failed outgoing processing of an association request command, then it is as defined in Table 46. If the primitive were generated following receipt of an association response command, then it is as defined in Table 48.
KeyIndex	Integer	As defined in Table 46 or Table 48	If the primitive were generated following failed outgoing processing of an association request command, then it is as defined in Table 46. If the primitive were generated following receipt of an association response command, then it is as defined in Table 48.

If the association request was successful, then the status parameter will be set to SUCCESS. Otherwise, the status parameter will be set to indicate the type of failure.

6.2.3 Disassociation primitives

These primitives are used by a device to disassociate from a PAN or by the coordinator to disassociate a device from a PAN.

6.2.3.1 MLME-DISASSOCIATE.request

The MLME-DISASSOCIATE.request primitive is used by an associated device to notify the coordinator of its intent to leave the PAN. It is also used by the coordinator to instruct an associated device to leave the PAN.

The semantics of this primitive are:

```
MLME-DISASSOCIATE.request       (
                                DeviceAddrMode,
                                DevicePANId,
                                DeviceAddress,
                                DisassociateReason,
                                TxIndirect,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

The primitive parameters are defined in Table 13.

If the DeviceAddrMode parameter is equal to SHORT_ADDRESS and the DeviceAddress parameter is equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to EXTENDED_ADDRESS and the DeviceAddress parameter is equal to *macCoordExtendedAddress*, the TxIndirect parameter is

Table 13—MLME-DISASSOCIATE.request parameters

Name	Type	Valid range	Description
DeviceAddrMode	Enumeration	SHORT_ADDRESS, EXTENDED_ADDRESS	The addressing mode of the device to which to send the disassociation notification command.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier of the device to which to send the disassociation notification command.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter	The address of the device to which to send the disassociation notification command.
DisassociateReason	Integer	0x00–0xff	The reason for the disassociation, as described in 5.3.2.3.
TxIndirect	Boolean	TRUE, FALSE	TRUE if the disassociation notification command is to be sent indirectly.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

ignored, and the MLME sends a disassociation notification command, as defined in 5.3.3, to its coordinator in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN.

If the DeviceAddrMode parameter is equal to SHORT_ADDRESS and the DeviceAddress parameter is not equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to EXTENDED_ADDRESS and the DeviceAddress parameter is not equal to *macCoordExtendedAddress*, and if this primitive was received by the MLME of a coordinator with the TxIndirect parameter set to TRUE, the disassociation notification command will be sent using indirect transmission, as described in 5.1.5.

If the DeviceAddrMode parameter is equal to SHORT_ADDRESS and the DeviceAddress parameter is not equal to *macCoordShortAddress* or if the DeviceAddrMode parameter is equal to EXTENDED_ADDRESS and the DeviceAddress parameter is not equal to *macCoordExtendedAddress*, and if this primitive was received by the MLME of a coordinator with the TxIndirect parameter set to FALSE, the MLME sends a disassociation notification command to the device in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN.

6.2.3.2 MLME-DISASSOCIATE.indication

The MLME-DISASSOCIATE.indication primitive is used to indicate the reception of a disassociation notification command.

The semantics of this primitive are:

```
MLME-DISASSOCIATE.indication    (
                                DeviceAddress,
                                DisassociateReason,
                                SecurityLevel,
                                KeyIdMode,
                                KeySource,
                                KeyIndex
                                )
```

The primitive parameters are defined in Table 14.

Table 14—MLME-DISASSOCIATE.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	An extended IEEE address	The address of the device requesting disassociation.
DisassociateReason	Integer	0x00–0xff	The reason for the disassociation, as defined in 5.3.3.2.
SecurityLevel	Integer	As defined in Table 48	As defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	As defined in Table 48.
KeySource	Set of octets	As defined in Table 48	As defined in Table 48.
KeyIndex	Integer	As defined in Table 48	As defined in Table 48.

6.2.3.3 MLME-DISASSOCIATE.confirm

The MLME-DISASSOCIATE.confirm primitive reports the results of an MLME-DISASSOCIATE.request primitive.

The semantics of this primitive are:

```
MLME-DISASSOCIATE.confirm    (
                             status,
                             DeviceAddrMode,
                             DevicePANId,
                             DeviceAddress
                             )
```

The primitive parameters are defined in Table 15.

This primitive returns a status of either SUCCESS, indicating that the disassociation request was successful, or the appropriate status parameter value indicating the reason for failure.

If the DevicePANId parameter is not equal to *macPANId* in the MLME-DISASSOCIATE.request primitive, the status parameter shall be set to INVALID_PARAMETER.

Table 15—MLME-DISASSOCIATE.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, NO_ACK, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, INVALID_PARAMETER	The status of the disassociation attempt.
DeviceAddrMode	Enumeration	SHORT_ADDRESS, EXTENDED_ADDRESS	The addressing mode of the device that has either requested disassociation or been instructed to disassociate by its coordinator.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier of the device that has either requested disassociation or been instructed to disassociate by its coordinator.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter	The address of the device that has either requested disassociation or been instructed to disassociate by its coordinator.

6.2.4 Communications notification primitives

This subclause defines two primitives. The first is used to notify the next higher layer when a beacon is received during normal operating conditions. The second is used to notify the next higher layer that an error has occurred during the processing of a frame that was instigated by a response primitive.

6.2.4.1 MLME-BEACON-NOTIFY.indication

The MLME-BEACON-NOTIFY.indication primitive is used to send parameters contained within a beacon frame received by the MAC sublayer to the next higher layer when either *macAutoRequest* is set to FALSE or when the beacon frame contains one or more octets of payload. The primitive also sends a measure of the LQI and the time the beacon frame was received.

The semantics of this primitive are:

```
MLME-BEACON-NOTIFY.indication (
                              BSN,
                              PANDescriptor,
                              PendAddrSpec,
                              AddrList,
                              sduLength,
                              sdu
                              )
```

The primitive parameters are defined in Table 16.

The elements of the PANDescriptor type are defined in Table 17.

Table 16—MLME-BEACON-NOTIFY.indication parameters

Name	Type	Valid range	Description
BSN	Integer	0x00–0xff	The beacon sequence number.
PANDescriptor	PANDescriptor value	As defined in Table 17	The PANDescriptor for the received beacon.
PendAddrSpec	Bitmap	As defined in 5.2.2.1.6	The beacon pending address specification.
AddrList	List of device addresses	—	The list of addresses of the devices for which the beacon source has data.
sduLength	Integer	0 – *aMaxBeaconPayloadLength*	The number of octets contained in the beacon payload of the beacon frame received by the MAC sublayer.
sdu	Set of octets	—	The set of octets comprising the beacon payload to be transferred from the MAC sublayer entity to the next higher layer.

Table 17—Elements of PANDescriptor

Name	Type	Valid range	Description
CoordAddrMode	Enumeration	SHORT_ADDRESS, EXTENDED_ADDRESS	The coordinator addressing mode corresponding to the received beacon frame.
CoordPANId	Integer	0x0000–0xffff	The PAN identifier of the coordinator as specified in the received beacon frame.
CoordAddress	Device address	As specified by the CoordAddrMode parameter	The address of the coordinator as specified in the received beacon frame.
ChannelNumber	Integer	Any valid channel number	The current channel number occupied by the network.
ChannelPage	Integer	Any valid channel page	The current channel page occupied by the network.
SuperframeSpec	Bitmap	As defined in 5.2.2.1.2	The superframe specification as specified in the received beacon frame.
GTSPermit	Boolean	TRUE, FALSE	TRUE if the beacon is from the PAN coordinator that is accepting GTS requests.
LinkQuality	Integer	0x00–0xff	The LQI at which the network beacon was received. Lower values represent lower LQI, as defined in 8.2.6.
TimeStamp	Integer	0x000000–0xffffff	The time at which the beacon frame was received, in symbols. This value is equal to the timestamp taken when the beacon frame was received, as described in 5.1.4.1. The precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.

Table 17—Elements of PANDescriptor *(continued)*

Name	Type	Valid range	Description
SecurityStatus	Enumeration	SUCCESS, COUNTER_ERROR, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, SECURITY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY	SUCCESS if there was no error in the security processing of the frame. One of the other status codes indicating an error in the security processing otherwise, as described in 7.2.3.
SecurityLevel	Integer	As defined in Table 48	As defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	As defined in Table 48.
KeySource	Set of octets	As defined in Table 48	As defined in Table 48.
KeyIndex	Integer	As defined in Table 48	As defined in Table 48.
CodeList	List of integers	—	The list of UWB preamble codes, as described in 14.2.5.1, or CSS subchirp codes, as described in 13.3, in use when the channel was detected. For all other PHY types, this is an empty list.

6.2.4.2 MLME-COMM-STATUS.indication

The MLME-COMM-STATUS.indication primitive allows the MLME to indicate a communications status.

The semantics of this primitive are:

```
MLME-COMM-STATUS.indication (
                            PANId,
                            SrcAddrMode,
                            SrcAddr,
                            DstAddrMode,
                            DstAddr,
                            status,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

The primitive parameters are defined in Table 18.

Table 18—MLME-COMM-STATUS.indication parameters

Name	Type	Valid range	Description
PANId	Integer	0x0000–0xffff	The PAN identifier of the device from which the frame was received or to which the frame was being sent.

Table 18—MLME-COMM-STATUS.indication parameters *(continued)*

Name	Type	Valid range	Description
SrcAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The source addressing mode for this primitive.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the frame causing the error originated.
DstAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The destination addressing mode for this primitive.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the device for which the frame was intended.
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, NO_ACK, COUNTER_ERROR, FRAME_TOO_LONG, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, SECURITY_ERROR, UNAVAILABLE_KEY, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY or INVALID_PARAMETER	The communications status.
SecurityLevel	Integer	0x00–0x07	If the primitive were generated following a transmission instigated through a response primitive, then it is as defined in Table 46. If the primitive were generated on receipt of a frame that generates an error in its security processing, then it is as defined in Table 48.
KeyIdMode	Integer	As defined in Table 46 or Table 48	If the primitive were generated following a transmission instigated through a response primitive, then it is as defined in Table 46. If the primitive were generated on receipt of a frame that generates an error in its security processing, then it is as defined in Table 48.
KeySource	Set of octets	As defined in Table 46 or Table 48	If the primitive were generated following a transmission instigated through a response primitive, then it is as defined in Table 46. If the primitive were generated on receipt of a frame that generates an error in its security processing, then it is as defined in Table 48.
KeyIndex	Integer	As defined in Table 46 or Table 48	If the primitive were generated following a transmission instigated through a response primitive, then it is as defined in Table 46. If the primitive were generated on receipt of a frame that generates an error in its security processing, then it is as defined in Table 48.

The MLME-COMM-STATUS.indication primitive is generated by the MLME and issued to its next higher layer either following a transmission instigated through a response primitive or on receipt of a frame that generates an error in its security processing, as described in 7.2.3. If the request to transmit was successful, the status parameter shall be set to SUCCESS. Otherwise, the status parameter shall be set to indicate the error with one of the following values:

— TRANSACTION_OVERFLOW – The MLME does not have the capacity to store the frame that was to be sent using indirect transmission.

— TRANSACTION_EXPIRED – The transaction was not handled within *macTransactionPersistenceTime*.

— CHANNEL_ACCESS_FAILURE – There was a failure in the CSMA-CA algorithm while attempting to send the frame.

— NO_ACK – An acknowledgment was expected but not received.

— INVALID_PARAMETER – One or more of the parameters in the response primitive were in error.

— A security error, as defined in 7.2.

6.2.5 Primitives for reading PIB attributes

These primitives are used to read values from the PIB.

6.2.5.1 MLME-GET.request

The MLME-GET.request primitive requests information about a given PIB attribute.

The semantics of this primitive are:

```
MLME-GET.request        (
                        PIBAttribute
                        )
```

The primitive parameters are defined in Table 19.

Table 19—MLME-GET.request parameters

Name	Type	Valid range	Description
PIBAttribute	Octet string	As defined in Table 52, Table 60, or Table 71	The name of the PIB attribute to read.

On receipt of the MLME-GET.request primitive, the MLME checks to see whether the PIB attribute is a MAC PIB attribute or PHY PIB attribute. If the requested attribute is a MAC attribute, the MLME attempts to retrieve the requested MAC PIB attribute from its database. If the requested attribute is a PHY PIB attribute, the MLME attempts to retrieve the value from the PHY.

6.2.5.2 MLME-GET.confirm

The MLME-GET.confirm primitive reports the results of an information request from the PIB.

The semantics of this primitive are:

```
MLME-GET.confirm        (
                        status,
                        PIBAttribute,
                        PIBAttributeValue
                        )
```

The primitive parameters are defined in Table 20.

Table 20—MLME-GET.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, UNSUPPORTED_ATTRIBUTE	The result of the request for PIB attribute information.
PIBAttribute	Octet string	As defined in Table 52, Table 60, or Table 71	The name of the PIB attribute that was read.
PIBAttributeValue	Various	Attribute specific; as defined in Table 52, Table 60, or Table 71	The value of the indicated PIB attribute that was read. This parameter has zero length when the status parameter is set to UNSUPPORTED_ATTRIBUTE.

If the request to read a PIB attribute was successful, the primitive returns with a status of SUCCESS. If the identifier of the PIB attribute is not found, the primitive returns with a status of UNSUPPORTED_ATTRIBUTE. When an error code of UNSUPPORTED_ATTRIBUTE is returned, the PIBAttribute value parameter will be set to length zero.

6.2.6 GTS management primitives

These primitives are used to request and maintain GTSs.

6.2.6.1 MLME-GTS.request

The MLME-GTS.request primitive allows a device to send a request to the PAN coordinator to allocate a new GTS or to deallocate an existing GTS. This primitive is also used by the PAN coordinator to initiate a GTS deallocation.

The semantics of this primitive are:

```
MLME-GTS.request        (
                        GTSCharacteristics,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

The primitive parameters are defined in Table 21.

Table 21—MLME-GTS.request parameters

Name	Type	Valid range	Description
GTSCharacteristics	GTS characteristics	As defined in 5.3.9.2	The characteristics of the GTS request, including whether the request is for the allocation of a new GTS or the deallocation of an existing GTS.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

On receipt of the MLME-GTS.request primitive by a device, the MLME of a device performs either the GTS request procedure,as described in 5.1.7.2, or the GTS deallocation procedure, as described in 5.1.7.4, depending on the value of the GTSCharacteristics field.

6.2.6.2 MLME-GTS.confirm

The MLME-GTS.confirm primitive reports the results of a request to allocate a new GTS or to deallocate an existing GTS.

The semantics of this primitive are:

```
MLME-GTS.confirm        (
                        GTSCharacteristics,
                        status
                        )
```

The primitive parameters are defined in Table 22.

Table 22—MLME-GTS.confirm parameters

Name	Type	Valid range	Description
GTSCharacteristics	GTS characteristics	As defined in 5.3.9.2	The characteristics of the GTS.
status	Enumeration	SUCCESS, DENIED, NO_SHORT_ADDRESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, INVALID_PARAMETER.	The status of the GTS request.

If the request to allocate or deallocate a GTS was successful, this primitive will return a status of SUCCESS and the Characteristics Type field of the GTSCharacteristics parameter will have the value of one or zero, respectively. Otherwise, the status parameter will indicate the appropriate error code, as defined in 5.1.7.2 or 5.1.7.4.

If *macShortAddress* is equal to 0xfffe or 0xffff, the device is not permitted to request a GTS and the status parameter will be set to NO_SHORT_ADDRESS.

6.2.6.3 MLME-GTS.indication

The MLME-GTS.indication primitive indicates that a GTS has been allocated or that a previously allocated GTS has been deallocated.

The semantics of this primitive are:

```
MLME-GTS.indication     (
                        DeviceAddress,
                        GTSCharacteristics,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

The primitive parameters are defined in Table 23.

Table 23—MLME-GTS.indication parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	0x0000–0xfffd	The short address of the device that has been allocated or deallocated a GTS.
GTSCharacteristics	GTS characteristics	As defined in 5.3.9.2	The characteristics of the GTS.
SecurityLevel	Integer	As defined in Table 48	If the primitive were generated when a GTS deallocation is initiated by the PAN coordinator itself, the security level to be used is set to 0x00. If the primitive were generated whenever a GTS is allocated or deallocated following the reception of a GTS request command, then it is as defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	If the primitive were generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive were generated whenever a GTS is allocated or deallocated following the reception of a GTS request command, then it is as defined in Table 48.

Table 23—MLME-GTS.indication parameters *(continued)*

Name	Type	Valid range	Description
KeySource	Set of octets	As defined in Table 48	If the primitive were generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive were generated whenever a GTS is allocated or deallocated following the reception of a GTS request command, then it is as defined in Table 48.
KeyIndex	Integer	As defined in Table 48	If the primitive were generated when a GTS deallocation is initiated by the PAN coordinator itself, this parameter is ignored. If the primitive were generated whenever a GTS is allocated or deallocated following the reception of a GTS request command, then it is as defined in Table 48.

The value of the Characteristics Type field, as defined in 5.3.9.2, in the GTSCharacteristics parameter indicates if the GTS has been allocated or if a GTS has been deallocated.

6.2.7 Primitives for orphan notification

These primitives are used by a coordinator to issue a notification of an orphaned device.

6.2.7.1 MLME-ORPHAN.indication

The MLME-ORPHAN.indication primitive is generated by the MLME of a coordinator and issued to its next higher layer on receipt of an orphan notification command, as defined in 5.3.6.

The semantics of this primitive are:

```
MLME-ORPHAN.indication      (
                            OrphanAddress,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

The primitive parameters are defined in Table 24.

6.2.7.2 MLME-ORPHAN.response

The MLME-ORPHAN.response primitive allows the next higher layer of a coordinator to respond to the MLME-ORPHAN.indication primitive.

Table 24—MLME-ORPHAN.indication parameters

Name	Type	Valid range	Description
OrphanAddress	Device address	Extended IEEE address	The address of the orphaned device.
SecurityLevel	Integer	As defined in Table 48	As defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	As defined in Table 48.
KeySource	Set of octets	As defined in Table 48	As defined in Table 48.
KeyIndex	Integer	As defined in Table 48	As defined in Table 48.

The semantics of this primitive are:

```
MLME-ORPHAN.response        (
                            OrphanAddress,
                            ShortAddress,
                            AssociatedMember,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

The primitive parameters are defined in Table 25.

Table 25—MLME-ORPHAN.response parameters

Name	Type	Valid range	Description
OrphanAddress	Device address	Extended IEEE address	The address of the orphaned device.
ShortAddress	Integer	0x0000–0xffff	The short address allocated to the orphaned device if it is associated with this coordinator. The special short address 0xfffe indicates that no short address was allocated, and the device will use its extended address in all communications. If the device was not associated with this coordinator, this field will contain the value 0xffff and be ignored on receipt.
AssociatedMember	Boolean	TRUE or FALSE	TRUE if the orphaned device is associated with this coordinator or FALSE otherwise.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.

Table 25—MLME-ORPHAN.response parameters *(continued)*

Name	Type	Valid range	Description
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

If the AssociatedMember parameter is set to TRUE, the orphaned device is associated with the coordinator. In this case, the MLME generates and sends the coordinator realignment command, as defined in 5.3.8, to the orphaned device containing the value of the ShortAddress field. This command is sent in the CAP if the coordinator is on a beacon-enabled PAN or immediately otherwise. If the AssociatedMember parameter is set to FALSE, the orphaned device is not associated with the coordinator and this primitive will be ignored. If the orphaned device does not receive the coordinator realignment command following its orphan notification within *macResponseWaitTime*, it will assume it is not associated to any coordinator in range.

If the frame was successfully transmitted and an acknowledgment was received, if requested, the MAC sublayer will issue the MLME-COMM-STATUS.indication primitive with a status of SUCCESS.

6.2.8 Primitives for resetting the MAC sublayer

These primitives are used to reset the MAC sublayer.

6.2.8.1 MLME-RESET.request

The MLME-RESET.request primitive is used by the next higher layer to request that the MLME performs a reset operation.

The semantics of this primitive are:

```
MLME-RESET.request      (
                        SetDefaultPIB
                        )
```

The primitive parameters are defined in Table 26.

Table 26—MLME-RESET.request parameter

Name	Type	Valid range	Description
SetDefaultPIB	Boolean	TRUE, FALSE	If TRUE, the MAC sublayer is reset, and all MAC PIB attributes are set to their default values. If FALSE, the MAC sublayer is reset, but all MAC PIB attributes retain their values prior to the generation of the MLME-RESET.request primitive.

On receipt of the MLME-RESET.request primitive, the MLME resets the PHY in an implementation-dependent manner.

6.2.8.2 MLME-RESET.confirm

The MLME-RESET.confirm primitive reports the results of the reset operation.

The semantics of this primitive are:

```
MLME-RESET.confirm      (
                        status
                        )
```

The primitive parameter is defined in Table 27.

Table 27—MLME-RESET.confirm parameter

Name	Type	Valid range	Description
status	Enumeration	SUCCESS	The result of the reset operation.

The status parameter is set to SUCCESS on completion of the reset procedure.

6.2.9 Primitives for specifying the receiver enable time

These primitives are used to enable or disable a device's receiver at a given time.

6.2.9.1 MLME-RX-ENABLE.request

The MLME-RX-ENABLE.request primitive allows the next higher layer to request that the receiver is either enabled for a finite period of time or disabled.

The semantics of this primitive are:

```
MLME-RX-ENABLE.request      (
                            DeferPermit,
                            RxOnTime,
                            RxOnDuration,
                            RangingRxControl
                            )
```

The primitive parameters are defined in Table 28.

The MLME-RX-ENABLE.request primitive is generated by the next higher layer and issued to the MLME to enable the receiver for a fixed duration, at a time relative to the start of the current or next superframe on a beacon-enabled PAN or immediately on a nonbeacon-enabled PAN. This primitive may also be generated to cancel a previously generated request to enable the receiver. The receiver is enabled or disabled exactly once per primitive request.

The MLME will treat the request to enable or disable the receiver as secondary to other responsibilities of the device (e.g., GTSs, coordinator beacon tracking, or beacon transmissions). When the primitive is issued to enable the receiver, the device will enable its receiver until either the device has a conflicting responsibility or the time specified by RxOnDuration has expired. In the case of a conflicting responsibility, the device will interrupt the receive operation. After the completion of the interrupting operation, the RxOnDuration will be checked to determine whether the time has expired. If so, the operation is complete. If

Table 28—MLME-RX-ENABLE.request parameters

Name	Type	Valid range	Description
DeferPermit	Boolean	TRUE, FALSE	TRUE if the requested operation can be deferred until the next superframe if the requested time has already passed. FALSE if the requested operation is only to be attempted in the current superframe. This parameter is ignored for nonbeacon-enabled PANs. If the issuing device is the PAN coordinator, the term *superframe* refers to its own superframe. Otherwise, the term refers to the superframe of the coordinator through which the issuing device is associated.
RxOnTime	Integer	0x000000–0xffffff	The number of symbols measured from the start of the superframe before the receiver is to be enabled or disabled. This is a 24-bit value, and the precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant. This parameter is ignored for nonbeacon-enabled PANs. If the issuing device is the PAN coordinator, the term *superframe* refers to its own superframe. Otherwise, the term refers to the superframe of the coordinator through which the issuing device is associated.
RxOnDuration	Integer	0x000000–0xffffff	The number of symbols for which the receiver is to be enabled. If this parameter is equal to 0x000000, the receiver is to be disabled.
RangingRx Control	Enumeration	RANGING_OFF, RANGING_ON	Configure the transceiver to Rx with ranging for a value of RANGING_ON or to not enable ranging for RANGING_OFF.

not, the receiver is re-enabled until either the device has another conflicting responsibility or the time specified by RxOnDuration has expired. When the primitive is issued to disable the receiver, the device will disable its receiver unless the device has a conflicting responsibility.

On a nonbeacon-enabled PAN, the MLME ignores the DeferPermit and RxOnTime parameters and requests that the PHY enable or disable the receiver immediately. If the request is to enable the receiver, the receiver will remain enabled until RxOnDuration has elapsed.

Before attempting to enable the receiver on a beacon-enabled PAN, the MLME first determines whether (RxOnTime + RxOnDuration) is less than the beacon interval, as defined by *macBeaconOrder*. If (RxOnTime + RxOnDuration) is not less than the beacon interval, the MLME issues the MLME-RX-ENABLE.confirm primitive with a status of ON_TIME_TOO_LONG.

The MLME then determines whether the receiver can be enabled in the current superframe. If the current time measured from the start of the superframe is less than (RxOnTime – *macSIFSPeriod*), the MLME attempts to enable the receiver in the current superframe. If the current time measured from the start of the superframe is greater than or equal to (RxOnTime – *macSIFSPeriod*) and DeferPermit is equal to TRUE, the MLME defers until the next superframe and attempts to enable the receiver in that superframe. Otherwise, if the MLME cannot enable the receiver in the current superframe and is not permitted to defer the receive operation until the next superframe, the MLME issues the MLME-RX-ENABLE.confirm primitive with a status of PAST_TIME.

If the RxOnDuration parameter is equal to zero, the MLME requests that the PHY disable its receiver.

6.2.9.2 MLME-RX-ENABLE.confirm

The MLME-RX-ENABLE.confirm primitive reports the results of the attempt to enable or disable the receiver.

The semantics of this primitive are:

```
MLME-RX-ENABLE.confirm      (
                            status
                            )
```

The primitive parameters are defined in Table 29.

Table 29—MLME-RX-ENABLE.confirm parameter

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, PAST_TIME, ON_TIME_TOO_LONG, INVALID_PARAMETER, RANGING_NOT_SUPPORTED	The result of the request to enable or disable the receiver.

The MLME-RX-ENABLE.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-RX-ENABLE.request primitive. This primitive returns a status of either SUCCESS, if the request to enable or disable the receiver was successful, or the appropriate error code. The status values are fully described in 6.2.9.1.

6.2.10 Primitives for channel scanning

These primitives are used to either find PANs in a channel or measure the energy in the channel.

6.2.10.1 MLME-SCAN.request

The MLME-SCAN.request primitive is used to initiate a channel scan over a given list of channels.

The semantics of this primitive are:

```
MLME-SCAN.request           (
                            ScanType,
                            ScanChannels,
                            ScanDuration,
                            ChannelPage,
                            SecurityLevel,
                            KeyIdMode,
                            KeySource,
                            KeyIndex
                            )
```

The primitive parameters are defined in Table 30.

When the MLME receives this primitive, it begins the appropriate scan procedure, as defined in 5.1.2.

Table 30—MLME-SCAN.request parameters

Name	Type	Valid range	Description
ScanType	Enumera-tion	ED, ACTIVE, PASSIVE, ORPHAN	Indicates the type of scan performed, as described in 5.1.2.1.
ScanChannels	List of integers	Any list of valid channel numbers	The channel numbers to be scanned.
ScanDuration	Integer	0–14	A value used to calculate the length of time to spend scanning each channel for ED, active, and passive scans. This parameter is ignored for orphan scans. The time spent scanning each channel is [*aBaseSuperframeDuration* × (2^n + 1)], where *n* is the value of the ScanDuration parameter.
ChannelPage	Integer	Any valid channel page	The channel page on which to perform the scan.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

The SecurityLevel, KeyIdMode, KeySource, and KeyIndex parameters are used only in an orphan scan.

6.2.10.2 MLME-SCAN.confirm

The MLME-SCAN.confirm primitive reports the result of the channel scan request.

The semantics of this primitive are:

```
MLME-SCAN.confirm       (
                        status,
                        ScanType,
                        ChannelPage,
                        UnscannedChannels,
                        ResultListSize,
                        EnergyDetectList,
                        PANDescriptorList,
                        DetectedCategory
                        UWBEnergyDetectList
                        )
```

The primitive parameters are defined in Table 31.

If the requested scan was successful, the status parameter will be set to SUCCESS.

If the MLME receives the MLME-SCAN.request primitive while performing a previously initiated scan operation, the MLME will not perform the scan and the status parameter will be set to SCAN_IN_PROGRESS.

Table 31—MLME-SCAN.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, LIMIT_REACHED, NO_BEACON, SCAN_IN_PROGRESS, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, INVALID_PARAMETER	The status of the scan request.
ScanType	Integer	0x00–0x03	Indicates the type of scan performed: 0x00 = ED scan (optional for RFD). 0x01 = active scan (optional for RFD). 0x02 = passive scan. 0x03 = orphan scan.
ChannelPage	Integer	Any valid channel page	The channel page on which the scan was performed, as defined in 8.1.2.
UnscannedChannels	List of integers	Any list of valid channels	A list of the channels given in the request which were not scanned. This parameter is not valid for ED scans.
ResultListSize	Integer	Implementation specific	The number of elements returned in the appropriate result lists. This value is zero for the result of an orphan scan.
EnergyDetectList	List of integers	0x00–0xff for each integer	The list of energy measurements, one for each channel searched during an ED scan. This parameter is null for active, passive, and orphan scans.
PANDescriptorList	Set of PAN descriptor values	As defined in Table 17	The list of PAN descriptors, one for each beacon found during an active or passive scan if *macAutoRequest* is set to TRUE. This parameter is null for ED and orphan scans or when *macAutoRequest* is set to FALSE during an active or passive scan.
DetectedCategory	Integer	0x00–0xff	Categorization of energy detected in channel with the following values: 0: Category detection is not supported 1: UWB PHY detected 2: Non-UWB PHY signal source detected 3–25: Reserved for future use
UWBEnergy DetectList	List of Integers	0x00–0xff for each element of the list	For UWB PHYs, the list of energy measurements taken. The total number of measurements is indicated by ResultListSize. This parameter is null for active, passive, and orphan scans. It is also null for non-UWB PHYs.

If, during an active scan, the MLME is unable to transmit a beacon request command on a channel specified by the ScanChannels parameter due to a channel access failure, the channel will appear in the list of unscanned channels returned by the MLME-SCAN.confirm primitive. If the MLME was able to send a beacon request command on at least one of the channels but no beacons were found, the MLME-

SCAN.confirm primitive will contain a null set of PAN descriptor values, regardless of the value of *macAutoRequest*, and a status of NO_BEACON.

If the MLME-SCAN.request primitive requested an orphan scan, the ResultListSize parameter will be set to zero. If the MLME is unable to transmit an orphan notification command on a channel specified by the ScanChannels parameter due to a channel access failure, the channel will appear in the list of unscanned channels returned by the MLME-SCAN.confirm primitive. If the MLME was able to send an orphan notification command on at least one of the channels but the device did not receive a coordinator realignment command, the MLME-SCAN.confirm primitive will contain a status of NO_BEACON.

If the MLME-SCAN.request primitive requested an active, passive, or orphan scan, the EnergyDetectList and UWBEnergyDetectList parameters will be null. If the MLME-SCAN.request primitive requested an ED or orphan scan, the PANDescriptorList parameter will be null.

If, during an ED, active, or passive scan, the implementation-specified maximum of PAN descriptors is reached thus terminating the scan procedure, the MAC sublayer will issue the MLME-SCAN.confirm primitive with a status of LIMIT_REACHED.

If the MLME-SCAN.request primitive requested an ED and the PHY type is UWB, as indicated by the *phyChannelPage*, then the UWBEnergyDetectList contains the results for the UWB channels scanned, and the EnergyDetectList and PANDescriptorList are null. The UWB scan is fully described in 5.1.2.1.

6.2.11 Primitives for writing PIB attributes

These primitives are used to write PIB attributes.

6.2.11.1 MLME-SET.request

The MLME-SET.request primitive attempts to write the given value to the indicated PIB attribute.

The semantics of this primitive are:

```
MLME-SET.request        (
                        PIBAttribute,
                        PIBAttributeValue
                        )
```

The primitive parameters are defined in Table 32.

Table 32—MLME-SET.request parameters

Name	Type	Valid range	Description
PIBAttribute	Octet string	As defined in Table 52, Table 60, or Table 71	The name of the PIB attribute to write.
PIBAttributeValue	Various	Attribute specific; as defined in Table 52, Table 60, or Table 71	The value to write to the indicated PIB attribute.

On receipt of the MLME-SET.request primitive, the MLME checks to see whether the PIB attribute is a MAC PIB attribute or a PHY PIB attribute. If the requested attribute is a MAC attribute, the MLME

attempts to write the given value to the indicated MAC PIB attribute. If the requested attribute is a PHY attribute, the MLME attempts to write the given value to the indicated PHY PIB attribute.

6.2.11.2 MLME-SET.confirm

The MLME-SET.confirm primitive reports the results of an attempt to write a value to a PIB attribute.

The semantics of this primitive are:

```
MLME-SET.confirm        (
                        status,
                        PIBAttribute,
                        )
```

The primitive parameters are defined in Table 33.

Table 33—MLME-SET.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, READ_ONLY, UNSUPPORTED_ATTRIBUTE, INVALID_INDEX, INVALID_PARAMETER	The result of the request to write the PIB attribute.
PIBAttribute	Octet string	As defined in Table 52, Table 60, or Table 71	The name of the PIB attribute that was written.

The MLME-SET.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-SET.request primitive. The MLME-SET.confirm primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated PIB attribute or with the status parameter set as follows:

— READ_ONLY – The PIBAttribute parameter specifies an attribute that is a read-only attribute.

— UNSUPPORTED_ATTRIBUTE – The PIBAttribute parameter specifies an attribute that was not found in the database.

— INVALID_PARAMETER– The PIBAttributeValue parameter specifies a value that is out of the valid range for the given attribute.

If the PIBAttribute parameter indicates that *macBeaconPayloadLength* is to be set and the length of the resulting beacon frame exceeds *aMaxPHYPacketSize* (e.g., due to the additional overhead required for security processing), the MAC sublayer shall not update *macBeaconPayloadLength* and will issue the MLME-GET.confirm primitive with a status of INVALID_PARAMETER.

6.2.12 Primitives for updating the superframe configuration

These primitives are used by an FFD to initiate a PAN, to begin using a new superframe configuration, or to stop transmitting beacons. In addition, a device uses these primitives to begin using a new superframe configuration.

6.2.12.1 MLME-START.request

The MLME-START.request primitive is used by the PAN coordinator to initiate a new PAN or to begin using a new superframe configuration. This primitive is also used by a device already associated with an existing PAN to begin using a new superframe configuration.

The semantics of this primitive are:

```
MLME-START.request      (
                        PANId,
                        ChannelNumber,
                        ChannelPage,
                        StartTime,
                        BeaconOrder,
                        SuperframeOrder,
                        PANCoordinator,
                        BatteryLifeExtension,
                        CoordRealignment,
                        CoordRealignSecurityLevel,
                        CoordRealignKeyIdMode,
                        CoordRealignKeySource,
                        CoordRealignKeyIndex,
                        BeaconSecurityLevel,
                        BeaconKeyIdMode,
                        BeaconKeySource,
                        BeaconKeyIndex
                        )
```

The primitive parameters are defined in Table 34.

Table 34—MLME-START.request parameters

Name	Type	Valid range	Description
PANId	Integer	0x0000–0xffff	The PAN identifier to be used by the device.
ChannelNumber	Integer	Any valid channel number	The channel number to use.
ChannelPage	Integer	Any valid channel page	The channel page to use.
StartTime	Integer	0x000000–0xffffff	The time at which to begin transmitting beacons. If this parameter is equal to 0x000000, beacon transmissions will begin immediately. Otherwise, the specified time is relative to the received beacon of the coordinator with which the device synchronizes. This parameter is ignored if either the BeaconOrder parameter has a value of 15 or the PANCoordinator parameter is TRUE. The time is specified in symbols and is rounded to a backoff period boundary. The precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.

Table 34—MLME-START.request parameters *(continued)*

Name	Type	Valid range	Description
BeaconOrder	Integer	0–15	Indicates the frequency with which the beacon is transmitted, as defined in 5.1.1.1.
SuperframeOrder	Integer	0–*BO* or 15	The length of the active portion of the superframe, including the beacon frame, as defined in 5.1.1.1.
PANCoordinator	Boolean	TRUE, FALSE	If this value is TRUE, the device will become the PAN coordinator of a new PAN. If this value is FALSE, the device will begin using a new superframe configuration on the PAN with which it is associated.
BatteryLifeExtension	Boolean	TRUE, FALSE	If this value is TRUE, the receiver of the beaconing device is disabled *macBattLifeExtPeriods* full backoff periods after the interframe spacing (IFS) period following the beacon frame. If this value is FALSE, the receiver of the beaconing device remains enabled for the entire CAP. This parameter is ignored if the BeaconOrder parameter has a value of 15.
CoordRealignment	Boolean	TRUE, FALSE	TRUE if a coordinator realignment command is to be transmitted prior to changing the superframe configuration or FALSE otherwise.
CoordRealignSecurity Level	Integer	0x00–0x07	The security level to be used for coordinator realignment command frames, as described in Table 58.
CoordRealignKeyId Mode	Integer	0x00–0x03	The mode used to identify the key to be used, as described in 7.4.1.2. This parameter is ignored if the CoordRealignSecurityLevel parameter is set to 0x00.
CoordRealignKey Source	Set of octets	As specified by the CoordRealignKeyIdMode parameter	The originator of the key to be used, as described in 7.4.3.1. This parameter is ignored if the CoordRealignKeyIdMode parameter is ignored or is set to 0x00.
CoordRealignKeyIndex	Integer	0x01–0xff	The index of the key to be used, as described in 7.4.3.2. This parameter is ignored if the CoordRealignKeyIdMode parameter is ignored or is set to 0x00.
BeaconSecurityLevel	Integer	0x00–0x07	The security level to be used for beacon frames, as described in Table 58.
BeaconKeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used, as described in Table 59. This parameter is ignored if the BeaconSecurityLevel parameter is set to 0x00.

Table 34—MLME-START.request parameters *(continued)*

Name	Type	Valid range	Description
BeaconKeySource	Set of octets	As specified by the BeaconKeyIdMode parameter	The originator of the key to be used, as described in 7.4.3.1. This parameter is ignored if the BeaconKeyIdMode parameter is ignored or set to 0x00.
BeaconKeyIndex	Integer	0x01–0xff	The index of the key to be used, as described in 7.4.3.2. This parameter is ignored if the BeaconKeyIdMode parameter is ignored or set to 0x00.

When the CoordRealignment parameter is set to TRUE, the coordinator attempts to transmit a coordinator realignment command frame as described in 5.1.2.3.2. If the transmission of the coordinator realignment command fails due to a channel access failure, the MLME will not make any changes to the superframe configuration. (i.e., no PIB attributes will be changed). If the coordinator realignment command is successfully transmitted, the MLME updates the PIB attributes BeaconOrder, SuperframeOrder, PANId, ChannelPage, and ChannelNumber parameters.

When the CoordRealignment parameter is set to FALSE, the MLME updates the appropriate PIB attributes with the values of the BeaconOrder, SuperframeOrder, PANId, ChannelPage, and ChannelNumber parameters, as described in 5.1.2.3.4.

The address used by the coordinator in its beacon frames is determined by the current value of *macShortAddress*, which is set by the next higher layer before issuing this primitive. If the BeaconOrder parameter is less than 15, the MLME sets *macBattLifeExt* to the value of the BatteryLifeExtension parameter. If the BeaconOrder parameter equals 15, the value of the BatteryLifeExtension parameter is ignored.

If the CoordRealignment parameter is set to TRUE, the CoordRealignSecurityLevel, CoordRealignKeyIdMode, CoordRealignKeySource, and CoordRealignKeyIndex parameters will be used to process the MAC command frame. If the BeaconOrder parameter indicates a beacon-enabled network, the BeaconSecurityLevel, BeaconKeyIdMode, BeaconKeySource, and BeaconKeyIndex parameters will be used to process the beacon frame.

The MLME shall ignore the StartTime parameter if the BeaconOrder parameter is equal to 15 because this indicates a nonbeacon-enabled PAN. If the BeaconOrder parameter is less than 15, the MLME examines the StartTime parameter to determine the time to begin transmitting beacons. If the PAN coordinator parameter is set to TRUE, the MLME ignores the StartTime parameter and begins beacon transmissions immediately. Setting the StartTime parameter to 0x000000 also causes the MLME to begin beacon transmissions immediately. If the PANCoordinator parameter is set to FALSE and the StartTime parameter is nonzero, the MLME calculates the beacon transmission time by adding StartTime to the time, obtained from the local clock, when the MLME receives the beacon of the coordinator through which it is associated. If the time calculated causes the outgoing superframe to overlap the incoming superframe, the MLME shall not begin beacon transmissions. Otherwise, the MLME then begins beacon transmissions when the current time, obtained from the local clock, equals the calculated time.

6.2.12.2 MLME-START.confirm

The MLME-START.confirm primitive reports the results of the attempt to start using a new superframe configuration.

The semantics of this primitive are:

```
MLME-START.confirm      (
                        status
                        )
```

The primitive parameters are defined in Table 35.

Table 35—MLME-START.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, NO_SHORT_ADDRESS, SUPERFRAME_OVERLAP, TRACKING_OFF, INVALID_PARAMETER, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, CHANNEL_ACCESS_FAILURE	The result of the attempt to start using an updated superframe configuration.

The MLME-START.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-START.request primitive. The MLME-START.confirm primitive returns a status of either SUCCESS, indicating that the MAC sublayer has started using the new superframe configuration, or the appropriate error code as follows:

— NO_SHORT_ADDRESS – The *macShortAddress* is set to 0xffff.

— CHANNEL_ACCESS_FAILURE – The transmission of the coordinator realignment frame failed.

— FRAME_TOO_LONG – The length of the beacon frame exceeds *aMaxPHYPacketSize*.

— SUPERFRAME_OVERLAP – The outgoing superframe overlaps the incoming superframe.

— TRACKING_OFF – The StartTime parameter is nonzero, and the MLME is not currently tracking the beacon of the coordinator through which it is associated.

— A security error code, as defined in 7.2.

6.2.13 Primitives for synchronizing with a coordinator

These primitives are used to synchronize with a coordinator and to communicate loss of synchronization to the next higher layer.

6.2.13.1 MLME-SYNC.request

The MLME-SYNC.request primitive requests to synchronize with the coordinator by acquiring and, if specified, tracking its beacons.

The semantics of this primitive are:

```
MLME-SYNC.request        (
                         ChannelNumber,
                         ChannelPage,
                         TrackBeacon
                         )
```

The primitive parameters are defined in Table 36.

Table 36—MLME-SYNC.request parameters

Name	Type	Valid range	Description
ChannelNumber	Integer	Any valid channel number	The channel number on which to attempt coordinator synchronization.
ChannelPage	Integer	Any valid channel page	The channel page on which to attempt coordinator synchronization.
TrackBeacon	Boolean	TRUE, FALSE	TRUE if the MLME is to synchronize with the next beacon and attempts to track all future beacons. FALSE if the MLME is to synchronize with only the next beacon.

If the MLME-SYNC.request primitive is received by the MLME on a beacon-enabled PAN, it will first set *phyCurrentPage* and *phyCurrentChannel* equal to the values of the ChannelPage and ChannelNumber parameters, respectively. If the TrackBeacon parameter is equal to TRUE, the MLME will track the beacon, i.e., enable its receiver just before the expected time of each beacon so that the beacon frame can be processed. If the TrackBeacon parameter is equal to FALSE, the MLME will locate the beacon but not continue to track it.

If this primitive is received by the MLME while it is currently tracking the beacon, the MLME will not discard the primitive but will treat it as a new synchronization request.

6.2.13.2 MLME-SYNC-LOSS.indication

The MLME-SYNC-LOSS.indication primitive indicates the loss of synchronization with a coordinator.

The semantics of this primitive are:

```
MLME-SYNC-LOSS.indication    (
                             LossReason,
                             PANId,
                             ChannelNumber,
                             ChannelPage,
                             SecurityLevel,
                             KeyIdMode,
                             KeySource,
                             KeyIndex
                             )
```

The primitive parameters are defined in Table 37.

Table 37—MLME-SYNC-LOSS.indication parameters

Name	Type	Valid range	Description
LossReason	Enumeration	PAN_ID_CONFLICT, REALIGNMENT, BEACON_LOST, SUPERFRAME_OVERLAP	The reason that synchronization was lost.
PANId	Integer	0x0000–0xffff	The PAN identifier with which the device lost synchronization or to which it was realigned.
ChannelNumber	Integer	Any valid channel	The channel number on which the device lost synchronization or to which it was realigned.
ChannelPage	Integer	Any valid channel page	The channel page on which the device lost synchronization or to which it was realigned.
SecurityLevel	Integer	As defined in Table 48	If the primitive were either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, the security level is set to 0x00. If the primitive were generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command, then it is as defined in Table 48.
KeyIdMode	Integer	As defined in Table 48	If the primitive were either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive were generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command, then it is as defined in Table 48.
KeySource	Set of octets	As defined in Table 48	If the primitive were either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive were generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command, then it is as defined in Table 48.
KeyIndex	Integer	As defined in Table 48	If the primitive were either generated by the device itself following loss of synchronization or generated by the PAN coordinator upon detection of a PAN ID conflict, this parameter is ignored. If the primitive were generated following the reception of either a coordinator realignment command or a PAN ID conflict notification command, then it is as defined in Table 48.

The MLME-SYNC-LOSS.indication primitive is generated by the MLME of a device and issued to its next higher layer in the event of a loss of synchronization with the coordinator. It is also generated by the MLME of the PAN coordinator and issued to its next higher layer in the event of either a PAN ID conflict or an overlap between the outgoing superframe and the incoming superframe, as described in 5.1.1.2.

The LossReason parameter indicates the reason why the primitive was issued. Values for the LossReason parameter are:

— PAN_ID_CONFLICT – The device has detected a PAN identifier conflict and has communicated it to the PAN coordinator or the PAN coordinator has received a PAN ID conflict notification command regarding a device that is associated with it, as described in 5.1.2.2.
— REALIGNMENT – The device has received the coordinator realignment command from the coordinator through which it is associated and the MLME was not carrying out an orphan scan, as described in 5.1.2.3.3.
— BEACON_LOST – The device has missed too many beacons, as described in 5.1.4.1.
— SUPERFRAME_OVERLAP – The device has received a coordinator realignment comment from its coordinator that would cause the incoming and outgoing superframes to overlap, as described in 5.1.1.2.

6.2.14 Primitives for requesting data from a coordinator

These primitives are used to request data from a coordinator.

6.2.14.1 MLME-POLL.request

The MLME-POLL.request primitive prompts the device to request data from the coordinator.

The semantics of this primitive are:

```
MLME-POLL.request       (
                        CoordAddrMode,
                        CoordPANId,
                        CoordAddress,
                        SecurityLevel,
                        KeyIdMode,
                        KeySource,
                        KeyIndex
                        )
```

The primitive parameters are defined in Table 38.

On receipt of the MLME-POLL.request primitive, the MLME requests data from the coordinator, as described in 5.1.6.3. If the poll is directed to the PAN coordinator, the data request command may be generated without any destination address information present. Otherwise, the data request command is always generated with the destination address information in the CoordPANId and CoordAddress parameters.

6.2.14.2 MLME-POLL.confirm

The MLME-POLL.confirm primitive reports the results of a request to poll the coordinator for data.

The semantics of this primitive are:

```
MLME-POLL.confirm       (
                        status
                        )
```

The primitive parameters are defined in Table 39.

Table 38—MLME-POLL.request parameters

Name	Type	Valid range	Description
CoordAddrMode	Enumeration	SHORT_ADDRESS, EXTENDED_ADDRESS	The addressing mode of the coordinator to which the poll is intended.
CoordPANId	Integer	0x0000–0xfffe	The PAN identifier of the coordinator to which the poll is intended.
CoordAddress	Device-Address	As specified by the CoordAddrMode parameter	The address of the coordinator to which the poll is intended.
SecurityLevel	Integer	As defined in Table 46	As defined in Table 46.
KeyIdMode	Integer	As defined in Table 46	As defined in Table 46.
KeySource	Set of octets	As defined in Table 46	As defined in Table 46.
KeyIndex	Integer	As defined in Table 46	As defined in Table 46.

Table 39—MLME-POLL.confirm parameters

Name	Type	Valid range	Description
status	Integer	SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, NO_DATA, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, INVALID_PARAMETER	The status of the data request.

The MLME-POLL.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-POLL.request primitive. If the request was successful, the status parameter will be equal to SUCCESS, indicating a successful poll for data. Otherwise, the status parameter indicates the appropriate error code. The status values are fully described in 6.2.14.1.

If the Frame Pending field of the acknowledgment frame is set to zero, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA.

If a frame is received from the coordinator with a zero length payload or if the frame is a MAC command frame, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA. If a frame is received from the coordinator with nonzero length payload, the MLME will issue the MLME-POLL.confirm primitive with a status of SUCCESS. In this case, the actual data are indicated to the next higher layer using the MCPS-DATA.indication primitive, as described in 6.3.3.

If a frame is not received within *macMaxFrameTotalWaitTime* even though the acknowledgment to the data request command has its Frame Pending field set to one, the MLME will issue the MLME-POLL.confirm primitive with a status of NO_DATA.

6.2.15 Primitives for specifying dynamic preamble

These primitives are used by a device to enable or disable DPS as well as to define the value of dynamic preamble for transmission and reception for a given time. DPS is only supported by the UWB PHY.

6.2.15.1 MLME-DPS.request

The MLME-DPS.request primitive allows the next higher layer to request that the PHY utilize a given pair of preamble codes for a single use pending expiration of the DPSIndexDuration.

The semantics of this primitive are:

```
MLME-DPS.request        (
                        TxDPSIndex
                        RxDPSIndex
                        DPSIndexDuration
                        )
```

The primitive parameters are defined in Table 40.

Table 40—MLME-DPS.request parameters

Name	Type	Valid range	Description
TxDPSIndex	Integer	0, 13–16, 21–24	The index value for the transmitter. A value of 0 disables the index and indicates that the *phyCurrentCode* value is to be used, as defined in 14.2.5.1. Other values indicate the preamble code, as defined in Table 103.
RxDPSIndex	Integer	0, 13–16, 21–24	The index value for the receiver. A value of 0 disables the index and indicates that the *phyCurrentCode* value is to be used, as defined in 14.2.5.1. Other values indicate the preamble code, as defined in Table 103.
DPSIndexDuration	Integer	0x000000–0xffffff	The number of symbols for which the transmitter and receiver will utilize the respective DPS indices if a MCPS-DATA.request primitive is not issued.

This primitive may also be generated to cancel a previously generated request to enable the transmitter and receiver dynamic preambles. The use of the index for the transmitter and receiver is enabled or disabled exactly once per primitive request.

The MLME starts the timer that assures that the device returns to a normal operating state with default preambles if a following MCPS-DATA.request primitive does not occur. After starting the timer, the MLME responds with a MLME-DPS.confirm primitive with the appropriate status parameter.

6.2.15.2 MLME-DPS.confirm

The MLME-DPS.confirm primitive reports the results of the attempt to enable or disable the DPS.

The semantics of this primitive are:

```
MLME-DPS.confirm        (
                        status
                        )
```

The primitive parameter is defined in Table 41.

Table 41—MLME-DPS.confirm parameter

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, DPS_NOT_SUPPORTED	The result of the request to enable or disable dynamic preambles.

The MLME-DPS.confirm primitive is generated by the MLME and issued to its next higher layer in response to an MLME-DPS.request primitive.

If any parameter in the MLME-DPS.request primitive is not supported or is out of range, the status of DPS_NOT_SUPPORTED is returned. If the request to enable or disable the DPS was successful, the MLME issues the MLME-DPS.confirm primitive with a status of SUCCESS.

6.2.15.3 MLME-DPS.indication

The MLME-DPS.indication primitive indicates the expiration of the DPSIndexDuration and the resetting of the DPS values in the PHY.

The semantics of this primitive are:

```
MLME-DPS.indication     ()
```

If a MCPS-DATA.request primitive is not received before the timer expires, the MLME issues the MLME-DPS.indication primitive to the next higher layer.

6.2.16 Primitives for channel sounding

These primitives are used to obtain the results of a channel sounding from an RDEV that supports the optional sounding capability.

6.2.16.1 MLME-SOUNDING.request

The MLME-SOUNDING.request primitive is used by the next higher layer to request that the PHY respond with channel sounding information. The MLME-SOUNDING.request primitive shall be supported by all RDEVs; however, the underlying sounding capability is optional in all cases.

The semantics of this primitive are:

```
MLME-SOUNDING.request   ()
```

If the feature is supported, the MLME will begin the sounding procedure.

6.2.16.2 MLME-SOUNDING.confirm

The MLME-CHANNEL.confirm primitive reports the result of a request to the PHY to provide channel sounding information. The MLME-SOUNDING.confirm primitive shall be supported by all RDEVs; however, the underlying sounding capability is optional in all cases.

The semantics of this primitive are:

```
MLME-SOUNDING.confirm (
                      SoundingList,
                      status
                      )
```

The primitive parameters are defined in Table 42.

Table 42—MLME-SOUNDING.confirm parameters

Name	Type	Valid range	Description
SoundingList	List of sounding points	—	Results of the sounding measurement.
status	Enumeration	SUCCESS, NO_DATA, SOUNDING_NOT_SUPPORTED	The status of the attempt to return sounding data.

The elements of a SoundingList are defined in Table 43.

Table 43—Elements of a SoundingList

Name	Type	Valid range	Description
SoundingTime	Signed integer	—	The LSB represents a nominal 16 ps.
SoundingAmplitude	Signed integer	—	A relative measurement or the received signal strength.

If the channel sounding information is available, the status parameter will be set to SUCCESS and the SoundingList will contain valid data.

If the MLME-SOUNDING.request primitive is received when there is no information present, e.g., when the PHY is in the process of performing a measurement, the status parameter will be set to NO_DATA.

If the channel sounding capability is not supported by the PHY, the status parameters will be set to UNSUPPORTED_ATTRIBUTE.

6.2.17 Primitives for ranging calibration (for UWB PHYs)

These primitives are used to obtain the results of a ranging calibration request from an RDEV.

6.2.17.1 MLME-CALIBRATE.request

The MLME-CALIBRATE.request primitive attempts to have the PHY respond with RMARKER offset information. The MLME-CALIBRATE.request primitive shall be implemented by RDEVs.

The semantics of this primitive are:

```
MLME-CALIBRATE.request          ()
```

The MLME issues the MLME-CALIBRATE.confirm primitive with the appropriate information.

6.2.17.2 MLME-CALIBRATE.confirm

The MLME-CALIBRATE.confirm primitive reports the result of a request to the PHY to provide internal propagation path information. The MLME-CALIBRATE.confirm primitive shall be implemented by RDEVs.

```
MLME-CALIBRATE.confirm          (
                                status,
                                CalTxRMARKEROffset,
                                CalRxRMARKEROffset,
                                )
```

The primitive parameters are defined in Table 44.

Table 44—MLME-CALIBRATE.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, NO_DATA, COMPUTATION_NEEDED, UNSUPPORTED_ATTRIBUTE	The status of the attempt to return sounding data.
CalTxRMARKER Offset	Unsigned Integer	0x00000000–0xffffffff	A count of the propagation time from the ranging counter to the transmit antenna. The LSB of a time value represents 1/ 128 of a chip time at the mandatory chipping rate of 499.2 MHz.
CalRxRMARKER Offset	Unsigned Integer	0x00000000–0xffffffff	A count of the propagation time from the receive antenna to the ranging counter. The LSB of a time value represents 1/ 128 of a chip time at the mandatory chipping rate of 499.2 MHz.

The MLME-CALIBRATE.confirm primitive is generated by the MLME and issued to its next higher layer in response to a MLME-CALIBRATE.request primitive.

If the feature is supported, the MLME issues the MLME-CALIBRATE.confirm primitive with a status of SUCCESS.

If the MLME-CALIBRATE.request primitive is received when there is no information present, e.g., when the PHY is in the process of performing a measurement, the status parameter will be set to NO_DATA.

If the PHY does not support autonomous self-calibration, the status parameter will be set to a value of COMPUTATION_NEEDED. This indicates to the next higher layer that it should use the sounding primitives to finish the calibration.

If the channel sounding capability is not present in the PHY, the status parameter will be set to a value of UNSUPPORTED_ATTRIBUTE.

6.3 MAC data service

The MCPS-SAP supports the transport of data. Table 45 lists the primitives supported by the MCPS-SAP. Primitives marked with a diamond (♦) are optional for an RFD. These primitives are discussed in the subclauses referenced in the table.

Table 45—MCPS-SAP primitives

MCPS-SAP primitive	Request	Confirm	Indication
MCPS-DATA	6.3.1	6.3.2	6.3.3
MCPS-PURGE	6.3.4♦	6.3.5♦	—

6.3.1 MCPS-DATA.request

The MCPS-DATA.request primitive requests the transfer of data to another device.

The semantics of this primitive are:

```
MCPS-DATA.request (
                  SrcAddrMode,
                  DstAddrMode,
                  DstPANId,
                  DstAddr,
                  msduLength,
                  msdu,
                  msduHandle,
                  AckTX,
                  GTSTX,
                  IndirectTX,
                  SecurityLevel,
                  KeyIdMode,
                  KeySource,
                  KeyIndex,
                  UWBPRF,
                  Ranging,
                  UWBPreambleSymbolRepetitions,
                  DataRate
                  )
```

The primitive parameters are defined in Table 46.

Table 46—MCPS-DATA.request parameters

Name	Type	Valid range	Description
SrcAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The source addressing mode for this MPDU.
DstAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The destination addressing mode for this MPDU.
DstPANId	Integer	0x0000–0xffff	The PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	≤ *aMaxMACPayloadSize*	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU to be transmitted by the MAC sublayer entity.
msduHandle	Integer	0x00–0xff	The handle associated with the MSDU to be transmitted by the MAC sublayer entity.
AckTX	Boolean	TRUE, FALSE	TRUE if acknowledged transmission is used, FALSE otherwise.
GTSTX	Boolean	TRUE, FALSE	TRUE if a GTS is to be used for transmission. FALSE indicates that the CAP will be used.
IndirectTX	Boolean	TRUE, FALSE	TRUE if indirect transmission is to be used, FALSE otherwise.
SecurityLevel	Integer	As defined in Table 58	The security level to be used.
KeyIdMode	Integer	As defined in Table 59	The mode used to identify the key to be used. This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of octets	As specified by the KeyIdMode parameter	The originator of the key to be used, as described in 7.4.3.1. This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used, as described in 7.4.3.2. This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
UWBPRF	Enumeration	PRF_OFF, NOMINAL_4_M, NOMINAL_16_M, NOMINAL_64_M	The pulse repetition value of the transmitted PPDU. Non-UWB PHYs use a value of PRF_OFF.
Ranging	Enumeration	NON_RANGING, ALL_RANGING, PHY_HEADER_ONLY	A value of NON_RANGING indicates that ranging is not to be used. A value of ALL_RANGING indicates that ranging operations using both the ranging bit in the PHR and the counter operation are enabled. A value of PHY_HEADER_ONLY indicates that only the ranging bit in the PHR will be used. A value of NON_RANGING is PHYs that do not support ranging.

Table 46—MCPS-DATA.request parameters *(continued)*

Name	Type	Valid range	Description
UWBPreambleSymbolRepetitions	Integer	0, 16, 64, 1024, 4096	The preamble symbol repetitions of the UWB PHY frame. A zero value is used for non-UWB PHYs.
DataRate	Integer	0–4	Indicates the data rate. For CSS PHYs, a value of one indicates 250 kb/s while a value of two indicates 1 Mb/s. For UWB PHYs, values 1–4 are valid and are defined in 14.2.6.1. For all other PHYs, the parameter is set to zero.

On receipt of the MCPS-DATA.request primitive, the MAC sublayer entity begins the transmission of the supplied MSDU.

If the msduLength parameter is greater than *aMaxMACSafePayloadSize*, the MAC sublayer will set the Frame Version field to one.

The TxOptions parameter indicates the method used by the MAC sublayer data service to transmit the supplied MSDU. If the TxOptions parameter specifies that an acknowledged transmission is required, the AR field will be set appropriately, as described in 5.1.6.4.

If the TxOptions parameter specifies that a GTS transmission is required, the MAC sublayer will determine whether it has a valid GTS as described 5.1.7.3. If a valid GTS could not be found, the MAC sublayer will discard the MSDU. If a valid GTS was found, the MAC sublayer will defer, if necessary, until the GTS. If the TxOptions parameter specifies that a GTS transmission is not required, the MAC sublayer will transmit the MSDU using either slotted CSMA-CA in the CAP for a beacon-enabled PAN or unslotted CSMA-CA for a nonbeacon-enabled PAN. Specifying a GTS transmission in the TxOptions parameter overrides an indirect transmission request.

If the TxOptions parameter specifies that an indirect transmission is required and this primitive is received by the MAC sublayer of a coordinator, the data frame is sent using indirect transmission, as described in 5.1.5 and 5.1.6.3.

If the TxOptions parameter specifies that an indirect transmission is required and if the device receiving this primitive is not a coordinator, the destination address is not present, or the TxOptions parameter also specifies a GTS transmission, the indirect transmission option will be ignored.

If the TxOptions parameter specifies that an indirect transmission is not required, the MAC sublayer will transmit the MSDU using CSMA-CA either in the CAP for a beacon-enabled PAN or immediately for a nonbeacon-enabled PAN.

6.3.2 MCPS-DATA.confirm

The MCPS-DATA.confirm primitive reports the results of a request to transfer data to another device.

The semantics of the MCPS-DATA.confirm primitive are:

```
MCPS-DATA.confirm (
                   msduHandle,
                   Timestamp,
                   RangingReceived,
                   RangingCounterStart,
                   RangingCounterStop,
                   RangingTrackingInterval,
                   RangingOffset,
                   RangingFOM,
                   status
                   )
```

The primitive parameters are defined in Table 47.

The MCPS-DATA.confirm primitive is generated by the MAC sublayer entity in response to an MCPS-DATA.request primitive. The MCPS-DATA.confirm primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, or the appropriate error code.

If both the SrcAddrMode and the DstAddrMode parameters are set to NO_ADDRESS in the MCPS-DATA.request primitive, the status shall be set to INVALID_ADDRESS.

If a valid GTS could not be found, the status shall be set to INVALID_GTS.

If there is no capacity to store the transaction, the status will be set to TRANSACTION_OVERFLOW. If the transaction is not handled within the required time, the transaction information will be discarded and the status will be set to TRANSACTION_EXPIRED.

If the TxOptions parameter specifies that a direct transmission is required and the MAC sublayer does not receive an acknowledgment from the recipient after *macMaxFrameRetries* retransmissions, as described in 5.1.6.4, it will discard the MSDU and issue the MCPS-DATA.confirm primitive with a status of NO_ACK.

If the requested transaction is too large to fit in the CAP or GTS, as appropriate, the MAC sublayer shall discard the frame and issue the MCPS-DATA.confirm primitive with a status of FRAME_TOO_LONG.

If the transmission uses CSMA-CA and the CSMA-CA algorithm failed due to adverse conditions on the channel, and the TxOptions parameter specifies that a direct transmission is required, the MAC sublayer will discard the MSDU and the status will be set to CHANNEL_ACCESS_FAILURE.

6.3.3 MCPS-DATA.indication

The MCPS-DATA.indication primitive indicates the reception of data from another device.

Table 47—MCPS-DATA.confirm parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle associated with the MSDU being confirmed.
Timestamp	Integer	0x000000–0xffffff	Optional. The time, in symbols, at which the data were transmitted, as described in 5.1.4.1. The value of this parameter will be considered valid only if the value of the status parameter is SUCCESS; if the status parameter is not equal to SUCCESS, the value of the Timestamp parameter shall not be used for any other purpose. The symbol boundary is described by *macSyncSymbolOffset*, as described in Table 52. The precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.
Ranging Received	Boolean	TRUE, FALSE	A value of FALSE indicates that ranging is either not supported by the PHY or that it was not indicated by the received PSDU. A value of TRUE indicates ranging operations were indicated for this PSDU.
Ranging CounterStart	Unsigned Integer	0x00000000–0xffffffff	A count of the time units corresponding to an RMARKER at the antenna at the beginning of a ranging exchange, as described in 14.7.1. A value of 0x00000000 is used if ranging is not supported, not enabled or if counter was not used for this PPDU.
Ranging CounterStop	Unsigned Integer	0x00000000–0xffffffff	A count of the time units corresponding to an RMARKER at the antenna at the end of a ranging exchange, as described in 14.7.1. A value of 0x00000000 is used if ranging is not supported, not enabled, or if the counter is not used for this PPDU.
Ranging Tracking Interval	Integer	0x00000000–0xffffffff	A count of the time units in a message exchange over which the tracking offset was measured, as described in 14.7.2.2. If tracking-based crystal characterization is not supported or if ranging is not supported, a value of 0x00000000 is used.
Ranging Offset	Signed Magnitude Integer	0x000000–0xffffff	A count of the time units slipped or advanced by the radio tracking system over the course of the entire tracking interval, as described in 14.7.2.1. The top 4 bits are reserved and set to zero. The most significant of the active bits is the sign bit.

Table 47—MCPS-DATA.confirm parameters *(continued)*

Name	Type	Valid range	Description
RangingFOM	Integer	0x00–0x7f	The FoM characterizing the ranging measurement, as described in 14.7.3. The most significant bit (MSB) is reserved and is zero. The remaining 7 bits are used in three fields: Confidence Level, Confidence Interval, and Confidence Interval Scaling Factor.
status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, INVALID_ADDRESS, INVALID_GTS, NO_ACK, COUNTER_ERROR, FRAME_TOO_LONG, UNAVAILABLE_KEY, UNSUPPORTED_SECURITY, INVALID_PARAMETER	The status of the last MSDU transmission.

The semantics of this primitive are:

```
MCPS-DATA.indication (
                     SrcAddrMode,
                     SrcPANId,
                     SrcAddr,
                     DstAddrMode,
                     DstPANId
                     DstAddr,
                     msduLength,
                     msdu,
                     mpduLinkQuality,
                     DSN,
                     Timestamp,
                     SecurityLevel,
                     KeyIdMode,
                     KeySource,
                     KeyIndex,
                     UWBPRF,
                     UWBPreambleSymbolRepetitions,
                     DataRate,
                     RangingReceived,
                     RangingCounterStart,
                     RangingCounterStop,
                     RangingTrackingInterval,
                     RangingOffset,
                     RangingFOM
                     )
```

The primitive parameters are defined in Table 48.

Table 48—MCPS-DATA.indication parameters

Name	Type	Valid range	Description
SrcAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The source addressing mode for this primitive corresponding to the received MPDU.
SrcPANId	Integer	0x0000–0xffff	The PAN identifier of the entity from which the MSDU was received.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the MSDU was received.
DstAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRESS	The destination addressing mode for this primitive corresponding to the received MPDU.
DstPANId	Integer	0x0000–0xffff	The PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	≤ *aMaxMACFrame-Size*	The number of octets contained in the MSDU being indicated by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU being indicated by the MAC sublayer entity.
mpduLinkQuality	Integer	0x00–0xff	LQI value measured during reception of the MPDU. Lower values represent lower LQI, as described in 8.2.6.
DSN	Integer	0x00–0xff	The DSN of the received data frame.
Timestamp	Integer	0x000000–0xffffff	Optional. The time, in symbols, at which the data were received, as described in 5.1.4.1. The symbol boundary is described by *macSyncSymbolOffset*, as described in Table 52. The precision of this value shall be a minimum of 20 bits, with the lowest 4 bits being the least significant.
SecurityLevel	Integer	0x00–0x07	The security level purportedly used by the received data frame, as defined in Table 58.
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key purportedly used by the originator of the received frame, as defined in Table 58. This parameter is invalid if the SecurityLevel parameter is set to 0x00.
KeySource	Set of octets	As specified by the KeyIdMode parameter	The originator of the key purportedly used by the originator of the received frame, as described in 7.4.3.1. This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key purportedly used by the originator of the received frame, as described in 7.4.3.2. This parameter is invalid if the KeyIdMode parameter is invalid or set to 0x00.
UWBPRF	Integer	0, 4, 16, 64	The pulse repetition value of the received PPDU. This parameter shall be ignored by non-UWB PHYs.

Table 48—MCPS-DATA.indication parameters *(continued)*

Name	Type	Valid range	Description
UWBPreamble Symbol Repetitions	Enumer-ation	As defined in Table 46	As defined in Table 46.
DataRate	Integer	As defined in Table 46	As defined in Table 46.
Ranging Received	Enumer-ation	NO_RANGING_REQUESTED, RANGING_ACTIVE, RANGING_REQUESTED_BUT_NOT_SUPPORTED	A value of RANGING_REQUESTED_ BUT_NOT_SUPPORTED indicates that ranging is not supported but has been requested. A value of NO_RANGING_ REQUESTED indicates that no ranging is requested for the PSDU received. A value of RANGING_ACTIVE denotes ranging operations requested for this PSDU. A value of NO_RANGING_ REQUESTED is used for PHYs that do not support ranging.
RangingCounter Start	Unsigned Integer	As defined in Table 47	As defined in Table 47.
RangingCounter Stop	Unsigned Integer	As defined in Table 47	As defined in Table 47.
Ranging TrackingInterval	Integer	As defined in Table 47	As defined in Table 47.
RangingOffset	Signed Magni-tude Integer	As defined in Table 47	As defined in Table 47.
RangingFOM	Integer	As defined in Table 47	As defined in Table 47.

The MCPS-DATA.indication primitive is generated by the MAC sublayer and issued to the next higher layer on receipt of a data frame at the local MAC sublayer entity that passes the appropriate message filtering operations as described in 5.1.6.2. If the primitive is received while the device is in promiscuous mode, the parameters will be set as specified in 5.1.6.5.

6.3.4 MCPS-PURGE.request

The MCPS-PURGE.request primitive allows the next higher layer to purge an MSDU from the transaction queue.

The semantics of this primitive are:

```
MCPS-PURGE.request          (
                            msduHandle
                            )
```

The primitive parameters are defined in Table 49.

On receipt of the MCPS-PURGE.request primitive, the MAC sublayer attempts to find in its transaction queue the MSDU indicated by the msduHandle parameter. If an MSDU has left the transaction queue, the handle will not be found, and the MSDU can no longer be purged. If an MSDU matching the given handle is found, the MSDU is discarded from the transaction queue.

Table 49—MCPS-PURGE.request parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle of the MSDU to be purged from the transaction queue.

6.3.5 MCPS-PURGE.confirm

The MCPS-PURGE.confirm primitive allows the MAC sublayer to notify the next higher layer of the success of its request to purge an MSDU from the transaction queue.

The semantics of this primitive are:

```
MCPS-PURGE.confirm      (
                        msduHandle,
                        status
                        )
```

The primitive parameters are defined in Table 50.

Table 50—MCPS-PURGE.confirm parameters

Name	Type	Valid range	Description
msduHandle	Integer	0x00–0xff	The handle of the MSDU requested to be purged from the transaction queue.
status	Enumeration	SUCCESS, INVALID_HANDLE	The status of the request to purge an MSDU from the transaction queue.

The MCPS-PURGE.confirm primitive is generated by the MAC sublayer entity in response to an MCPS-PURGE.request primitive. If an MSDU matching the given handle is found, the status will be set to SUCCESS. If an MSDU matching the given handle is not found, the status will be set to INVALID_HANDLE.

6.4 MAC constants and PIB attributes

This subclause specifies the constants and attributes required by the MAC sublayer.

6.4.1 MAC constants

The constants that define the characteristics of the MAC sublayer are presented in Table 51.

6.4.2 MAC PIB attributes

The MAC PIB comprises the attributes required to manage the MAC sublayer of a device. The attributes contained in the MAC PIB are presented in Table 52. Attributes marked with a dagger (†) are read-only attributes (i.e., attribute can only be set by the MAC sublayer), which can be read by the next higher layer using the MLME-GET.request primitive. All other attributes can be read or written by the next higher layer using the MLME-GET.request or MLME-SET.request primitives, respectively. Attributes marked with a

Table 51—MAC sublayer constants

Constant	Description	Value
aBaseSlotDuration	The number of symbols forming a superframe slot when the superframe order is equal to zero, as described in 5.1.1.1.	60
aBaseSuperframeDuration	The number of symbols forming a superframe when the superframe order is equal to zero.	*aBaseSlotDuration* × *aNumSuperframeSlots*
aGTSDescPersistenceTime	The number of superframes in which a GTS descriptor exists in the beacon frame of the PAN coordinator.	4
aMaxBeaconOverhead	The maximum number of octets added by the MAC sublayer to the MAC payload of a beacon frame.	75
aMaxBeaconPayloadLength	The maximum size, in octets, of a beacon payload.	*aMaxPHYPacketSize* – *aMaxBeaconOverhead*
aMaxLostBeacons	The number of consecutive lost beacons that will cause the MAC sublayer of a receiving device to declare a loss of synchronization.	4
aMaxMACSafePayloadSize	The maximum number of octets that can be transmitted in the MAC Payload field of an unsecured MAC frame that will be guaranteed not to exceed *aMaxPHYPacketSize*.	*aMaxPHYPacketSize* – *aMaxMPDUUnsecuredOverhead*
aMaxMACPayloadSize	The maximum number of octets that can be transmitted in the MAC Payload field.	*aMaxPHYPacketSize* – *aMinMPDUOverhead*
aMaxMPDUUnsecuredOverhead	The maximum number of octets added by the MAC sublayer to the PSDU without security.	25
aMaxSIFSFrameSize	The maximum size of an MPDU, in octets, that can be followed by a SIFS period.	18
aMinCAPLength	The minimum number of symbols forming the CAP. This ensures that MAC commands can still be transferred to devices when GTSs are being used. An exception to this minimum shall be allowed for the accommodation of the temporary increase in the beacon frame length needed to perform GTS maintenance, as described in 5.2.2.1.3.	440
aMinMPDUOverhead	The minimum number of octets added by the MAC sublayer to the PSDU.	9
aNumSuperframeSlots	The number of slots contained in any superframe.	16
aUnitBackoffPeriod	The number of symbols forming the basic time period used by the CSMA-CA algorithm.	20

diamond (◆) are optional for an RFD; attributes marked with an asterisk (*) are optional for both device types (i.e., RFD and FFD).

Table 52—MAC PIB attributes

Attribute	Type	Range	Description	Default
macExtendedAddress†	IEEE address	Device specific	The extended address assigned to the device.	Implementation specific
macAckWaitDuration†	Integer	As defined in 6.4.3	The maximum number of symbols to wait for an acknowledgment frame to arrive following a transmitted data frame. This value is dependent on the supported PHY, which determines both the selected channel and channel page. The calculated value is the time to commence transmitting the ACK plus the length of the ACK frame. The commencement time is described in 5.1.6.4.2.	Dependent on currently selected PHY, indicated by *phyCurrentPage*
macAssociatedPANCoord	Boolean	TRUE, FALSE	Indication of whether the device is associated to the PAN through the PAN coordinator. A value of TRUE indicates the device has associated through the PAN coordinator. Otherwise, the value is set to FALSE.	FALSE
macAssociationPermit◆	Boolean	TRUE, FALSE	Indication of whether a coordinator is currently allowing association. A value of TRUE indicates that association is permitted.	FALSE
macAutoRequest	Boolean	TRUE, FALSE	Indication of whether a device automatically sends a data request command if its address is listed in the beacon frame. A value of TRUE indicates that the data request command is automatically sent. This attribute also affects the generation of the MLME-BEACON-NOTIFY.indication primitive, as described in 6.2.4.1.	TRUE
macBattLifeExt	Boolean	TRUE, FALSE	Indication of whether BLE, through the reduction of coordinator receiver operation time during the CAP, is enabled. A value of TRUE indicates that it is enabled. The effect of this attribute on the backoff exponent in the CSMA-CA algorithm is explained in 5.1.1.4.	FALSE
macBattLifeExtPeriods	Integer	6-41	In BLE mode, the number of backoff periods during which the receiver is enabled after the IFS following a beacon. This value is dependent on the supported PHY and is the sum of three terms: Term 1: The value $2^x - 1$, where x is the maximum value of *macMinBE* in BLE mode (equal to two). This term is thus equal to threebackoff periods. Term 2: The duration of the initial contention window length, as described in 5.1.1.4. Term 3: The Preamble field length and the SFD field length of the supported PHY summed together and rounded up (if necessary) to an integer number of backoff periods.	Dependent on currently selected PHY, indicated by *phyCurrentPage*

Table 52—MAC PIB attributes *(continued)*

Attribute	Type	Range	Description	Default
macBeaconPayload◆	Set of octets	—	The contents of the beacon payload.	NULL
macBeaconPayload Length◆	Integer	0 – *aMaxBeacon PayloadLength*	The length, in octets, of the beacon payload.	0
macBeaconOrder◆	Integer	0–15	Indicates the frequency with which the beacon is transmitted, as defined in 5.1.1.1.	15
macBeaconTxTime†◆	Integer	0x000000–0xffffff	The time that the device transmitted its last beacon frame, in symbol periods. The measurement shall be taken at the same symbol boundary within every transmitted beacon frame, the location of which is implementation specific. The precision of this value shall be a minimum of 20 bits, with the lowest four bits being the least significant.	0x000000
macBSN◆	Integer	0x00–0xff	The sequence number added to the transmitted beacon frame.	Random value from within the range
macCoordExtended Address	IEEE address	An extended IEEE address	The address of the coordinator through which the device is associated.	—
macCoordShort Address	Integer	0x0000–0xffff	The short address assigned to the coordinator through which the device is associated. A value of 0xfffe indicates that the coordinator is only using its extended address. A value of 0xffff indicates that this value is unknown.	0xffff
macDSN	Integer	0x00–0xff	The sequence number added to the transmitted data or MAC command frame.	Random value from within the range
*macGTSPermit**	Boolean	TRUE, FALSE	TRUE if the PAN coordinator is to accept GTS requests. FALSE otherwise.	TRUE
macMaxBE	Integer	3–8	The maximum value of the backoff exponent, BE, in the CSMA-CA algorithm, as defined in 5.1.1.4.	5
macMaxCSMABackoffs	Integer	0–5	The maximum number of backoffs the CSMA-CA algorithm will attempt before declaring a channel access failure.	4
macMaxFrameTotal WaitTime	Integer	As defined in 6.4.3	The maximum time to wait either for a frame intended as a response to a data request frame or for a broadcast frame following a beacon with the Frame Pending field set to one.	PHY dependent
macMaxFrameRetries	Integer	0–7	The maximum number of retries allowed after a transmission failure.	3

Table 52—MAC PIB attributes *(continued)*

Attribute	Type	Range	Description	Default
macMinBE	Integer	0–*macMaxBE*	The minimum value of the backoff exponent (BE) in the CSMA-CA algorithm, as described in 5.1.1.4.	3
macLIFSPeriod†	Integer	As defined in 8.1.3	The minimum time forming a LIFS period.	PHY dependent
macSIFSPeriod†	Integer	As defined in 8.1.3	The minimum time forming a SIFS period.	PHY dependent
macPANId	Integer	0x0000–0xffff	The identifier of the PAN on which the device is operating. If this value is 0xffff, the device is not associated.	0xffff
macPromiscuous Mode◆	Boolean	TRUE, FALSE	Indication of whether the MAC sublayer is in a promiscuous (receive all) mode. A value of TRUE indicates that the MAC sublayer accepts all frames received from the PHY.	FALSE
macRanging Supported†*	Boolean	TRUE, FALSE	This indicates whether the MAC sublayer supports the optional ranging features.	Implementation specific
macResponseWaitTime	Integer	2–64	The maximum time, in multiples of *aBaseSuperframeDuration*, a device shall wait for a response command frame to be available following a request command frame.	32
macRxOnWhenIdle	Boolean	TRUE, FALSE	Indication of whether the MAC sublayer is to enable its receiver during idle periods. For a beacon-enabled PAN, this attribute is relevant only during the CAP of the incoming superframe. For a non-beacon-enabled PAN, this attribute is relevant at all times.	FALSE
macSecurityEnabled	Boolean	TRUE, FALSE	Indication of whether the MAC sublayer has security enabled. A value of TRUE indicates that security is enabled, while a value of FALSE indicates that security is disabled.	FALSE
macShortAddress	Integer	0x0000–0xffff	The address that the device uses to communicate in the PAN. If the device is the PAN coordinator, this value shall be chosen before a PAN is started. Otherwise, the short address is allocated by a coordinator during association. A value of 0xfffe indicates that the device has associated but has not been allocated an address. A value of 0xffff indicates that the device does not have a short address.	0xffff
macSuperframe Order†◆	Integer	0–15	The length of the active portion of the outgoing superframe, including the beacon frame, as defined in 5.1.1.1	15

Table 52—MAC PIB attributes *(continued)*

Attribute	Type	Range	Description	Default
macSyncSymbolOffset†	Integer	0x000–0x100 for the 2.4 GHz band 0x000–0x400 for the 868 MHz and 915 MHz bands	The offset, measured in symbols, between the symbol boundary at which the MLME captures the timestamp of each transmitted or received frame, and the onset of the first symbol past the SFD, namely, the first symbol of the Length field.	Implementation specific
macTimestamp Supported†	Boolean	TRUE, FALSE	Indication of whether the MAC sublayer supports the optional timestamping feature for incoming and outgoing data frames.	Implementation specific
macTransaction PersistenceTime◆	Integer	0x0000–0xffff	The maximum time (in unit periods) that a transaction is stored by a coordinator and indicated in its beacon. The unit period is governed by *macBeaconOrder*, *BO*, as follows: For $0 \le BO \le 14$, the unit period will be *aBaseSuperframeDuration* $\times 2^{BO}$. For $BO = 15$, the unit period will be *aBaseSuperframeDuration*.	0x01f4
macTxControlActive Duration	Integer	0–100000	The duration for which transmit is permitted without pause specified in symbols.	2000 for BPSK PHY and 10000 for GFSK PHY
macTxControlPause Duration	Integer	2000 or 10000	The duration after transmission before another transmission is permitted specified in symbols.	2000 for BPSK PHY and 10000 for GFSK PHY
macTxTotalDuration	Integer	0x0–0xffffffff	The total transmit duration (including PHY header and FCS) specified in symbols. This can be read and cleared by NHL.	0

6.4.3 Calculating PHY dependent MAC PIB values

The read-only attribute *macAckWaitDuration* is dependent on a combination of constants and PHY PIB attributes. The PHY PIB attributes are listed in Table 71. The formula for relating the constants and attributes is:

$$macAckWaitDuration = aUnitBackoffPeriod + aTurnaroundTime + phySHRDuration + \text{ceiling}(6 \times phySymbolsPerOctet)$$

where the number 6 comes from the number of PHY header octets plus the number of PSDU octets in an acknowledgment frame and ceiling() is a function that returns the smallest integer value greater than or equal to its argument value.

The attribute *macMaxFrameTotalWaitTime* may be set by the next higher layer and is dependent upon a combination of PHY and MAC PIB attributes and constants. The formula relating the attributes and constants is:

$$macMaxFrameTotalWaitTime = \left[\left(\sum_{k=0}^{m-1} 2^{macMinBE+k}\right) + (2^{macMinBE} - 1) \times (macMaxCSMABackoffs - m)\right] \times aUnitBackoffPeriod + phyMaxFrameDuration$$

where

m = min(*macMaxBE-macMinBE, macMaxCSMABackoffs)*

For the UWB PHY, where the CCA mode is ALOHA and there are two octets in the PHR, the formula for *macAckWaitDuration* reduces to the following:

macAckWaitDuration = *macSIFSPeriod* + *phySHRDuration* + ceiling(7 × p*hySymbolsPerOctet*)

where the number 7 comes from the number of PHR octets plus the number of PSDU octets in an acknowledgment frame and *phySHRDuration* is defined in 9.4.

For UWB PHYs, *macMaxBE*, *macMaxCSMABackoffs*, and *m* become zero; therefore, the formula for *macMaxFrameTotalWaitTime* reduces to the following:

macMaxFrameTotalWaitTime = *phyMaxFrameDuration*/T_{psym}

where T_{psym} is defined in Table 100 (appropriate for the channel, PRF, and preamble code) and *phyMaxFrameDuration* is defined in 9.4.

Note that T_{psym} depends on the transmit parameters UWBPRF, UWBPreambleSymbolRepetitions, and DataRate provided by the next higher layer with the MCPS-DATA.request primitive. The formula and the T_{psym} values in Table 100 are intended to aid the next higher layer implementer in determining appropriate values for turnaround and timeouts.

For the CSS PHY, the values of the read-only attribute *macAckWaitDuration* depend on the selected data rate of the PPDU.

For the mandatory data rate (1 Mb/s), *macAckWaitDuration* is calculated as:

$$macAckWaitDuration_{1M} = aUnitBackoffPeriod + macSIFSPeriod + phySHRDuration_{1M} + [1.5 + 3/4 \times \text{ceiling}(4/3 \times 5)] \times phySymbolsPerOctet_{1M}$$

For the optional data rate (250 kb/s), *macAckWaitDuration* is calculated as:

$$macAckWaitDuration_{250k} = aUnitBackoffPeriod + macSIFSPeriod + phySHRDuration_{250k} + 3 \times \text{ceiling}(1/3 \times [1.5 + 5]) \times phySymbolsPerOctet_{250k}$$

7. Security

7.1 Overview

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. This standard supports the following security services:

— Data confidentiality
— Data authenticity
— Replay protection

7.2 Functional description

A device may optionally implement security. A device that does not implement security shall not provide a mechanism for the MAC sublayer to perform any cryptographic transformation on incoming and outgoing frames nor require any PIB attributes associated with security. A device that implements security shall provide a mechanism for the MAC sublayer to provide cryptographic transformations on incoming and outgoing frames using information in the PIB attributes associated with security only if the *macSecurityEnabled* attribute is set to TRUE.

If the MAC sublayer is required to transmit a frame or receives an incoming frame, the MAC sublayer shall process the frame as specified in 7.2.1 and 7.2.3, respectively.

7.2.1 Outgoing frame security procedure

The inputs to this procedure are the frame to be secured and the SecurityLevel, KeyIdMode, KeySource, and KeyIndex parameters from the originating primitive or automatic request PIB attributes. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the secured frame.

The outgoing frame security procedure involves the following steps:

a) If the *macSecurityEnabled* attribute is set to FALSE and the SecurityLevel parameter is not equal to zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
b) The procedure shall determine whether the frame to be secured satisfies the constraint on the maximum length of MAC frames, as follows:
    1) The procedure shall determine the length AuthLen, in octets, of the Authentication field, AuthLen, from the SecurityLevel parameter and Table 58.
    2) The procedure shall determine the length AuxLen, in octets, of the auxiliary security header, as described in 7.4, using KeyIdMode and the SecurityLevel parameter.
    3) The procedure shall determine the data expansion as AuxLen + AuthLen.
    4) The procedure shall check whether the length of the frame to be secured, including data expansion and FCS, is less than or equal to *aMaxPHYPacketSize*. If this check fails, the procedure shall return with a status of FRAME_TOO_LONG.
c) If the SecurityLevel parameter is zero, the procedure shall set the secured frame to be the frame to be secured and return with a status of SUCCESS.
d) The procedure shall set the frame counter to the *macFrameCounter* attribute. If the frame counter has the value 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.
e) The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 7.2.2 with the device addressing mode set to DstAddrMode, the device PAN ID set to

DstPANId, and the device address set to DstAddr. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

f) The procedure shall insert the auxiliary security header into the frame, with fields set as follows:

1) The Security Level field of the Security Control field shall be set to the SecurityLevel parameter.

2) The Key Identifier Mode field of the Security Control field shall be set to the KeyIdMode parameter.

3) The Frame Counter field shall be set to the frame counter.

4) If the KeyIdMode parameter is set to a value not equal to zero, the Key Source and Key Index fields of the Key Identifier field shall be set to the KeySource and KeyIndex parameters, respectively.

g) The Private Payload field and Open Payload field shall be set as indicated in Table 53. The procedure shall then use the Private Payload field, the Open Payload field, the *macExtendedAddress*, the frame counter, the SecurityLevel parameter, and the Key element of the KeyDescriptor to produce the secured frame according to the CCM* transformation process defined in 7.3.4.

Table 53—Private Payload field and Open Payload field definitions

Frame type	Private Payload field	Open Payload field
Beacon	Beacon Payload field	All other fields in the MAC Payload field
Data	Data Payload field	None
MAC command	Command Payload	All other fields in the MAC Payload field

h) The procedure shall increment the frame counter by one and set the *macFrameCounter* attribute to the resulting value.

i) The procedure shall return with a status of SUCCESS.

7.2.2 KeyDescriptor lookup procedure

The inputs to this procedure are the KeyIdMode, KeySource, KeyIndex, device addressing mode, device PAN ID, and device address. The outputs from this procedure are a status and, if passed, a KeyDescriptor.

The procedure involves the following steps:

a) If the KeyIdMode parameter is set to 0x00, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor:

1) If the device addressing mode is set to NO_ADDRESS, then the device PAN ID shall be set to *macPANId*. Otherwise, the device PAN ID shall be the value passed to the procedure.

2) If the device addressing mode is set to NO_ADDRESS and the frame type is beacon, then the device address shall be *macCoordExtendedAddress*.

3) If the device addressing mode is set to NO_ADDRESS and the frame type is not beacon, then:

i) If the *macCoordShortAddress* attribute is set to 0xfffe, then the device address shall be set to the *macCoordExtendedAddress*.

ii) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the device address shall be set to the *macCoordShortAddress*.

iii) If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with a failed status.

4) If the device addressing mode is set to SHORT_ADDRESS or EXTENDED_ADDRESS, then the device address shall be the value passed to the procedure.

5) If the device addressing mode, device PAN ID, and device address match the DeviceAddrMode, DevicePANId, and DeviceAddress of a KeyIdLookupDescriptor, then the procedure returns with the corresponding KeyDescriptor and passed status.

b) If the KeyID mode parameter is set to 0x01, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor, if the KeyIndex matches the KeyIndex of a KeyIdLookupDescriptor and the KeySource matches *macDefaultKeySource*, then the procedure returns with the corresponding KeyDescriptor and passed status.

c) If the KeyID mode parameter is set to 0x02 or 0x03, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor, if the KeySource and KeyIndex match the KeySource and KeyIndex of a KeyIdLookupDescriptor, then the procedure returns with the KeyDescriptor and passed status.

d) The procedure shall return with a failed status.

NOTE—For broadcast frames, the KeyDescriptor lookup procedure will result in a failed status if implicit key identification is used. Hence, explicit key identification should be used for broadcast frames.[6]

7.2.3 Incoming frame security procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the unsecured frame, the security level, the key identifier mode, the key source, and the key index.

All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure.

The incoming frame security procedure involves the following steps:

a) If the Security Enabled field of the frame to be unsecured is set to zero, the procedure shall set the security level to zero.

b) If the Security Enabled field of the frame to be unsecured is set to one and the Frame Version field of the frame to be unsecured is set to zero, the procedure shall return with a status of UNSUPPORTED_LEGACY.

c) If the Security Enabled field of the frame to be unsecured is set to one, the procedure shall set the security level and the key identifier mode to the corresponding fields of the Security Control field of the auxiliary security header of the frame to be unsecured, and the key source and key index to the corresponding fields of the Key Identifier field of the auxiliary security header of the frame to be unsecured, if present. If the resulting security level is zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.

d) If the *macSecurityEnabled* attribute is set to FALSE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS if the security level is equal to zero and with a status of UNSUPPORTED_SECURITY otherwise.

e) The device PAN ID shall be set to the Source PAN Identifier field, if it is present. If the PAN ID compression field is set to one, then the device PAN ID shall be set to the Destination PAN Identifier field. The device addressing mode shall be set according to the Source Addressing Mode field, as defined in Table 54. The device address shall be set to the Source Address, if present. The KeyIdMode shall be set to the Key Identifier Mode field, the KeyIndex shall be set to the Key Index field, if present, and the KeySource shall be set to the Key Source field, if present.

[6]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

Table 54—Mapping of Source Addressing Mode field to device addressing mode

Source Addressing Mode field	Device addressing mode
0x00	NO_ADDRESS
0x02	SHORT_ADDRESS
0x03	EXTENDED_ADDRESS

f) The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 7.2.2 with using the KeyIdMode, KeyIndex, KeySource, device addressing mode, device PANID, and device address. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

g) The procedure shall obtain the DeviceDescriptor using the DeviceDescriptor lookup procedure described in 7.2.4. If that procedure fails, then the procedure shall return with a status of UNAVAILABLE_DEVICE.

h) The procedure shall obtain the SecurityLevelDescriptor by passing the frame type and, if the frame is a MAC command frame, the command frame identifier, to the SecurityLevelDescriptor lookup procedure described in 7.2.5. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_SECURITY_LEVEL.

i) The procedure shall determine whether the frame to be unsecured conforms to the security level policy by passing the SecurityLevelDescriptor and the security level to the incoming security level checking procedure, as described in 7.2.6. If that procedure returns with a failed status, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL; otherwise, if that procedure returns with a passed status and the security level is equal to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS.

j) If the incoming security level checking procedure of step i) had as output the ‘conditionally passed’ status and the Exempt element of the DeviceDescriptor is set to TRUE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS.

k) If the incoming security level checking procedure of step i) had as output the ‘conditionally passed’ status and the Exempt element of the DeviceDescriptor is set to FALSE, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL.

l) The procedure shall set the frame counter to the Frame Counter field of the frame to be unsecured. If the frame counter has the value 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.

m) If the frame counter is less than the FrameCounter element of the DeviceDescriptor, the procedure shall return with a status of COUNTER_ERROR.

n) The procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the KeyDescriptor, the frame type, and, if the frame is a MAC command frame, the command frame identifier, to the incoming key usage policy checking procedure, as described in 7.2.7. If that procedure fails, the procedure shall return with a status of IMPROPER_KEY_TYPE.

o) The Private Payload field and Open Payload field shall be set as indicated in Table 53. The procedure shall then use the Private Payload field, the Open Payload field, the ExtAddress element of the DeviceDescriptor, the frame counter, the security level, and the Key element of the KeyDescriptor to produce the unsecured frame, according to the CCM* inverse transformation process described in the security operations, as described in 7.3.5.

p) If the CCM* inverse transformation process fails, the procedure shall return with a status of SECURITY_ERROR.

q) The procedure shall increment the frame counter by one and set the FrameCounter element of the DeviceDescriptor to the resulting value.

r) The procedure shall return with a status of SUCCESS.

7.2.4 DeviceDescriptor lookup procedure

The inputs to this procedure are the device addressing mode, the device PAN ID, and the device address. The output from this procedure is a passed or failed status and, if passed, a DeviceDescriptor.

The DeviceDescriptor lookup procedure involves the following steps:

a) If the device addressing mode is set to NO_ADDRESS, then the device PAN ID shall be set to *macPANId*. Otherwise, the device PAN ID shall be the value passed to the procedure.

b) If the device addressing mode is set to NO_ADDRESS, then:

   1) If the *macCoordShortAddress* attribute is set to 0xfffe, then the device address shall be set to the *macCoordExtendedAddress*.

   2) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the device address shall be set to the *macCoordShortAddress*.If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with a failed status.

c) If the device addressing mode is set to SHORT_ADDRESS or EXTENDED_ADDRESS, then the device address shall be the value passed to the procedure.

d) For each DeviceDescriptor in DeviceDescriptorHandleList, if the device PAN ID matches the PANId and the and device address matches the ShortAddress, if the device addressing mode is set to SHORT_ADDRESS, or the ExtAddress, if the device addressing mode is set to EXTENDED_ADDRESS, then the procedure shall return with the corresponding DevicesDescriptor and a passed status.

e) The procedure shall return with a failed status.

7.2.5 SecurityLevelDescriptor lookup procedure

The inputs to this procedure are the frame type and, if the frame is a MAC command, the command frame identifier. The output from this procedure are a passed or failed status and, if passed, a SecurityLevelDescriptor.

The SecurityLevelDescriptor lookup procedure involves the following steps:

a) For each SecurityLevelDescriptor in the macSecurityLevelTable attribute:

   1) If the frame type indicates that the frame is not a MAC command and the frame type is equal to the FrameType element of the SecurityLevelDescriptor (i.e., there is a match), the procedure shall return with the SecurityLevelDescriptor and a passed status.

   2) If the frame type indicates that the frame is a MAC command and the frame type is equal to the FrameType element of the SecurityLevelDescriptor and the command frame identifier is equal to the CommandFrameIdentifier element of the SecurityLevelDescriptor, the procedure shall return with the SecurityLevelDescriptor and a passed status.

b) The procedure shall return with a failed status.

7.2.6 Incoming security level checking procedure

The inputs to this procedure are the SecurityLevelDescriptor and the incoming security level. The output from this procedure is a passed, failed, or ‘conditionally passed’ status.

The incoming security level checking procedure involves the following steps:

a) If the AllowedSecurityLevels is empty, then the procedure shall compare the incoming security level (as SEC1) with the SecurityMinimum element of the SecurityLevelDescriptor (as SEC2) according to the algorithm described in 7.4.1.1. If this comparison evaluates to TRUE, the procedure shall return with a passed status.

b) If the AllowedSecurityLevels is not empty, the procedure shall check whether the incoming security level is equal to any of the elements of the AllowedSecurityLevels. If this check is successful (i.e., there is a match), the procedure shall return with a passed status.

c) If the incoming security level is equal to 0x00 and the DeviceOverrideSecurityMinimum element of the SecurityLevelDescriptor is set to TRUE, the procedure shall return with a 'conditionally passed' status.

d) The procedure shall return with a failed status.

7.2.7 Incoming key usage policy checking procedure

The inputs to this procedure are the KeyDescriptor, the frame type, and the command frame identifier. The output from this procedure is a passed or failed status.

The incoming key usage policy checking procedure involves the following steps:

a) For each KeyUsageDescriptor in the KeyUsageList of the KeyDescriptor:
   1) If the frame type indicates that the frame is not a MAC command and the frame type is equal to the FrameType element of the KeyUsageDescriptor, the procedure shall return with a passed status.
   2) If the frame type indicates that the frame is a MAC command, the frame type is equal to the FrameType element of the KeyUsageDescriptor, and the command frame identifier is equal to the CommandFrameIdentifier element of the KeyUsageDescriptor, the procedure shall return with a passed status.

b) The procedure shall return with a failed status.

7.3 Security operations

This subclause describes the parameters for the CCM* security operations, as specified in B.3.2.

7.3.1 Integer and octet representation

The integer and octet representation conventions specified in B.2 are used throughout 7.3.

7.3.2 CCM* Nonce

The CCM* nonce is a 13-octet string and is used for the advanced encryption standard (AES)-CCM* mode of operation, as described in B.3.2. The nonce shall be formatted as shown in Figure 61, with the leftmost field in the figure defining the first (and leftmost) octets and the rightmost field defining the last (and rightmost) octet of the nonce.

Octets: 8	**4**	**1**
Source address	Frame counter	Security level

Figure 61—CCM* nonce

The source address shall be set to the extended address *macExtendedAddress* of the device originating the frame, the frame counter to the value of the respective field in the auxiliary security header, as defined in 7.4, and the security level to the value corresponding to the Security Level field, as defined in Table 58.

The source address, frame counter, and security level shall be represented as specified in 7.3.1.

7.3.3 CCM* prerequisites

Securing a frame involves the use of the CCM* mode encryption and authentication transformation, as described in B.4.1. Unsecuring a frame involves the use of the CCM* decryption and authentication checking transformation, as described in B.4.2.

The length *M* of the Authentication field for the CCM* forward transformation and the CCM* inverse transformation is determined from Table 58, using the Security Level field of the Security Control field of the auxiliary security header of the frame.

7.3.4 CCM* transformation data representation

This subclause describes how the inputs and outputs of the CCM* forward transformation, as described in B.4.1, are formed.

The inputs are:

— Key
— Nonce
— *a* data
— *m* data

The output is *c* data.

7.3.4.1 Key and nonce data inputs

The Key data for the CCM* forward transformation is passed by the outgoing frame security procedure described in 7.2.1. The Nonce data for the CCM* transformation is constructed as described in 7.3.2.

7.3.4.2 *a* data and *m* data

In the CCM* transformation process, the data fields shall be applied as in Table 55.

Table 55—*a* data and *m* data for all security levels

Security level	*a* data	*m* data
0	None	None
1	MHR || Open Payload field || Unsecured Private Payload field	None
2	MHR || Open Payload field || Unsecured Private Payload field	None
3	MHR || Open Payload field || Unsecured Private Payload field	None
4	None	Unsecured Private Payload field
5	MHR || Open Payload field	Unsecured Private Payload field

Table 55—*a* data and *m* data for all security levels *(continued)*

Security level	*a* data	*m* data
6	MHR || Open Payload field	Unsecured Private Payload field
7	MHR || Open Payload field	Unsecured Private Payload field

NOTE—The MHR contains the Auxiliary Security Header field, as defined in 5.2.1.

7.3.4.3 *c* data output

In the CCM* transformation process, the data fields that are applied, or right-concatenated and applied, represent octet strings.

The Private Payload field of the original unsecured frame shall be replaced by the right-concatenation of that field and the *c* field if data confidentiality is not provided and shall be replaced by the *c* field otherwise. The contents of the *c* data for each of the security levels is defined in Table 56.

Table 56—*c* data for all security levels

Security level	*c* data
0	None
1	MIC-32
2	MIC-64
3	MIC-128
4	Encrypted Private Payload field
5	Encrypted Private Payload field || MIC-32
6	Encrypted Private Payload field || MIC-64
7	Encrypted Private Payload field || MIC-128

7.3.5 CCM* inverse transformation data representation

This subclause describes how the inputs and outputs of the CCM* inverse transformation, as described in B.4.2, are formed.

The inputs are:

— Key
— Nonce
— *c* data
— *a* data

The output is *m* data.

7.3.5.1 Key and nonce data inputs

The Key data for the CCM* inverse transformation is passed by the incoming frame security procedure described in 7.2.3. The Nonce data for the CCM* transformation is constructed as described in 7.3.2.

7.3.5.2 *c* data and *a* data

In the CCM* inverse transformation process, the data fields shall be applied as in Table 57.

Table 57—*c* data and *a* data for all security levels

Security level	*c* data	*a* data
0	None	None
1	MIC-32	MHR || Open Payload field || Private Payload field
2	MIC-64	MHR || Open Payload field || Private Payload field
3	MIC-128	MHR || Open Payload field || Private Payload field
4	Encrypted Private Payload field	MHR || Open Payload field
5	Encrypted Private Payload field || MIC-32	MHR || Open Payload field
6	Encrypted Private Payload field || MIC-64	MHR || Open Payload field
7	Encrypted Private Payload field || MIC-128	MHR || Open Payload field

NOTE—The MHR contains the Auxiliary Security Header field, as defined in 5.2.1.

7.3.5.3 *m* data output

The Private Payload field of the MAC Payload shall be set to the *m* data if frame security includes providing confidentiality and shall be set to the Private Payload field of the MAC Payload, with the rightmost substring, *c*, deleted, otherwise.

7.4 Auxiliary security header

The Auxiliary Security Header field has a variable length and contains information required for security processing, including a Security Control field, a Frame Counter field, and a Key Identifier field. The Auxiliary Security Header field shall be present only if the Security Enabled field is set to one. The Auxiliary Security Header field shall be formatted as illustrated in Figure 62.

Octets: 1	4	0/1/5/9
Security Control	Frame Counter	Key Identifier

Figure 62—Format of the auxiliary security header

The auxiliary security header uses the representation conventions specified in 5.2.

7.4.1 Security Control field

The Security Control field is used to provide information about what protection is applied to the frame. The Security Control field shall be formatted as shown in Figure 63.

Bit: 0–2	3–4	5–7
Security Level	Key Identifier Mode	Reserved

Figure 63—Security Control field format

7.4.1.1 Security Level field

The Security Level field indicates the actual frame protection that is provided. This value can be adapted on a frame-by-frame basis and allows for varying levels of data authenticity (to allow minimization of security overhead in transmitted frames where required) and for optional data confidentiality. The cryptographic protection offered by the various security levels is shown in Table 58. When nontrivial protection is required, replay protection is always provided.

Table 58—Security levels available to the MAC sublayer

Security level	Security level field b_2 b_1 b_0	Security attributes	Data confidentiality	Data authenticity	Encrypted authentication tag length, *M*, (octets)
0	000	None	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

Security levels can be ordered according to the corresponding cryptographic protection offered. Here, a first security level SEC1 is greater than or equal to a second security level SEC2 if and only if SEC1 offers at least the protection offered by SEC2, both with respect to data confidentiality and with respect to data authenticity. The statement “SEC1 is greater than or equal to SEC2” shall be evaluated as TRUE if both of the following conditions apply:

a) Bit position b_2 in SEC1 is greater than or equal to bit position b_2 in SEC2 (where Encryption OFF < Encryption ON).

b) The integer value of bit positions b_1 b_0 in SEC1 is greater than or equal to the integer value of bit positions b_1 b_0 in SEC2 (where increasing integer values indicate increasing levels of data authenticity provided, i.e., message integrity code MIC-0 < MIC-32 < MIC-64 < MIC-128).

Otherwise, the statement shall be evaluated as FALSE.

For example, ENC-MIC-64 ≥ MIC-64 is TRUE because ENC-MIC-64 offers the same data authenticity protection as MIC-64, plus confidentiality. On the other hand, MIC-128 ≥ ENC-MIC-64 is FALSE because even though MIC-128 offers stronger data authenticity than ENC-MIC-64, it offers no confidentiality.

7.4.1.2 Key Identifier Mode field

The Key Identifier Mode field indicates whether the key that is used to protect the frame can be derived implicitly or explicitly; furthermore, it is used to indicate the particular representations of the Key Identifier field, as defined in 7.4.3, if derived explicitly. The Key Identifier Mode field shall be set to one of the values listed in Table 59. The Key Identifier field of the auxiliary security header, as defined in 7.4.3, shall be present only if this field has a value that is not equal to 0x00.

Table 59—Values of the key identifier mode

Key identifier mode	Key Identifier Mode field b_1 b_0	Description	Key Identifier field length (octets)
0x00	'00'	Key is determined implicitly from the originator and recipient(s) of the frame, as indicated in the frame header.	0
0x01	'01'	Key is determined from the Key Index field in conjunction with *macDefault-KeySource*.	1
0x02	'10'	Key is determined explicitly from the 4-octet Key Source field and the Key Index field.	5
0x03	'11'	Key is determined explicitly from the 8-octet Key Source field and the Key Index field.	9

7.4.2 Frame Counter field

The Frame Counter field represents the *macFrameCounter* attribute of the originator of a protected frame. It is used to provide semantic security of the cryptographic mechanism used to protect a frame and to offer replay protection.

7.4.3 Key Identifier field

The Key Identifier field has a variable length and identifies the key that is used for cryptographic protection of outgoing frames, either explicitly or in conjunction with implicitly defined side information. The Key Identifier field shall be present only if the Key Identifier Mode field, as defined in 7.4.1.2, is set to a value different from 0x00. The Key Identifier field shall be formatted as illustrated in Figure 64.

Octets: 0/4/8	1
Key Source	Key Index

Figure 64—Format for the Key Identifier field, if present

7.4.3.1 Key Source field

The KeySource field, when present, indicates the originator of a group key. If the Key Identifier Mode field indicates a 4 octet Key Source field, then the Key Source field shall be the *macPANId* of the originator of the group key right concatenated with the *macShortAddress* of the originator of the group key. If the Key Identifier Mode field indicates an 8 octet Key Source field, then the Key Source field shall be set to the *macExtendedAddress* of the originator of the group key.

7.4.3.2 Key Index field

The Key Index field allows unique identification of different keys with the same originator.

It is the responsibility of each key originator to make sure that the actively used keys that it issues have distinct key indices and that the key indices are all different from 0x00.

7.5 Security-related MAC PIB attributes

The PIB security-related attributes are defined in Table 60. A MAC implementation may impose additional constraints on read/write operations for the security-related PIB attributes.

Table 60— Security-related MAC PIB attributes

Attribute	Type	Range	Description	Default
macKeyTable	Set of *Key Descriptors*, as defined Table 61	—	KeyDescriptor entries, each containing keys and security related information.	(empty)
macDeviceTable	Set of *Device Descriptors*, as defined in Table 64	—	DeviceDescriptors for each remote device with which this device securely communicates.	(empty)
macSecurity LevelTable	Set of *SecurityLevel-Descriptors*, as defined in Table 63	—	Provides information about the security level required for each MAC frame type and subtype.	(empty)
macFrameCounter	Integer	0x00000000–0xffffffff	The outgoing frame counter for this device.	0x00000000
macAutoRequest SecurityLevel	Integer	As defined in Table 58	The security level used for automatic data requests.	0x06
macAutoRequest KeyIdMode	Integer	0x00–0x03	The key identifier mode used for automatic data requests. This attribute is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.	0x00
macAutoRequest KeySource	As specified by the *macAutoRequesKeyIdMode* parameter	—	The originator of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* element is invalid or set to 0x00.	All octets 0xff
macAutoRequest KeyIndex	Integer	0x01–0xff	The index of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* attribute is invalid or set to 0x00.	0xff

Table 60— Security-related MAC PIB attributes *(continued)*

Attribute	Type	Range	Description	Default
macDefaultKey Source	Set of 8 octets	—	The originator of the default key used for key identifier mode 0x01.	All octets 0xff
macPANCoord ExtendedAddress	IEEE address	An extended IEEE address	The extended address of the PAN coordinator.	—
macPANCoord ShortAddress	Integer	0x0000–0xffff	The short address assigned to the PAN coordinator. A value of 0xfffe indicates that the PAN coordinator is only using its extended address. A value of 0xffff indicates that this value is unknown.	0x0000

Table 61 defines the elements in a KeyDescriptor.

Table 61—Elements of KeyDescriptor

Name	Type	Range	Description
KeyIdLookup List	List of KeyIdLookupDescriptor entries, as defined in Table 65	—	A list of KeyIdLookupDescriptor entries used to identify this KeyDescriptor.
DeviceDescriptor HandleList	—	—	A list of implementation specific handles to DeviceDescriptor entries in *macDeviceTable* for each of the devices that are currently using this key.
KeyUsageList	List of KeyUsageDescriptor entries, as defined in Table 62	—	A list of KeyUsageDescriptor entries indicating the frame types with which this key may be used.
Key	Set of 16 octets	—	The value of the key.

Table 62 defines the elements of a KeyUsageDescriptor.

Table 62—Elements of KeyUsageDescriptor

Name	Type	Range	Description
FrameType	Integer	As defined in 5.2.1.1.1	As defined in 5.2.1.1.1.
CommandFrameIdentifier	Integer	As defined in Table 5	As defined in Table 5.

Table 63 defines the elements of a SecurityLevelDescriptor.

Table 64 defines the elements of a DeviceDescriptor.

Table 65 defines the elements of a KeyIdLookupDescriptor.

Table 63—Elements of SecurityLevelDescriptor

Name	Type	Range	Description
FrameType	Integer	As defined in 5.2.1.1.1	As defined in 5.2.1.1.1.
CommandFrameIdentifier	Integer	As defined in Table 5	As defined in Table 5.
SecurityMinimum	Integer	As defined in Table 58	The minimal required/expected security level, as defined in Table 58, for incoming MAC frames with the indicated frame type and, if present, command frame type.
DeviceOverrideSecurity Minimum	Boolean	TRUE, FALSE	Indication of whether originating devices for which the Exempt flag is set may override the security level indicated by the AllowedSecurityLevels, or if the or SecurityMinimum. If TRUE, this indicates that for originating devices with Exempt status, the incoming security level zero is acceptable, in addition to the incoming security levels meeting the minimum expected security level indicated by the SecurityMinimum element.
AllowedSecurityLevels	Set of integers	—	A set of allowed security levels, as defined in Table 58, for incoming MAC frames with the indicated frame type, and, if present, command frame identifier. If the set is empty, then the SecurityMinimum parameter applies instead.

Table 64—Elements of DeviceDescriptor

Name	Type	Range	Description
PANId	Device PAN ID	0x0000–0xffff	The PAN identifier of the device in this DeviceDescriptor.
ShortAddress	Device short address	0x0000–0xffff	The short address of the device in this DeviceDescriptor. A value of 0xfffe indicates that this device is using only its extended address. A value of 0xffff indicates that this value is unknown.
ExtAddress	IEEE address	Any valid extended IEEE address	The extended IEEE address of the device.
FrameCounter	Integer	0x00000000–0xffffffff	The incoming frame counter of the device.
Exempt	Boolean	TRUE, FALSE	Indication of whether the device may override the minimum security level settings defined in Table 63.

Table 65—Elements of KeyIdLookupDescriptor

Name	Type	Range	Description
KeyIdMode	Integer	As defined in Table 59	The mode used to for this descriptor.
KeySource	Set of octets	As defined in 7.4.3.1	Information to identify the key. Present only if KeyIdMode is equal to 0x02 or 0x03
KeyIndex	Integer	As defined in 7.4.3.1	Information used to identify the key. Present only if KeyIdMode is not equal to 0x00.
DeviceAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRES	The addressing mode for this descriptor. Present only if KeyIdMode is equal to 0x00.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier for this descriptor. Present only if KeyIdMode is equal to 0x00.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter	The device address for this descriptor. Present only if KeyIdMode is equal to 0x00.

8. General PHY requirements

8.1 General requirements and definitions

Unless otherwise specified in a PHY clause, all reserved fields shall be set to zero on transmission and may be ignored upon reception.

The PHY is responsible for the following tasks:

— Activation and deactivation of the radio transceiver
— Energy detection (ED) within the current channel
— Link quality indicator (LQI) for received packets
— Clear channel assessment (CCA) for carrier sense multiple access with collision avoidance (CSMA-CA)
— Channel frequency selection
— Data transmission and reception
— Precision ranging for ultra-wide band (UWB) PHYs

The PHYs defined in this standard are:

— **O-QPSK PHY**: direct sequence spread spectrum (DSSS) PHY employing offset quadrature phase-shift keying (O-QPSK) modulation, operating in the 780 MHz bands, 868 MHz, 915 MHz, and 2450 MHz, as defined in Clause 10.
— **BPSK PHY**: DSSS PHY employing binary phase-shift keying (BPSK) modulation, operating in the 868 MHz, 915 MHz, and 950 MHz bands, as defined in Clause 11.
— **ASK PHY**: parallel sequence spread spectrum (PSSS) PHY employing amplitude shift keying (ASK) and BPSK modulation, operating in the 868 MHz and 915 MHz bands, as defined in Clause 12.
— **CSS PHY**: chirp spread spectrum (CSS) employing differential quadrature phase-shift keying (DQPSK) modulation, operating in the 2450 MHz band, as defined in Clause 13.
— **UWB PHY**: combined burst position modulation (BPM) and BPSK modulation, operating in the sub-gigahertz and 3–10 GHz bands, as defined in Clause 14.
— **MPSK PHY**: M-ary phase-shift keying (MPSK) modulation, operating in the 780 MHz band, as defined in Annex G.
— **GFSK PHY**: Gaussian frequency-shift keying (GFSK), operating in the 950 MHz band, as defined in Clause 15.

8.1.1 Operating frequency range

A compliant device shall operate in one or several frequency bands using the modulation and spreading formats summarized in Table 66.

Devices shall start in the PHY mode in which they are instructed to start. If the device is capable of operating in the 868 MHz or 915 MHz bands using one of the optional PHYs described in Clause 10 and Clause 12, it shall be able to switch dynamically between the optional PHY and the mandatory BPSK PHY in that band when instructed to do so.

If the 950 MHz band is supported, then at least one of the 950 MHz band PHYs shall be implemented.

Table 66—Frequency bands and data rates

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
780	779–787	1000	O-QPSK	250	62.5	16-ary orthogonal
780	779–787	1000	MPSK	250	62.5	16-ary orthogonal
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary orthogonal
950	950–956	—	GFSK	100	100	Binary
950	950–956	300	BPSK	20	20	Binary
2450 DSSS	2400–2483.5	2000	O-QPSK	250	62.5	16-ary orthogonal
UWB sub-gigahertz (optional)	250–750	As defined in 14.4.1				
2450 CSS (optional)	2400–2483.5	As defined in 13.2		250	167 (as defined in 13.4.2)	
		As defined in 13.2		1000	167 (as defined in 13.4.2)	
UWB low band (optional)	3244–4742	As defined in 14.4.1				
UWB high band (optional)	5944–10 234	As defined in 14.4.1				

Operation of the MPSK PHY in the 780 MHz band is described in Annex G. The CWPAN specification also defines operation in the 314–316 MHz and 430–434 MHz bands, which is not described in this standard.

8.1.2 Channel assignments

Channel assignments are defined through a combination of channel numbers and channel pages.

The *phyChannelsSupported* PHY PAN information base (PIB) attribute indicates which channel pages are supported by the current PHY, while the *phyCurrentPage* PHY PIB attribute identifies the channel page that is currently used. The PHY PIB attributes are described in 9.3.

If the requested PHY PIB attribute is the *phyCurrentPage*, the attribute was successfully set to a different value from the current value, and the channel is no longer valid, then the PHY shall also set the *phyCurrentChannel* to the lowest valid channel for the requested page.

For each PHY supported, a compliant device shall support all channels allowed by regulations for the region in which the device operates. An exception to this is the UWB PHY where specific mandatory and optional behaviors are as defined in 14.4.1.

8.1.2.1 Channel numbering for 780 MHz band

For channel page five, channels numbered zero to seven are available across the 780 MHz band. The center frequency of these channels is defined as follows:

$F_c = 780 + 2\,k$ in megahertz, for $k = 0, \ldots, 3$

$F_c = 780 + 2\,(k - 4)$ in megahertz, for $k = 4, \ldots, 7$

where k is the channel number.

8.1.2.2 Channel numbering for 868 MHz, 915 MHz, and 2450 MHz bands

For channel page zero, 16 channels are available in the 2450 MHz band, 10 in the 915 MHz band, and 1 in the 868 MHz band. This channel page supports the channels defined in the 2003 edition of this standard. The center frequency of these channels is defined as follows:

$F_c = 868.3$ in megahertz, for $k = 0$

$F_c = 906 + 2\,(k - 1)$ in megahertz, for $k = 1, 2, \ldots, 10$

and $F_c = 2405 + 5\,(k - 11)$ in megahertz, for $k = 11, 12, \ldots, 26$

where

k is the channel number.

For channel pages one and two, 11 channels numbered zero to ten are available across the two frequency bands to support the ASK and O-QPSK PHYs, respectively. Ten channels are available in the 915 MHz band and one in the 868 MHz band. The center frequency of these channels is defined as follows:

$F_c = 868.3$ in megahertz, for $k = 0$

and $F_c = 906 + 2\,(k - 1)$ in megahertz, for $k = 1, 2, \ldots, 10$

where

k is the channel number.

8.1.2.3 Channel numbering for 950 MHz band

For channel page six, channels numbered 0 to 21 are available across the 950 MHz band. Channels 0–7 are for 1 mW BPSK, 8–9 are for 10 mW BPSK, and 10–21 are for GFSK. The center frequency of these channels is defined as follows:

$F_c = 951.2 + 0.6\ k$ in megahertz, for $k = 0, \ldots, 7$

$F_c = 954.4 + 0.2\ (k - 8)$ in megahertz, for $k = 8, 9$

and $F_c = 951.1 + 0.4\ (k - 10)$ in megahertz, for $k = 10, \ldots, 21$

where

k is the channel number.

For each PHY supported, a compliant device shall support all channels allowed by regulations for the region in which the device operates, except for channels 14 and 17, which are optional.

8.1.2.4 Channel numbering for CSS PHY

The CSS PHY uses channel page three with the channel numbers defined in Table 67. Different subsets of these frequency channels are available in different regions of the world. In North America and Europe, three frequency channels can be selected so that the nonoverlapping frequency channels are used.

Table 67—Center frequencies of CSS

Channel number	Frequency (MHz)
0	2412
1	2417
2	2422
3	2427
4	2432
5	2437
6	2442
7	2447
8	2452
9	2457
10	2462
11	2467
12	2472
13	2484

8.1.2.5 Channel numbering for UWB PHY

The UWB PHY uses channel page four with the channel numbers defined in Table 68. A compliant UWB device shall be capable of transmitting in at least one of three specified bands, sub-gigahertz, low, or high. A UWB device that implements the sub-gigahertz band shall implement channel 0. A UWB device that implements the low band shall support channel 3. The remaining low-band channels are optional. A UWB

device that implements the high band shall support channel 9. The remaining high-band channels are optional.

Table 68—UWB PHY channel frequencies

Channel number	Center frequency (MHz)	UWB band/mandatory
0	499.2	Sub-gigahertz
1	3494.4	Low band
2	3993.6	
3	4492.8	
4	3993.6	
5	6489.6	High band
6	6988.8	
7	6489.6	
8	7488.0	
9	7987.2	
10	8486.4	
11	7987.2	
12	8985.6	
13	9484.8	
14	9984.0	
15	9484.8	

8.1.3 Minimum LIFS and SIFS periods

For all PHYs other than the UWB PHY, the minimum LIFS period and SIFS period are:

— *macLIFSPeriod* – 40 symbols
— *macSIFSPeriod* – 12 symbols

For the UWB PHY, the minimum LIFS period and SIFS period are:

— *macLIFSPeriod* – 40 preamble symbols
— *macSIFSPeriod* – 12 preamble symbols

For the UWB PHY, the actual time for *macLIFSPeriod* and *macSIFSPeriod* depends on the PRF in use and on the channel, as described in Table 100.

8.1.4 RF power measurement

Unless otherwise stated, all RF power measurements, either transmit or receive, shall be made at the appropriate transceiver to antenna connector. The measurements shall be made with equipment that is either matched to the impedance of the antenna connector or corrected for any mismatch. For devices without an antenna connector, the measurements shall be interpreted as effective isotropic radiated power (EIRP) (i.e., a 0 dBi gain antenna), and any radiated measurements shall be corrected to compensate for the antenna gain in the implementation.

8.1.5 Transmit power

The maximum transmit power shall conform with local regulations. A compliant device shall have its nominal transmit power level indicated by its PHY parameter, *phyTXPower*, as defined in 9.3.

For UWB PHYs, the parameter *phyTXPower* refers to the total power transmitted across the entire occupied bandwidth (not the dBm/MHz usually found in regulations).

8.1.6 Out-of-band spurious emission

The out-of-band spurious emissions shall conform with local regulations.

8.1.7 Receiver sensitivity definitions

The conditions for measuring receiver sensitivity are defined in Table 69.

Table 69—Receiver sensitivity conditions

Term	Definition of term	Conditions
Packet error rate (PER)	Average fraction of transmitted packets that are not correctly received.	– Average measured over random PSDU data.
Receiver sensitivity	Lowest input power for which the PER conditions are met.	– PSDU length = 20 octets. – PER < 1%. – Power measured at antenna terminals. – Interference not present.

8.2 General radio specifications

8.2.1 TX-to-RX turnaround time

The TX-to-RX turnaround time shall be less than or equal to *aTurnaroundTime*, as defined in 9.2.

The TX-to-RX turnaround time is defined as the time at the air interface from the trailing edge of the last part/chip (of the last symbol) of a transmitted PPDU to the leading edge of the first part/chip (of the first symbol) of the next received PPDU.

8.2.2 RX-to-TX turnaround time

The RX-to-TX turnaround time shall be less than or equal to *aTurnaroundTime*, as defined in 9.2.

The RX-to-TX turnaround time is defined as the time at the air interface from the trailing edge of the last chip (of the last symbol) of a received PPDU to the leading edge of the first chip (of the first symbol) of the next transmitted PPDU.

8.2.3 Error-vector magnitude (EVM) definition

The modulation accuracy of the transmitter is determined with an EVM measurement. In order to calculate the EVM, a time record of N received complex chip values $(\tilde{I}_j, \tilde{Q}_j)$ is captured. For each received complex chip, a decision is made about which complex chip value was transmitted. The ideal position of the chosen complex chip (the center of the decision box) is represented by the vector (I_j, Q_j). The error vector $(\delta I_j, \delta Q_j)$ is defined as the distance from this ideal position to the actual position of the received point, as illustrated in Figure 65.

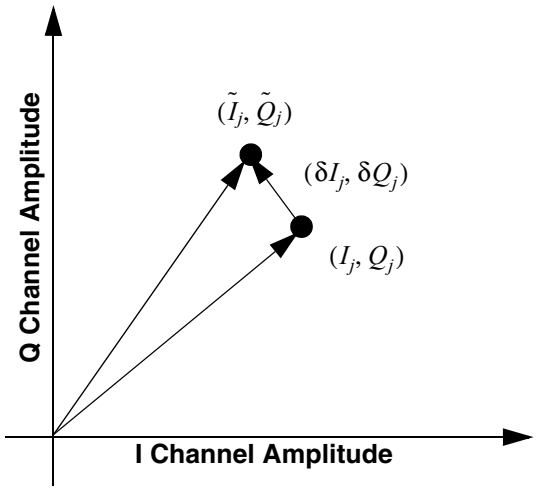


Figure 65—Error-vector calculation

Thus, the received vector is the sum of the ideal vector and the error vector as:

$$(\tilde{I}_j, \tilde{Q}_j) = (I_j, Q_j) + (\delta I_j, \delta Q_j)$$

The EVM is defined as:

$$\mathrm{EVM} \equiv \sqrt{\frac{\frac{1}{N}\sum_{j=1}^{N}(\delta I_j^2 + \delta Q_j^2)}{S^2}} \times 100\%$$

where

S is the magnitude of the vector to the ideal constellation point (for PSSS in 915/868 MHz S is the ASK step size, and the PHR and PHY payload should be set to 0 for testing)

$(\delta I_j, \delta Q_j)$ is the error vector

The error-vector measurement shall be made on baseband I and Q chips after recovery through a reference receiver system. The reference receiver shall perform carrier lock, symbol timing recovery, and amplitude adjustment while making the measurements.

8.2.4 Receiver maximum input level of desired signal

The receiver maximum input level is the maximum power level of the desired signal present at the input of the receiver for which the error rate criterion in 8.1.7 is met.

8.2.5 Receiver ED

The receiver ED measurement is intended for use by a network layer as part of a channel selection algorithm. It is an estimate of the received signal power within the bandwidth of the channel. No attempt is made to identify or decode signals on the channel. The ED measurement time, to average over, shall be equal to 8 symbol periods.

The ED result shall be reported to the next higher layer using MLME-SCAN.confirm. The minimum ED value (zero) shall indicate received power less than 10 dB above the maximum allowed receiver sensitivity for the PHY. The range of received power spanned by the ED values shall be at least 40 dB. Within this range, the mapping from the received power in decibels to ED value shall be linear with an accuracy of ± 6 dB.

8.2.6 Link quality indicator (LQI)

The LQI measurement is a characterization of the strength and/or quality of a received packet. The measurement may be implemented using receiver ED, a signal-to-noise ratio estimation, or a combination of these methods. The use of the LQI result by the network or application layers is not specified in this standard.

The LQI measurement shall be performed for each received packet. The minimum and maximum LQI values (0x00 and 0xff) should be associated with the lowest and highest quality compliant signals detectable by the receiver, and LQI values in between should be uniformly distributed between these two limits. At least eight unique values of LQI shall be used.

8.2.7 Clear channel assessment (CCA)

The PHY shall provide the capability to perform CCA according to at least one of the following methods:

- *CCA Mode 1: Energy above threshold.* CCA shall report a busy medium upon detecting any energy above the ED threshold.
- *CCA Mode 2: Carrier sense only.* CCA shall report a busy medium only upon the detection of a signal compliant with this standard with the same modulation and spreading characteristics of the PHY that is currently in use by the device. This signal may be above or below the ED threshold.
- *CCA Mode 3: Carrier sense with energy above threshold.* CCA shall report a busy medium using a logical combination of:
  - Detection of a signal with the modulation and spreading characteristics of this standard, and
  - Energy above the ED threshold, where the logical operator may be AND or OR.
- *CCA Mode 4: ALOHA.* CCA shall always report an idle medium.
- *CCA Mode 5: UWB preamble sense based on the SHR of a frame.* CCA shall report a busy medium upon detection of a preamble symbol as specified in 14.2.5. An idle channel shall be reported if no preamble symbol is detected up to a period not shorter than the maximum packet duration plus the maximum period for acknowledgment.

— *CCA Mode 6: UWB preamble sense based on the packet with the multiplexed preamble as specified in 14.6.* CCA shall report a busy medium upon detection of a preamble symbol as specified in 14.2.5.

For any of the CCA modes, if a request to perform CCA is received by the PHY during reception of a PPDU, CCA shall report a busy medium. PPDU reception is considered to be in progress following detection of the SFD, and it remains in progress until the number of octets specified by the decoded PHR has been received.

The PHY PIB attribute *phyCCAMode*, as described in 9.3, shall indicate the appropriate operation mode. The CCA parameters are subject to the following criteria:

a) The ED threshold shall correspond to a received signal power of at most 10 dB greater than the specified receiver sensitivity for that PHY.
b) The CCA detection time shall be equal to 8 symbol periods or *phyCCADuration* symbol periods for the 950 MHz band PHY.

For the 950 MHz band, if channel 14 is supported, CCA shall be performed on channel 13 and channel 14. If channel 17 is supported, CCA shall be performed on channel 16 and channel 17 (English translation of ARIB STD-T96 [B6]).

9. PHY services

9.1 Overview

The PHY provides an interface between the MAC sublayer and the physical radio channel, via the RF firmware and the RF hardware. The PHY conceptually includes a management entity called the PLME. This entity provides the layer management service interfaces through which layer management functions may be invoked. The PLME is also responsible for maintaining a database of managed objects pertaining to the PHY. This database is referred to as the PHY PAN information base (PIB).

Figure 66 depicts the components and interfaces of the PHY.

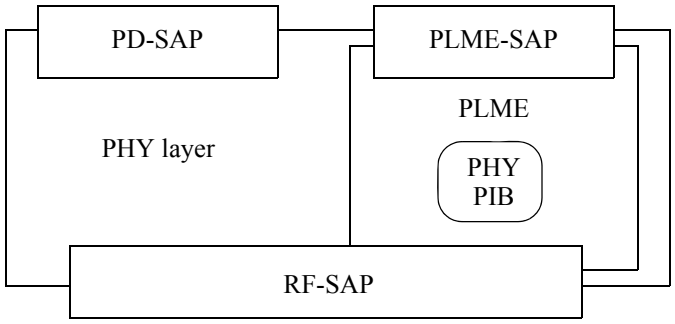


Figure 66—PHY reference model

The PHY provides two services, accessed through two SAPs: the PHY data service, accessed through the PHY data SAP (PD-SAP), and the PHY management service, accessed through the PLME-SAP. The PD-SAP and PLME-SAP are not defined in this standard as they are not expected to be exposed in a typical implementation. The PHY PIB attributes are accessed through the MLME SAP with the MLME-GET and MLME-SET primitives.

Constants and attributes that are specified and maintained by the PHY are written in the text of this clause in italics. Constants have a general prefix of "a", e.g., *aMaxPHYPacketSize*, and are listed in Table 70. Attributes have a general prefix of "phy", e.g., *phyCurrentChannel*, and are listed in Table 71.

Attributes that have a prefix of "phyUWB", e.g., *phyUWBDataRatesSupported*, apply only to the UWB PHY and are not used for other PHYs.

9.2 PHY constants

The constants that define the characteristics of the PHY are presented in Table 70. These constants are hardware dependent and cannot be changed during operation.

Table 70—PHY constants

Constant	Description	Value
aMaxPHYPacketSize	The maximum PSDU size (in octets) the PHY shall be able to receive.	127
aTurnaroundTime	RX-to-TX or TX-to-RX turnaround time (in symbol periods), as defined in 8.2.1 and 8.2.2.	12

9.3 PHY PIB attributes

The PHY PIB comprises the attributes required to manage the PHY of a device. The attributes contained in the PHY PIB are presented in Table 71. Attributes marked with a dagger (†) are read-only attributes (i.e., attribute can only be set by the PHY), which can be read by the next higher layer using the MLME-GET.request primitive. All other attributes can be read or written by the next higher layer using the MLME-GET.request or MLME-SET.request primitives, respectively.

Table 71—PHY PIB attributes

Attribute	Type	Range	Description
phyCurrentChannel	Integer	As defined in 8.1.2	The RF channel to use for all following transmissions and receptions, 8.1.2.
phyChannelsSupported†	List of channel descriptions	—	Each entry in the list consists of a channel page and a list of channel numbers supported for that channel page.
phyTXPowerTolerance	Enumeration	1_dB, 3_dB, 6_dB	The tolerance on the transmit power setting, plus or minus the indicated value.
phyTXPower	Signed integer	—	The transmit power of the device in dBm.
phyCCAMode	Integer	1–6	The CCA mode, as defined in 8.2.7.
phyCurrentPage	Integer	Any valid channel page	This is the current PHY channel page. This is used in conjunction with *phyCurrentChannel* to uniquely identify the channel currently being used.
phyMaxFrameDuration†	Integer	—	The maximum number of symbols in a frame, as defined in 9.4.
phySHRDuration†	Integer	PHY dependent	The duration of the synchronization header (SHR) in symbols for the current PHY.
phySymbolsPerOctet†	Float	0.4, 1.3, 1.6, 2, 5.3, 8	The number of symbols per octet for the current PHY. For the UWB PHY this is defined in 14.2.3. For the CSS PHY, 1.3 corresponds to 1 Mb/s while 5.3 corresponds to 250 kb/s.
phyPreambleSymbol Length†	Integer	0, 1	Zero indicates preamble symbol length is 31, and one indicates that length 127 symbol is used. Present for UWB PHY.
phyUWBDataRates Supported†	List of integers	—	A list of the data rates available in the operating channel as defined in Table 105.
phyCSSLowDataRate Supported†	Boolean	TRUE, FALSE	A value of TRUE indicates that 250 kb/s is supported. Present for CSS PHY.
phyUWBCoU Supported†	Boolean	TRUE, FALSE	TRUE if CoU pulses are supported, FALSE otherwise.
phyUWBCS Supported†	Boolean	TRUE, FALSE	TRUE if CS pulses are supported, FALSE otherwise.
phyUWBLCP Supported†	Boolean	TRUE, FALSE	TRUE if LCP pulses are supported, FALSE otherwise.

Table 71—PHY PIB attributes *(continued)*

Attribute	Type	Range	Description
phyUWBCurrentPulse Shape	Enumeration	MANDATORY, COU, CS, LCP	Indicates the current pulse shape setting of the UWB PHY. The mandatory pulse is described in 14.4.5. Optional pulse shapes include CoU, as defined in 14.5.1, CS, as defined in 14.5.2, and LCP, as defined in 14.5.3.
phyUWBCoUpulse	Enumeration	CCh.1, CCh.2, CCh.3, CCh.4, CCh.5, CCh.6	Defines the slope of the frequency chirp and bandwidth of pulse. CCh.3–CCh.6 are valid only for wideband UWB channels, e.g., 4, 7, 11, or 15, as defined in 14.5.1.
phyUWBCSpulse	Enumeration	No.1, No.2, No.3, No.4, No.5, No.6	Defines the group delay of the continuous spectrum filter. No.3–No.6 are valid only for wideband UWB channels, e.g., 4, 7, 11, or 15, as described in 14.5.2.
phyUWBLCPWeight1	Signed integer	0x00–0xff	The weights are represented in twos-complement form. A value of 0x80 represents –1 while a value of 0x7F represents 1.
phyUWBLCPWeight2	Signed integer	0x00–0xff	The weights are represented in twos-complement form. A value of 0x80 represents –1 while a value of 0x7F represents 1.
phyUWBLCPWeight3	Signed integer	0x00–0xff	The weights are represented in twos-complement form. A value of 0x80 represents –1 while a value of 0x7f represents 1.
phyUWBLCPWeight4	Signed integer	0x00–0xff	The weights are represented in twos-complement form. A value of 0x80 represents –1 while a value of 0x7f represents 1.
phyUWBLCPDelay2	Integer	0x00–0xff	The range is from 0 to 4 ns with a resolution is 4/255 = 15.625 ps. For example, a value of 0x00 represents 0 while 0x02 represents 31.25 ps, as defined in 14.5.3.
phyUWBLCPDelay3	Integer	0x00–0xff	The range is from 0 to 4 ns with a resolution is 4/255 = 15.625 ps. For example, a value of 0x00 represents 0 while 0x02 represents 31.25 ps, as defined in 14.5.3.
phyUWBLCPDelay4	Integer	0x00–0xff	The range is from 0 to 4 ns with a resolution is 4/255 = 15.625 ps. For example, a value of 0x00 represents 0 while 0x02 represents 31.25 ps, as defined in 14.5.3.
phyRanging†	Boolean	TRUE, FALSE	TRUE if ranging is supported, FALSE otherwise.
phyRangingCrystal Offset†	Boolean	TRUE, FALSE	TRUE if crystal offset characterization is supported, FALSE otherwise.
phyRangingDPS†	Boolean	TRUE, FALSE	TRUE if DPS is supported, FALSE otherwise.
phyCurrentCode	Integer	0–24	This value is zero for PHYs other than UWB or CSS. For UWB PHYs, this represents the current preamble code index in use by the transmitter, as defined in Table 102 and Table 103. For the CSS PHY, the value indicates the subchirp, as defined in 13.3.

Table 71—PHY PIB attributes *(continued)*

Attribute	Type	Range	Description
phyNativePRF	Enumeration	0–3	For UWB PHYs, the native PRF. Zero is for non-UWB PHYs; one is for PRF of 4; two is for a PRF of 16; and three is for PHYs that have no preference.
phyUWBScanBins PerChannel	Integer	0–255	Number of frequency intervals used to scan each UWB channel (scan resolution). Set to zero for non-UWB PHYs.
phyUWBInserted PreambleInterval	Enumeration	0, 4	The time interval between two neighboring inserted preamble symbols in the data portion, as defined in 14.6, for UWB PHYs operating with CCA mode 6. The resolution is a data symbol duration at a data rate of 850 kb/s for all channels. Set to four for UWB PHY in CCA mode 6; otherwise, set to zero.
phyTXRMARKER Offset	Integer	0x00000000–0xffffffff	A count of the propagation time from the ranging counter to the transmit antenna. The LSB of a time value represents 1/128 of a chip time at the mandatory chipping rate of 499.2 MHz.
phyRXRMARKER Offset	Integer	0x00000000–0xffffffff	A count of the propagation time from the receive antenna to the ranging counter. The LSB of a time value represents 1/128 of a chip time at the mandatory chipping rate of 499.2 MHz.
phyRFRAME ProcessingTime	Integer	0x00–0xff	A count of the processing time required by the PHY to handle an arriving RFRAME. The LSB represents 2 ms. The meaning of the value is that if a sequence of RFRAMEs arrive separated by *phyRFRAMEProcessingTime*, then the PHY can keep up with the processing indefinitely.
phyCCADuration	Integer	0-1000	The duration for CCA, specified in symbols. This attribute shall only be implemented with PHYs operating in the 950 MHz band.

9.4 PHY PIB attribute values for *phyMaxFrameDuration* and *phySHRDuration*

For PHYs other than CSS and UWB, the attribute *phyMaxFrameDuration* is given by:

$$phyMaxFrameDuration = phySHRDuration + \text{ceiling}([aMaxPHYPacketSize + 1] \times phySymbolsPerOctet)$$

where ceiling() is a function that returns the smallest integer value greater than or equal to its argument value.

For the CSS PHY type, the values of the attribute *phyMaxFrameDuration* depend on the selected data rate of the PPDU.

For the mandatory data rate (1 Mb/s), *phyMaxFrameDuration* is calculated as follows:

$$phyMaxFrameDuration_{1M} = phySHRDuration1M + [1.5 + 3/4 \times \text{ceiling}(4/3 \times aMaxPHYPacketSize)] \times phySymbolsPerOctet_{1M}$$

For the optional data rate (250 kb/s), *phyMaxFrameDuration* is calculated as follows:

$$phyMaxFrameDuration_{250k} = phySHRDuration_{250k} + 3 \times \text{ceiling}(1/3 \times [1.5 + aMaxPHYPacketSize]) \times phySymbolsPerOctet_{250k}$$

For the UWB PHY types, the values for the PIB attributes *phyMaxFrameDuration* and *phySHRDuration* vary depending upon the UWB PHY operating mode. The symbol duration varies by data rate and is different for preamble symbols and body symbols. Also note that the preamble and PHR are sent at a different data rate from the body.

$$phyMaxFrameDuration = T_{SHR} + T_{PHR} + T_{PSDU} + T_{CCApreamble}$$

$$T_{SHR} = T_{psym} \times (N_{preamblesymbols} + N_{SFD})$$

$$T_{PHR} = N_{PHR} \times T_{dsym1M}$$

$$T_{PSDU} = \left\lceil T_{dsym} \times \frac{N_{PSDUoctets} \times N_{symPerOctet}}{R_{FEC}} \right\rceil$$

$$T_{CCApreamble} = \left\lceil T_{psym} \times \frac{T_{PHR} + T_{PSDU}}{4 \times T_{dsym1M}} \right\rceil$$

where

T_{psym}	is the base symbol time for preamble symbols for the selected channel, as defined in Table 101
T_{dsym}	is the PSDU data symbol duration, as defined in Table 99
T_{dsym1M}	is the nominal 1 Mb/s data symbol duration for the selected channel, as defined in Table 99
$N_{preamblesymbols}$	is the UWBPreambleSymbolRepetitions
N_{SFD}	is the number of delimiter symbols, as defined in 14.2.5.2
N_{PHR}	is the number of bits in the PHR
$N_{PSDUoctets}$	is set to *aMaxPHYPacketSize*
$N_{symPerOctet}$	is the number of symbols per octet, uncoded = 8
R_{FEC}	is the Reed-Solomon FEC rate, as described in Table 99, includes coded bits/ symbol when convolutional code is used)

The PHY PIB attribute *phySHRDuration* is given by:

$$phySHRDuration = N_{preamblesymbols} + N_{SFD}$$

10. O-QPSK PHY

10.1 PPDU format

For convenience, the PPDU structure is presented so that the leftmost field as written in this standard shall be transmitted or received first. All multiple octet fields shall be transmitted or received least significant octet first, and each octet shall be transmitted or received least significant bit (LSB) first.

The PPDU shall be formatted as illustrated in Figure 67.

		Octets		
		1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figure 67—Format of the PPDU

10.1.1 Preamble field

The length of the preamble for the O-QPSK PHYs shall be 8 symbols (i.e., 4 octets), and the bits in the Preamble field shall be binary zeros.

10.1.2 SFD field

The SFD is a field indicating the end of the SHR and the start of the packet data. The SFD shall be formatted as illustrated in Figure 68.

Bits: 0	1	2	3	4	5	6	7
1	1	1	0	0	1	0	1

Figure 68—Format of the SFD field

10.1.3 Frame Length field

The Frame Length field specifies the total number of octets contained in the PSDU (i.e., PHY payload). It is a value between 0 and *aMaxPHYPacketSize*, as described in 9.2. Table 72 summarizes the type of payload versus the frame length value

10.1.4 PSDU field

The PSDU field carries the data of the PPDU.

Table 72—Frame length values

Frame length values	Payload
0–4	Reserved
5	MPDU (Acknowledgment)
6–8	Reserved
9 to *aMaxPHYPacketSize*	MPDU

10.2 Modulation and spreading

The O-QPSK PHY employs a 16-ary quasi-orthogonal modulation technique. During each data symbol period, four information bits are used to select 1 of 16 nearly orthogonal pseudo-random noise (PN) sequences to be transmitted. The PN sequences for successive data symbols are concatenated, and the aggregate chip sequence is modulated onto the carrier using offset quadrature phase-shift keying (O-QPSK).

10.2.1 Data rate

The data rate of the O-QPSK PHY shall be 250 kb/s when operating in the 2450 MHz, 915 MHz, or 780 MHz bands and shall be 100 kb/s when operating in the 868 MHz band.

Support for the 2450 MHz O-QPSK PHY is mandatory when operating in the 2450 MHz band.

The O-QPSK PHY is not mandatory in the 868 MHz or 915 MHz band. If the O-QPSK PHY is used in the 868 MHz or 915 MHz band, then the same device shall be capable of signaling using the BPSK PHY as well.

10.2.2 Reference modulator diagram

The functional block diagram in Figure 69 is provided as a reference for specifying the O-QPSK PHY modulation and spreading functions.

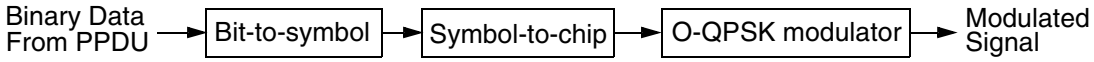


Figure 69—Modulation and spreading functions for the O-QPSK PHYs

10.2.3 Bit-to-symbol mapping

All binary data contained in the PPDU shall be encoded using the modulation and spreading functions shown in Figure 69. This subclause describes how binary information is mapped into data symbols.

The 4 LSBs (b_0, b_1, b_2, b_3) of each octet shall map into one data symbol, and the 4 MSBs (b_4, b_5, b_6, b_7) of each octet shall map into the next data symbol. Each octet of the PPDU is processed through the modulation and spreading functions, as illustrated in Figure 69, sequentially, beginning with the Preamble field and ending with the last octet of the PSDU. Within each octet, the least significant symbol (b_0, b_1, b_2, b_3) is processed first and the most significant symbol (b_4, b_5, b_6, b_7) is processed second.

10.2.4 Symbol-to-chip mapping

In the 2450 MHz band, each data symbol shall be mapped into a 32-chip PN sequence as specified in Table 73. The PN sequences are related to each other through cyclic shifts and/or conjugation (i.e., inversion of odd-indexed chip values).

Table 73—Symbol-to-chip mapping for the 2450 MHz band

Data symbol	Chip values (c_0 c_1 … c_{30} c_{31})
0	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

In the 915 MHz, 868 MHz, and 780 MHz bands, each data symbol shall be mapped into a 16-chip PN sequence as specified in Table 74.

Table 74—Symbol-to-chip mapping for the 915 MHz, 868 MHz, and 780 MHz bands

Data symbol	Chip values (c_0 c_1 … c_{14} c_{15})
0	0 0 1 1 1 1 1 0 0 0 1 0 0 1 0 1
1	0 1 0 0 1 1 1 1 1 0 0 0 1 0 0 1
2	0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0
3	1 0 0 1 0 1 0 0 1 1 1 1 1 0 0 0
4	0 0 1 0 0 1 0 1 0 0 1 1 1 1 1 0

Table 74—Symbol-to-chip mapping for the 915 MHz, 868 MHz, and 780 MHz bands *(continued)*

Data symbol	Chip values (c_0 c_1 … c_{14} c_{15})
5	1 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1
6	1 1 1 0 0 0 1 0 0 1 0 1 0 0 1 1
7	1 1 1 1 1 0 0 0 1 0 0 1 0 1 0 0
8	0 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0
9	0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 0
10	0 0 0 0 0 1 1 0 1 0 1 1 0 1 1 1
11	1 1 0 0 0 0 0 1 1 0 1 0 1 1 0 1
12	0 1 1 1 0 0 0 0 0 1 1 0 1 0 1 1
13	1 1 0 1 1 1 0 0 0 0 0 1 1 0 1 0
14	1 0 1 1 0 1 1 1 0 0 0 0 0 1 1 0
15	1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1

10.2.5 O-QPSK modulation

The chip sequences representing each data symbol are modulated onto the carrier using O-QPSK with half-sine pulse shaping. Even-indexed chips are modulated onto the in-phase (I) carrier, and odd-indexed chips are modulated onto the quadrature-phase (Q) carrier. In the 2450 MHz band, each data symbol is represented by a 32-chip sequence, and so the chip rate is 32 times the symbol rate. In the 915 MHz, 868 MHz, and 780 MHz bands, each data symbol is represented by a 16-chip sequence, and so the chip rate is 16 times the symbol rate. To form the offset between I-phase and Q-phase chip modulation, the Q-phase chips shall be delayed by T_c with respect to the I-phase chips, as illustrated in Figure 70, where T_c is the inverse of the chip rate.

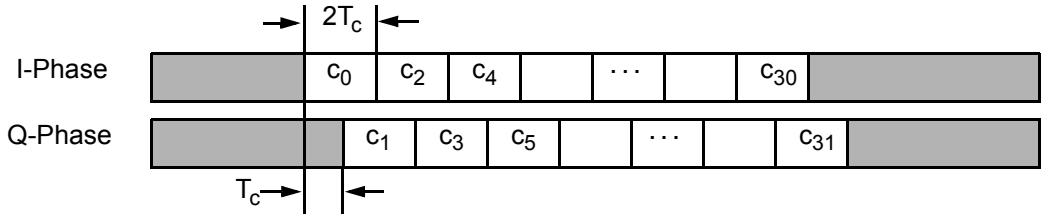


Figure 70—O-QPSK chip offsets

10.2.6 Pulse shape

In the 2450 MHz, 915 MHz, and 868 MHz bands, the half-sine pulse shape is used to represent each baseband chip and is given by:

$$p(t) = \begin{cases} \sin\left(\pi \frac{t}{2T_c}\right), & 0 \le t \le 2T_c \\ 0, & \text{otherwise} \end{cases}$$

Figure 71 shows a sample baseband chip sequence (the zero sequence) with half-sine pulse shaping.

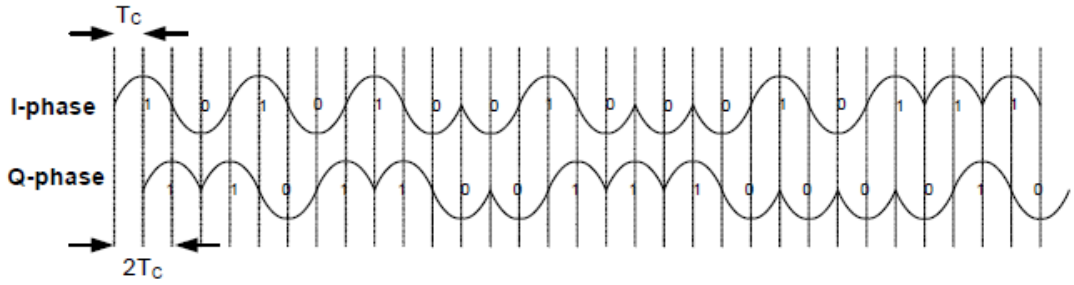


Figure 71—Sample baseband chip sequences with pulse shaping

In the 780 MHz band, a raised cosine pulse shape with roll-off factor of $r = 0.8$ is used to represent each baseband chip and is described by:

$$p(t) = \begin{cases} \dfrac{\sin(\pi t/T_c)}{\pi t/T_c} \times \dfrac{\cos(r\pi t/T_c)}{1-4r^2t^2/T_c^2}, & t \neq 0 \\ 1, & t = 0 \end{cases}$$

Given the discrete-time sequence of consecutive complex-valued chip samples, c_k, the continuous-time pulse shaped complex baseband signal is given by:

$$y(t) = \sum_{k=-\infty}^{\infty} c_{2k}p(t-2kT_c) + jc_{2k+1}p(t-2kT_c-T_c)$$

10.2.7 Chip transmission order

During each symbol period, the least significant chip, c_0, is transmitted first and the most significant chip, either c_{31}, for the 2450 MHz band, or c_{15}, for the 915 MHz, 868 MHz, and 780 MHz bands, is transmitted last.

10.3 O-QPSK PHY RF requirements

10.3.1 Operating frequency range

The O-QPSK PHY operates in the following bands:

— 779–787 MHz
— 868.0–868.6 MHz
— 902–928 MHz
— 2400.0–2483.5 MHz

10.3.2 Transmit power spectral density (PSD) mask

When operating in the 868 MHz band, the signal shall be filtered before transmission to regulate the transmit PSD. The filter shall approximate an ideal raised cosine filter with a roll-off factor $r = 0.2$, given as:

$$p(t) = \begin{cases} \frac{\sin \pi t/T_c}{\pi t/T_c} \frac{\cos r\pi t/T_c}{1-4r^2t^2/T_c^2}, & t \neq 0 \\ 1, & t = 0 \end{cases}$$

When operating in the 915 MHz or 780 MHz band, the transmitted spectral products shall be less than the limits specified in Table 75. For both relative and absolute limits, average spectral power shall be measured using a 100 kHz resolution bandwidth. For the relative limit, the reference level shall be the highest average spectral power measured within ±600 kHz of the carrier frequency f_c.

Table 75—915 MHz and 780 MHz band O-QPSK PHY transmit PSD limits

Frequency	Relative limit	Absolute limit
$\|f - f_c\| > 1.2$ MHz	–20 dB	–20 dBm

When operating in the 2450 MHz band, the transmitted spectral products shall be less than the limits specified in Table 76. For both relative and absolute limits, average spectral power shall be measured using a 100 kHz resolution bandwidth. For the relative limit, the reference level shall be the highest average spectral power measured within ±1 MHz of the carrier frequency.

Table 76—2450 MHz band O-QPSK transmit PSD limits

Frequency	Relative limit	Absolute limit
$\|f - f_c\| > 3.5$ MHz	–20 dB	–30 dBm

10.3.3 Symbol rate

The O-QPSK PHY symbol rate shall be 25 ksymbol/s when operating in the 868 MHz band and 62.5 ksymbol/s when operating in the 780 MHz, 915 MHz, or 2450 MHz band with an accuracy of ±40 ppm.

10.3.4 Receiver sensitivity

Under the conditions specified in 8.1.7, a compliant device shall be capable of achieving a receiver sensitivity of –85 dBm or better.

10.3.5 Receiver interference rejection

This subclause applies only to the 780 MHz, 915 MHz, and 2450 MHz bands as there is only one channel available in the 868 MHz band.

The minimum receiver interference rejection levels are given in Table 77. The adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel, and the alternate channel is one more removed from the adjacent channel. For example, when channel 5 is the desired channel, channel 4 and channel 6 are the adjacent channels, and channel 3 and channel 7 are the alternate channels.

Table 77—Minimum receiver interference rejection requirements for the 780 MHz, 915 MHz, and 2450 MHz bands

Adjacent channel rejection	Alternate channel rejection
0 dB	30 dB

The adjacent channel rejection shall be measured as follows: the desired signal shall be a compliant O-QPSK PHY signal, as defined by 10.2, of pseudo-random data. The desired signal is input to the receiver at a level 3 dB greater than the maximum allowed receiver sensitivity given in 10.3.4.

In either the adjacent or the alternate channel, a compliant O-QPSK PHY signal, as defined by 10.2, is input at the level specified in Table 77 relative to the desired signal. The test shall be performed for only one interfering signal at a time. The receiver shall meet the error rate criteria defined in 8.1.7 under these conditions.

10.3.6 TX-to-RX turnaround time

The O-QPSK PHY shall have a TX-to-RX turnaround time as defined in 8.2.1.

10.3.7 RX-to-TX turnaround time

The O-QPSK PHY shall have an RX-to-TX turnaround time as defined in 8.2.2.

10.3.8 Error vector magnitude (EVM)

The O-QPSK PHY shall have EVM values of less than 35% when measured for 1000 chips using the measurement process defined in 8.2.3.

10.3.9 Transmit center frequency tolerance

The O-QPSK PHY transmit center frequency tolerance shall be ±40 ppm maximum.

10.3.10 Transmit power

The O-QPSK PHY shall be capable of transmitting at a power level of least –3 dBm.

10.3.11 Receiver maximum input level of desired signal

The O-QPSK PHY shall have a receiver maximum input level greater than or equal to –20 dBm using the measurement defined in 8.2.4.

10.3.12 Receiver ED

The O-QPSK PHY shall provide the receiver ED measurement as described in 8.2.5.

10.3.13 Link quality indicator (LQI)

The O-QPSK PHY shall provide the LQI measurement as described in 8.2.6.

10.3.14 Clear channel assessment (CCA)

The O-QPSK PHY shall use the one of the CCA modes as described in 8.2.7.

11. Binary phase-shift keying (BPSK) PHY

11.1 PPDU format

The BPSK PHY shall use the PPDU formats described in 10.1, except that the preamble is 32 symbols (4 octets).

11.2 Modulation and spreading

The BPSK PHY shall employ direct sequence spread spectrum (DSSS) with BPSK used for chip modulation and differential encoding used for data symbol encoding.

11.2.1 BPSK PHY data rates

The data rate of the BPSK PHY shall be 20 kb/s when operating in the 868/950 MHz band and 40 kb/s when operating in the 915 MHz band.

11.2.2 Reference modulator diagram

The functional block diagram in Figure 72 is provided as a reference for specifying the BPSK PHY modulation and spreading functions. Each bit in the PPDU shall be processed through the differential encoding, bit-to-chip mapping, and modulation functions in octet-wise order, beginning with the Preamble field and ending with the last octet of the PSDU. Within each octet, the LSB, b_0, is processed first and the MSB, b_7, is processed last.

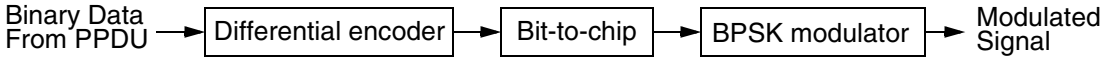


Figure 72—BPSK modulation and spreading functions

11.2.3 Differential encoding

Differential encoding is the modulo-2 addition (exclusive or) of a raw data bit with the previous encoded bit. This is performed by the transmitter and can be described by:

$$E_n = R_n \oplus E_{n-1}$$

where

R_n is the raw data bit being encoded
E_n is the corresponding differentially encoded bit
E_{n-1} is the previous differentially encoded bit

For each packet transmitted, R_1 is the first raw data bit to be encoded and E_0 is assumed to be zero.

Conversely, the decoding process, as performed at the receiver, can be described by:

$$R_n = E_n \oplus E_{n-1}$$

For each packet received, E_1 is the first bit to be decoded, and E_0 is assumed to be zero.

11.2.4 Bit-to-chip mapping

Each input bit shall be mapped into a 15-chip PN sequence as specified in Table 78.

Table 78—Symbol-to-chip mapping

Input bits	Chip values (c_0 c_1 … c_{14})
0	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0
1	0 0 0 0 1 0 1 0 0 1 1 0 1 1 1

11.2.5 BPSK modulation

The chip sequences are modulated onto the carrier using BPSK with raised cosine pulse shaping (roll-off factor = 1) where a chip value of one corresponds to a positive pulse and a chip value of zero corresponds to a negative pulse. The chip rate is 300 kchip/s for the 868/950 MHz band and 600 kchip/s in the 915 MHz band.

11.2.5.1 Pulse shape

The raised cosine pulse shape (roll-off factor = 1) used to represent each baseband chip is described by:

$$p(t) = \begin{cases} \dfrac{\sin \pi t / T_c}{\pi t / T_c} \dfrac{\cos \pi t / T_c}{1 - 4t^2 / T_c^2}, & t \neq 0 \\ 1, & t = 0 \end{cases}$$

11.2.5.2 Chip transmission order

During each symbol period, the least significant chip, c_0, is transmitted first, and the most significant chip, c_{15}, is transmitted last.

11.3 BPSK PHY RF requirements

11.3.1 Operating frequency range

The BPSK PHY operates in the following frequency bands:

— 868.0–868.6 MHz
— 902–928 MHz
— 950-956 MHz.

11.3.2 915/950 MHz band transmit PSD mask

The transmitted spectral products shall be less than the limits specified in Table 79. For the 915 MHz band, both relative and absolute limits, average spectral power shall be measured using a 100 kHz resolution bandwidth. For the relative limit, the reference level shall be the highest average spectral power measured within ±600 kHz of the carrier frequency.

Table 79—915/950 MHz band transmit PSD limits

Frequency band	Frequency	Relative limit	Absolute limit
915 MHz	$\|f - f_c\| > 1.2$ MHz	–20 dB	–20 dBm
950 MHz (1 mW channels)	0.5 MHz > $\|f - f_c\| > 0.3$ MHz	—	–26 dBm/200 kHz
	$\|f - f_c\| > 0.5$ MHz	—	–39 dBm
950 MHz (10 mW channels)	0.5 MHz > $\|f - f_c\| > 0.3$ MHz	—	–18 dBm/200 kHz
	$\|f - f_c\| > 0.5$ MHz	—	–39 dBm

11.3.3 Symbol rate

The symbol rate of a BPSK PHY conforming to this standard shall be 20 ksymbol/s when operating in the 868/950 MHz band and 40 ksymbol/s when operating in the 915 MHz band with an accuracy of ±40 ppm.

NOTE—Current regulations for 950 MHz band PHY requires more stringent accuracy. Implementations need to comply with the current regulatory requirements.

11.3.4 Receiver sensitivity

Under the conditions specified in 8.1.7, a compliant device shall be capable of achieving a receiver sensitivity of –92 dBm or better.

11.3.5 Receiver interference rejection

This subclause applies only to the 915 MHz and the 950 MHz band as there is only one channel available in the 868 MHz band.

The minimum receiver interference rejection levels are given in Table 80.

Table 80—Minimum receiver interference rejection requirements for the 915/950 MHz BPSK PHYs

Frequency band	Adjacent channel rejection	Alternate channel rejection
915 MHz band	0 dB	30 dB
950 MHz band	0 dB	24 dB

For the 915 MHz band, the adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel, and the alternate channel is one more removed from the adjacent channel. For example, when channel 5 is the desired channel, channel 4 and channel 6 are the adjacent channels, and channel 3 and channel 7 are the alternate channels.

For the 950 MHz band, the adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel and has a distance of at least 0.6 MHz to the desired channel. The alternate channel is one more removed from the adjacent channel and has a distance of at least 1.2 MHz to the desired channel.

The adjacent channel rejection shall be measured as follows: the desired signal shall be a compliant 915/950 MHz BPSK PHY signal, as defined by 11.2, of pseudo-random data. The desired signal is input to the receiver at a level 3 dB greater than the maximum allowed receiver sensitivity given in 11.3.4.

In either the adjacent or the alternate channel, a compliant 915/950 MHz BPSK PHY signal, as defined by 11.2, is input at the relative level specified in Table 80. The test shall be performed for only one interfering signal at a time. The receiver shall meet the error rate criteria defined in 8.1.7 under these conditions.

11.3.6 TX-to-RX turnaround time

The BPSK PHY shall have a TX-to-RX turnaround time as defined in 8.2.1.

11.3.7 RX-to-TX turnaround time

The BPSK PHY shall have an RX-to-TX turnaround time as defined in 8.2.2.

11.3.8 Error vector magnitude (EVM)

The BPSK PHY shall have EVM values of less than 35% when measured for 1000 chips using the measurement process defined in 8.2.3.

11.3.9 Transmit center frequency tolerance

The BPSK PHY transmit center frequency tolerance shall be ±40 ppm maximum.

11.3.10 Transmit power

The BPSK PHY shall be capable of transmitting at a power level of least –3 dBm.

11.3.11 Receiver maximum input level of desired signal

The BPSK PHY shall have a receiver maximum input level greater than or equal to –20 dBm using the measurement defined in 8.2.4.

11.3.12 Receiver ED

The BPSK PHY shall provide the receiver ED measurement as described in 8.2.5.

11.3.13 Link quality indicator (LQI)

The BPSK PHY shall provide the LQI measurement as described in 8.2.6.

11.3.14 Clear channel assessment (CCA)

The BPSK PHY shall use the one of the CCA modes as described in 8.2.7.

12. Amplitude shift keying (ASK) PHY

12.1 PPDU format

The ASK PHY shall use the PPDU formats described in 10.1, except that the SHR is defined differently.

The SHR uses a subset of the PSSS coding used for the PHR and PSDU. The SHR chips shall be transmitted with BPSK modulation using the same chip rates, as described in 12.2.5, and pulse shaping, as described in 12.2.6, that are used for the PHR and PSDU. However, the symbol-to-chip mapping for the synchronization header is different, as described in 12.1.1 and 12.1.2.

12.1.1 Preamble for ASK PHY

The preamble is generated by repeating 2 times for 868 MHz and repeating 6 times for 915 MHz sequence number 0 from Table 81 and Table 82, respectively. The resulting preamble duration is 160 μs for 868 MHz and 120 μs for 915 MHz. The leftmost chip number "0" in the diagram, with a value of "–1", is transmitted first. The preamble will be BPSK modulated as shown in Figure 73. The pulse shaping is the same as for the PHY payload as defined in 12.2.6.

12.1.2 SFD for ASK PHY

The SFD for 868 MHz and 915 MHz is the inverted sequence 0 from Table 81 and Table 82, respectively. The SFD will be BPSK modulated as shown in Figure 73. The pulse shaping is the same as for the PHY payload as defined in 12.2.6.

The preamble length for the ASK PHY shall be 2 symbols (5 octets). The SFD for the ASK PHY shall be 0.625 octets (1 symbol).

The preamble length and SFD for the ASK PHY is expressed in equivalent octet times as the ASK is defined using a special symbol.

12.2 Modulation and spreading

The ASK PHY employs a multi-code modulation technique named parallel sequence spread spectrum (PSSS), which is also more generically known as orthogonal code division multiplexing (OCDM). During each data symbol period, 20 information bits for 868 MHz and 5 information bits for 915 MHz are separately modulated onto 20 and 5 nearly orthogonal PN sequences, respectively. The 20 for 868 MHz and 5 for 915 MHz PN sequences are linearly summed to create a multi-level 32-chip symbol, equal to a 64-half-chip symbol for 868 MHz and a multi-level 32-chip symbol for 915 MHz. A simple precoding is then executed per symbol, and the resulting multi-level 64-half-chip sequence for 868 MHz and multi-level 32-chip sequence for 915 MHz are modulated onto the carrier using ASK.

12.2.1 ASK PHY data rates

The data rate of the ASK PHY shall be 250 kb/s when operating in the 868 MHz band and in the 915 MHz band. The ASK PHY is not mandatory in the 868 MHz or 915 MHz band. If the ASK PHY is used in the 868 MHz or 915 MHz band, then the same device shall be capable of signaling using the BPSK PHY as well.

12.2.2 Reference modulator diagram

The functional block diagram in Figure 73 is provided as a reference for specifying the PSSS modulation and spreading functions.

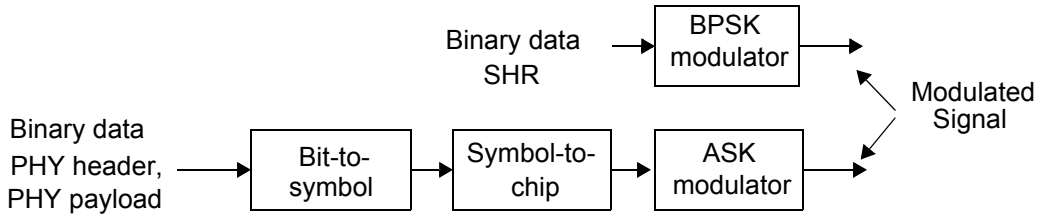


Figure 73—Modulation and spreading functions

Each octet of the PPDU is sequentially processed through the spreading and modulation functions. The synchronization header shall be encoded using the BPSK modulator, and the PHY header and PHY payload shall be processed using the bit-to-symbol mapping, and symbol-to-chip spreading and the ASK modulator as shown in Figure 73.

12.2.3 Bit-to-symbol mapping

The first 20 bits for 868 MHz and the first 5 bits for 915 MHz, starting with the LSB (b_0) of the first octet of the PHY header and continuing with the subsequent octet of the PHY header, shall be mapped into the first data symbol. Further 20 bits for 868 MHz and further 5 bits for 915 MHz continuing until the end of the PPDU shall be mapped sequentially to each subsequent data symbol until all octets of the PHY header and PHY payload have been mapped into symbols, always mapping the LSBs of any octet first. For each symbol, the least significant chip shall be the first chip transmitted over the air. The last input bits shall be followed by padding, with "0" bits, of its high order bits to fill out the symbol. These zero pad bits will also be spread via the PSSS encoding to keep the over the air signaling as random as possible.

12.2.4 Symbol-to-chip mapping

Each data symbol shall be mapped into a multi-level 64-half-chip symbol for 868 MHz and a multi-level 32-chip symbol for 915 MHz as described in this subclause. Figure 74 provides an overview of the symbol-to-chip mapping.

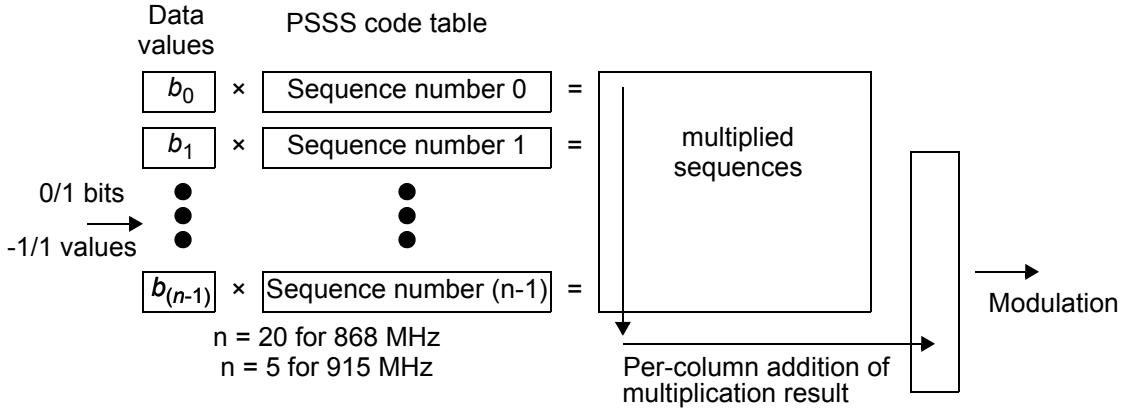


Figure 74—Symbol-to-chip mapping for PHY header and PHY payload

Each bit of the data stream is multiplied with its corresponding sequence of the code table defined in Table 81 for 868 MHz and Table 82 for 915 MHz. The PSSS code tables were generated by selecting 20 for 868 MHz and 5 for 915 MHz cyclically shifted sequences of a 31-chip base sequence and then adding a 1-bit cyclic extension to each sequence. For 868 MHz PSSS, the 31-chip base sequence is cyclically shifted by 1.5 chips to generate the next sequence in the table. For 915 MHz PSSS, the 31-chip base sequence is cyclically shifted by 6 chips to generate the next sequence in the table.

The vector of bits comprising the data symbol is multiplied with the PSSS code table. In other words, bit b_0 of the data symbol is multiplied with sequence number "0", bit b_1 is multiplied with sequence number "1", etc. Prior to multiplication, the data bits are converted to bipolar levels: data bit "1" becomes +1 and data bit "0" becomes –1. The result of multiplication is a modulated code table, which is similar to Table 81 or Table 82, but with each row either inverted or not according to the data bits.

Subsequently, all rows of the modulated code table are linearly summed to create a multi-level 64-half-chip symbol for 868 MHz and multi-level 32-chip symbol for 915 MHz. For example, chip 0 of the multi-level sequence is produced by linearly summing chip 0 from each of the modulated sequences. The per-column results are 32 chips $c_0 \ldots c_{31}$ for 915 MHz and 64 half-chips $c_0 \ldots c_{63}$ for 868 MHz.

Next, a precoding operation is applied to the multi-level 32-chip symbol for 915 MHz and the multi-level 64-half-chip symbol for 868 MHz. The precoding is independent from one symbol to the next and is performed in two steps. In the first step, a constant value is added to each of the 32 chips for 915 MHz and 64 half-chips for 868 MHz. The constant is selected so that the minimum and maximum values of the resulting sequence are symmetric about zero. Representing the original multi-level 32-chip symbol sequence by $p(m)$, then the modified sequence $p'(m)$ after step one of precoding is:

$$p'(m) = p(m) - \frac{(\mathrm{Max} + \mathrm{Min})}{2}$$

In the second step, a scaling constant is multiplied by each chip for 915 MHz and half-chip for 868 MHz in $p'(m)$. The scaling constant is selected such that the resulting sequence has a maximum amplitude of one. Letting $p''(m)$ be the output of the second precoding step, then:

$$p''(m) = \frac{p'(m)}{\mathrm{Max}'}$$

where Max' is the maximum within $p'(m)$. An example is given in 12.3.15.

The precoded sequence of 32 multi-level chips for 915 MHz and 64 multi-value half-chips for 868 MHz is modulated onto the carrier as described in 12.2.5.

12.2.5 ASK modulation

The chip sequences representing each data symbol are modulated onto the carrier using ASK with square root raised cosine pulse shaping. The chip rate is 400 kchip/s, equal to 800 kHalf-chip/s for 868 MHz. The chip rate is 1600 kchip/s for 915 MHz.

12.2.6 Pulse shape

The root-raised-cosine pulse shape used to represent each baseband chip is described by:

Table 81—PSSS code table used in symbol-to-chip mapping for 868 MHz

Sequence number	Chip number																																																															
	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31	
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		1		-1		1		-1		1		-1	
1	-1	1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1
2	1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1	
3	1	-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1
4	1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1	
5	-1	1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1
6	1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1	
7	-1	1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1
8	-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1	
9	1	-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1
10	1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1	
11	1	1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1		1		1
12	1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1		1	
13	-1	-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1		-1		-1
14	1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1		1	
15	-1	1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1		-1		1
16	1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1		1	
17	-1	-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1		-1		-1
18	1		-1		-1		1		-1		1		1		-1		-1		1		1		1		1		1		-1		-1		-1		1		1		-1		1		1		1		-1		1		-1		1		-1		-1		-1		-1		1	
19	-1	-1		1		-1		-1		1		-1		1		1		-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	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	half-chip number																																																															

Table 82—PSSS code table used in symbol-to-chip mapping for 915 MHz

Sequence number	Chip number 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	1	-1	1	-1	1	-1
1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	1	1	1	-1	-1	-1	1	1	-1	1	1
2	-1	-1	1	1	-1	1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	1	1	1	-1	-1
3	1	1	1	1	1	-1	-1	-1	1	1	-1	1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1	-1	1
4	1	-1	1	1	-1	-1	1	1	1	1	1	-1	-1	-1	1	1	-1	1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1

$$h(t)=\begin{cases}\dfrac{\left\{\pi(r+1)\cdot\sin\left(\dfrac{\pi}{4}\dfrac{(r+1)}{r}\right)+\pi(r-1)\cdot\cos\left(\dfrac{\pi}{4}\dfrac{(r-1)}{r}\right)-4r\cdot\sin\left(\dfrac{\pi}{4}\dfrac{(r-1)}{r}\right)\right\}}{2\pi\sqrt{T_c}}, t=(\pm T_c/(4r)) \\ \dfrac{4r}{\pi\sqrt{T_c}}-\dfrac{(r-1)}{\sqrt{T_c}}, t=0 \\ 4r\dfrac{\cos((1+r)\pi t/T_c)+\sin((1-r)\pi t/T_c)/(4rt/T_c)}{\pi\sqrt{T_c}(1-(4rt/T_c)^2)}, t\neq 0 \text{ and } t\neq(\pm T_c/(4r))\end{cases}$$

with a roll-off factor $r = 0.2$ for 868 MHz and 915 MHz. The pulse for stimulating the pulse shaping filter will be generated at chip rate/half-chip rate for 915/868 MHz. T_c is the chip duration (not half-chip) for 868 MHz and 915 MHz.

12.2.7 Chip transmission order

During each symbol period, the least significant chip/half-chip, c_0/hc_0, is transmitted first and the most significant chip, c_{31}/hc_{63}, is transmitted last.

12.3 ASK PHY RF requirements

12.3.1 Operating frequency range

The ASK PHY operates in the 868.0–868.6 MHz frequency band and in the 902–928 MHz frequency band.

12.3.2 915 MHz band transmit PSD mask

The transmitted spectral products shall be less than the limits specified in Table 83. For both relative and absolute limits, average spectral power shall be measured using a 100 kHz resolution bandwidth. For the relative limit, the reference level shall be the highest average spectral power measured within ±600 kHz of the carrier frequency.

Table 83—915 MHz band ASK PHY transmit PSD limits

Frequency	Relative limit	Absolute limit
$\lvert f-f_c\rvert > 1.2$ MHz	–20 dB	–20 dBm

12.3.3 Symbol rate

The ASK PHY symbol rate shall be 12.5 ksymbol/s ± 40 ppm for 868 MHz and 50 ksymbol/s ± 40 ppm for 915 MHz.

12.3.4 Receiver sensitivity

Under the conditions specified in 8.1.7, a compliant device shall be capable of achieving a receiver sensitivity of –85 dBm or better.

12.3.5 Receiver interference rejection

This subclause applies only to the 902–928 MHz band as there is only one channel available in the 868.0–868.6 MHz band.

The minimum receiver interference rejection levels are given in Table 84. The adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel, and the alternate channel is one more removed from the adjacent channel. For example, when channel 5 is the desired channel, channel 4 and channel 6 are the adjacent channels, and channel 3 and channel 7 are the alternate channels.

Table 84—Minimum receiver interference rejection requirements for 915 MHz ASK PHY

Adjacent channel rejection	Alternate channel rejection
0 dB	30 dB

The adjacent channel rejection shall be measured as follows: the desired signal shall be a compliant ASK PHY signal, as defined by 12.2, of pseudo-random data. The desired signal is input to the receiver at a level 3 dB greater than the maximum allowed receiver sensitivity given in 12.3.4.

In either the adjacent or the alternate channel, a compliant ASK PHY signal, as defined by 12.2, is input at the relative level specified in Table 84. The test shall be performed for only one interfering signal at a time. The receiver shall meet the error rate criteria defined in 8.1.7 under these conditions.

12.3.6 TX-to-RX turnaround time

The ASK PHY shall have a TX-to-RX turnaround time as defined in 8.2.1.

12.3.7 RX-to-TX turnaround time

The ASK PHY shall have an RX-to-TX turnaround time as defined in 8.2.2.

12.3.8 Error vector magnitude (EVM)

The ASK PHY shall have EVM values of less than 35% when measured for 1000 chips using the measurement process defined in 8.2.3.

12.3.9 Transmit center frequency tolerance

The ASK PHY transmit center frequency tolerance shall be ±40 ppm maximum.

12.3.10 Transmit power

The ASK PHY shall be capable of transmitting at a power level of least –3 dBm.

12.3.11 Receiver maximum input level of desired signal

The ASK PHY shall have a receiver maximum input level greater than or equal to –20 dBm using the measurement defined in 8.2.4.

12.3.12 Receiver ED

The ASK PHY shall provide the receiver ED measurement as described in 8.2.5.

12.3.13 Link quality indicator (LQI)

The ASK PHY shall provide the LQI measurement as described in 8.2.6.

12.3.14 Clear channel assessment (CCA)

The ASK PHY shall use the one of the CCA modes as described in 8.2.7.

12.3.15 Example of PSSS encoding

Table 85 shows the example 5-bit symbol that will be encoded using Table 82, the code table for 915 MHz.

Table 85—Example 5 bit symbol

Data bits	
Bit	**Bit value**
b_0	1
b_1	–1
b_2	1
b_3	–1
b_4	–1

By weighting sequence 0 with b_0, sequence 1 with b_1, …, sequence 4 with b_4, the weighted sequences in Table 86 are calculated. The PSSS symbol $p(m)$ is calculated by adding the columns of weighted sequences. For 868 MHz, the half-chips are added column-wise.

For the example $p(m)$ in Table 86, the Max$\{p(m) = 5$ and the Min$\{p(m) = –5$. The precoding of one symbol is executed independently of the precoding of any other symbol with the two steps described in 12.2.4. The resulting $p'(m)$ and $p''(m)$ are shown in Table 87.

A graph illustrating the example $p''(m)$ chip weights that will be applied to the pulse shaping filter in 12.2.6 is shown in Figure 75.

Table 86—Weighted sequences

Weighted sequence	Chip number																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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	1	-1	1	-1	1	-1
1	-1	-1	1	-1	1	-1	1	1	1	1	-1	1	1	-1	1	-1	-1	1	1	-1	-1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1
2	-1	-1	1	1	-1	1	1	1	-1	1	-1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1	-1	1	1	1	1	1	-1	-1
3	-1	-1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	-1	1	1	1	1	-1	1	1	-1	1	-1	-1	1	1	-1
4	-1	1	-1	-1	1	1	-1	-1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	-1	1	1	1	1	-1	1	1	-1
p(m)	-5	-3	-1	-3	1	1	1	3	-3	1	-1	1	-1	-1	1	-3	3	-1	-1	-1	-1	1	1	-3	3	3	5	-1	-1	3	1	-5

Table 87—*p*'(*m*), aligned symmetric to zero PSSS symbol, and *p*''(*m*), precoded PSSS symbol

p'(m)	-5	-3	-1	-3	1	1	1	3	-3	1	-1	1	-1	-1	1	-3	3	-1	-1	-1	-1	1	1	-3	3	3	5	-1	-1	3	1	-5
p''(m)	-1.0	-0.6	-0.2	-0.6	0.2	0.2	0.2	0.6	-0.6	0.2	-0.2	0.2	-0.2	-0.2	0.2	-0.6	0.6	-0.2	-0.2	-0.2	-0.2	0.2	0.2	-0.6	0.6	0.6	1.0	-0.2	-0.2	0.6	0.2	-1.0

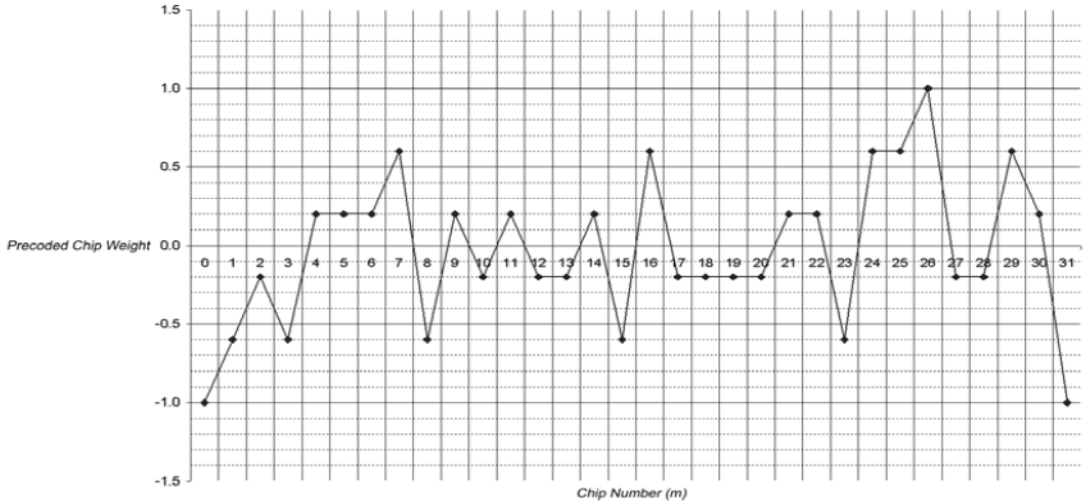


Figure 75—Precoded PSSS symbol *p*''(*m*)

13. Chirp spread spectrum (CSS) PHY

13.1 CSS PPDU format

The CSS PPDU shall be formatted as illustrated in Figure 76.

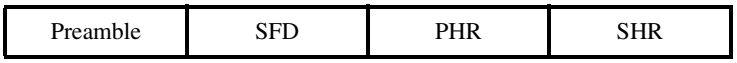


Figure 76—Format of the CSS PPDU

13.1.1 Preamble

The preamble for 1 Mb/s consists of 8 chirp symbols, and the preamble for optional 250 kb/s consists of 20 chirp symbols as specified in Table 88. The preamble sequence from Table 88 should be applied directly to both the I input and the Q input of QPSK.

Table 88—Preamble sequence

Data rate	Preamble sequence
1 Mb/s	ones (0:31)
250 kb/s	ones (0:79)

where

ones(0:*N*) for integer number *N* is defined an 1-by-*N* matrix of ones

13.1.2 SFD field

Start-of-frame delimiter (SFD) bit sequences for the CSS PHY type are defined in Table 89. Different SFD sequences are defined for the two different data rates. An SFD sequence from Table 89 shall be applied directly to both inputs (I and Q) of the QPSK mapper. An SFD sequence starts with bit 0.

Table 89—CSS SFD sequence

Data rate	Bit (0:15)
1 Mb/s	-1 1 1 1 -1 1 -1 -1 1 -1 -1 1 1 1 -1 -1
250 kb/s	-1 1 1 1 1 -1 1 -1 -1 -1 1 -1 -1 -1 1 1

13.1.3 PHY header (PHR)

The CSS PHR shall be formatted as illustrated in Figure 77.

bit 0 … bit 6	bit 7, bit 8	bit 9 … bit 11
Length of payload	Not used	Reserved

Figure 77—Format of the CSS PHR

13.2 Modulation and spreading

This PHY uses CSS techniques in combination with differential quadrature phase-shift keying (DQPSK) and 8-ary or 64-ary bi-orthogonal coding for 1 Mb/s data rate or 250 kb/s data rate, respectively. By using alternating time gaps in conjunction with sequences of chirp signals (subchirps) in different frequency subbands with different chirp directions, this CSS PHY provides subchirp sequence division as well as frequency division.

13.2.1 Data rates

The data rate of the CSS (2450 MHz) PHY shall be 1 Mb/s. An additional data rate of 250 kb/s shall be optional.

13.2.2 Reference modulator diagram

The functional block diagram in Figure 78 is provided as a reference for specifying the 2450 MHz CSS PHY modulation for both 1 Mb/s and optional 250 kb/s. All binary data contained in the PHR and PSDU shall be encoded using the modulation shown in Figure 78.

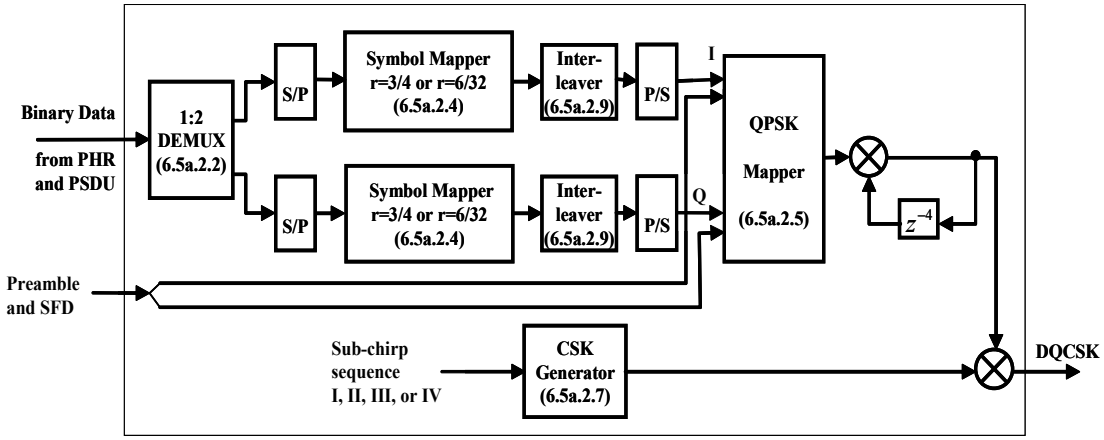


Figure 78—Differential bi-orthogonal quaternary-chirp-shift-keying modulator and spreading (r = 3/4 for 8-ary 1 Mb/s, r = 3/16 for 64-ary 250 kb/s)

13.2.3 De-multiplexer (DEMUX)

For each packet, the initial position of the DEMUX shown in Figure 78 shall be set to serve the I path (upper path). Thus, the first bit of the incoming stream of information bits of a packet shall be switched to the I path, and the second bit shall be switched to the Q path.

13.2.4 Serial-to-parallel mapping

By using two serial-to-parallel converters, the substreams are independently partitioned into sets of bits to form data symbols. For the mandatory data rate of 1 Mb/s, a data symbol shall consist of three bits. Within the binary data symbol (b0,b1,b2), the first input data bit for each of I and Q is assigned b0, and the third input data bit is assigned b2. For the optional data rate of 250 kb/s, a data symbol shall consist of 6 bits. Within the binary data symbol (b0,b1,b2,b3,b4,b5), the first input data bit for each of I and Q is assigned b0, and the sixth input data bit is assigned b5.

13.2.5 Data-symbol-to-bi-orthogonal-codeword mapping

Each 3-bit data symbol shall be mapped onto a 4-chip bi-orthogonal codeword (c0, c1, c2, c3) for the 1 Mb/s data rate as specified in Table 90. Each 6-bit data symbol shall be mapped onto a 32-chip bi-orthogonal codeword (c0, c1, c2, …, c31) for the optional 250 kb/s data rate as specified in Table 91.

Table 90—8-ary bi-orthogonal mapping (r = 3/4, 1 Mb/s)

Data symbol	Codeword (c_0 c_1 c_2 c_3)
0	1 1 1 1
1	1 -1 1 -1
2	1 1 -1 -1
3	1 -1 -1 1
4	-1 -1 -1 -1
5	-1 1 -1 1
6	-1 -1 1 1
7	-1 1 1 -1

Table 91—64-ary bi-orthogonal mapping (r = 3/16, 250 kb/s)

Data symbol	Codeword (c_0 c_1 c_2 … c_{31})
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 1 1 1 1 1 1
1	1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
2	1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1
3	1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1
4	1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1
5	1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1
6	1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1

Table 91—64-ary bi-orthogonal mapping (r = 3/16, 250 kb/s) *(continued)*

Data symbol	Codeword (c_0 c_1 c_2 … c_{31})
7	1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1
8	1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1
9	1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1
10	1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1
11	1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1
12	1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1
13	1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1
14	1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1
15	1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1
16	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
17	1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1
18	1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1
19	1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1
20	1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1
21	1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1
22	1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1
23	1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1
24	1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1
25	1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1
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 -1 -1 1 1 -1 -1
27	1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1

Table 91—64-ary bi-orthogonal mapping (r = 3/16, 250 kb/s) ***(continued)***

Data symbol	Codeword (c_0 c_1 c_2 … c_{31})
28	1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1
29	1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1
30	1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1
31	1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1
32	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
33	-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1
34	-1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1
35	-1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1
36	-1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1
37	-1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1
38	-1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1
39	-1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1
40	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1
41	-1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1
42	-1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1
43	-1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1
44	-1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1
45	-1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1
46	-1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1
47	-1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1
48	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Table 91—64-ary bi-orthogonal mapping (r = 3/16, 250 kb/s) *(continued)*

Data symbol	Codeword (c_0 c_1 c_2 … c_{31})
49	-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
50	-1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1
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
52	-1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1
53	-1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1
54	-1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1
55	-1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1
56	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1
57	-1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1
58	-1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1
59	-1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1
60	-1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1
61	-1 1 -1 1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1
62	-1 -1 1 1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1
63	-1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1

13.2.6 Parallel-to-serial converter and QPSK symbol mapping

Each bi-orthogonal codeword shall be converted to a serial chip sequence. Within each 4-chip codeword (c0, c1, c2, c3) for the 1 Mb/s data rate, the least significant chip c0 is processed first, and the most significant chip c3 is processed last for I and Q, respectively. Within each 32-chip codeword (c0, c1, c2, …, c31) for the 250 kb/s data rate, the least significant chip c0 is processed first, and the most significant chip c31 is processed last for I and Q, respectively. Each pair of I and Q chips shall be mapped onto a QPSK symbol as specified in Table 92.

13.2.7 DQPSK coding

The stream of QPSK symbols shall be differentially encoded by using a differential encoder with a QPSK symbol feedback memory of length 4. (In other words, the phase differences between QPSK symbol 1 and 5,

Table 92—QPSK symbol mapping

Input chips ($I_{n,k}$ $Q_{n,k}$)	Magnitude	Output phase (rad)
1, 1	1	0
-1, 1	1	$\pi/2$
1, -1	1	$-\pi/2$
-1, -1	1	π

2 and 6, 3 and 7, 4 and 8, and so on are computed.) For a detailed explanation of the index variables *n* and *k*, as described in 13.3.3.

DQPSK output:

$$e^{j\theta_{n,k}} = e^{j\theta_{n-1,k}} \times e^{j\varphi_{n,k}}$$

where

$e^{j\varphi_{n,k}}$ is DQPSK input

$e^{j\theta_{n-1,k}}$ is stored in feedback memory

For every packet, the initial values of all four feedback memory stages of the differential encoder shall be set:

$$e^{j\pi/4}$$

or equivalently:

$$\theta_{0,k} = \frac{\pi}{4}[\text{rad}]$$

13.2.8 DQPSK-to-DQCSK modulation

The stream of DQPSK symbols shall be modulated onto the stream of subchirps that is generated by the chirp-shift keying (CSK) generator. The effect of the differential quadrature chirp-shift keying (DQCSK) modulation shall be that each subchirp is multiplied with a DQPSK value that has unit magnitude and has constant phase for the duration of the subchirp. An example of this operation can be found in 13.3.6.

13.2.9 CSK generator

The CSK generator shall periodically generate one of the four defined subchirp sequences (chirp symbols) as specified in 13.3.3. Since each chirp symbol consists of four subchirps, the subchirp rate is four times higher than the chirp symbol rate.

13.2.10 Bit interleaver

The bit interleaver is applied only for the optional data rate of 250 kb/s. The 32 chip bi-orthogonal codewords for the optional 250 kb/s data rate are interleaved prior to the parallel to serial converter. Bit interleaving provides robustness against double intra-symbol errors caused by the differential detector. The interleaver permutes the chips across two consecutive codewords for each of I and Q, independently.

The memory of the interleaver shall be initialized with zeros before the reception of a packet.

The data stream going into the interleaver shall be padded with zeros if the number of octets to be transmitted does not align with the bounds of the interleaver blocks.

The input-output relationship of this interleaver shall be given as follows:

Input:

even-symbol (c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15,
c16, c17, c18, c19, c20, c21,c22,c23, c24, c25, c26, c27, c28, c29, c30, c31)
odd-symbol (d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15,
d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26, d27, d28, d29, d30, d31)

Output:

even-symbol (c0, c1, c2, c3, d20, d21, d22, d23, c8, c9, c10, c11, d28, d29, d30, d31,
c16, c17, c18, c19, d4, d5, d6, d7, c24, c25, c26, c27, d12, d13, d14, d15)
odd-symbol (d0, d1, d2, d3, c20, c21, c22, c23, d8, d9, d10, d11, c28, c29, c30, c31,
d16, d17, d18, d19, c4, c5, c6, c7, d24, d25, d26, d27, c12, c13, c14, c15)

NOTE—As shown in Figure 78, coding is applied to every bit following the SFD. The first codeword generated shall be counted as zero and thus is even.

13.3 Waveform and subchirp sequences

Four individual chirp signals, here called subchirps, shall be concatenated to form a full chirp symbol (subchirp sequence), which occupies two adjacent frequency subbands. Four different subchirp sequences are defined. Each subchirp is weighted with a raised cosine window in the time domain.

13.3.1 Graphical presentation of chirp symbols (subchirp sequences)

Four different sequences of subchirp signals are available for use. Figure 79 shows the four different chirp symbols (subchirp sequences) as time frequency diagrams. It can be seen that four subchirps, which have either a linear down-chirp characteristic or a linear up-chirp characteristic, and a center frequency, which has either a positive or a negative frequency offset, are concatenated. The frequency discontinuities between subsequent chirps will not affect the spectrum because the signal amplitude will be zero at these points.

13.3.2 Active usage of time gaps

In conjunction with the subchirp sequence, different pairs of time gaps are defined. The time gaps are chosen to make the four sequences even closer to being orthogonal. The time gaps shall be applied alternatively between subsequent chirp symbols as shown in Figure 80. The values of the time gaps are calculated from the timing parameters specified in Table 95.

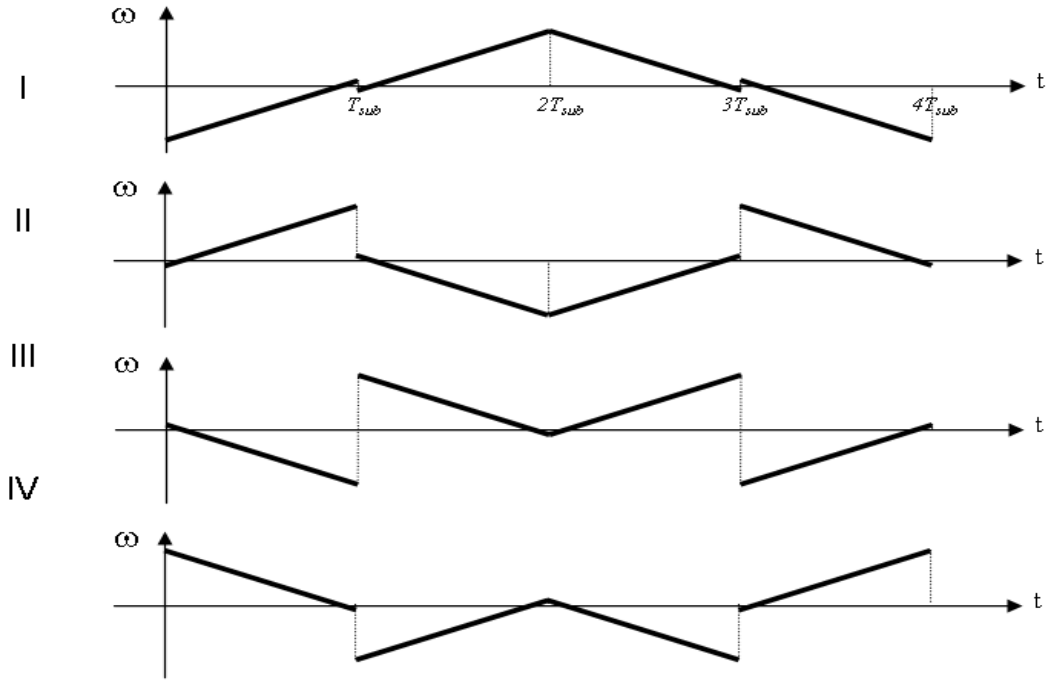


Figure 79—Four different combinations of subchirps

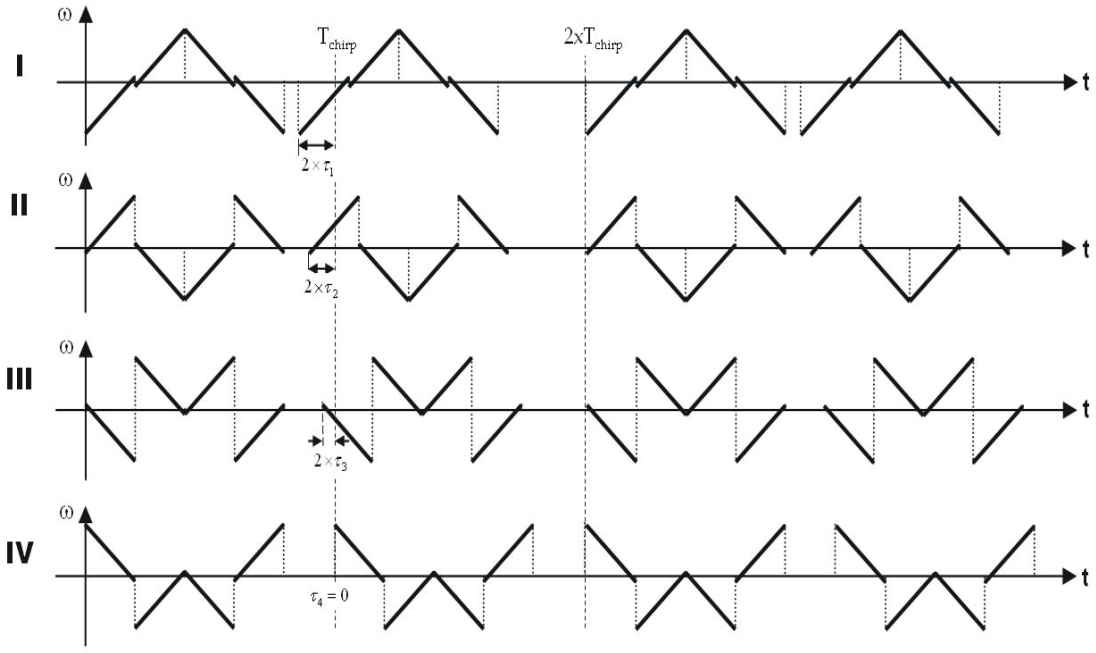


Figure 80—Four different time-gap pairs for the four different subchirp sequences

13.3.3 Mathematical representation of the continuous time CSS base-band signal

The mathematical representation of the continuous time-domain base-band signal $\tilde{s}^m(t)$ built of chirp symbols (subchirp sequences) as shown in Figure 79 with alternating time gaps as shown in Figure 80 is given by Equation (1). The subchirp sequence with its associated time gap is defined to be a chirp symbol.

$$\begin{aligned}\tilde{s}^m(t) &= \sum_{n=0}^{\infty} \tilde{s}^m(t, n) \\ &= \sum_{n=0}^{\infty} \sum_{k=1}^{4} \tilde{c}_{n,k} \exp\left[j\left(\hat{\omega}_{k,m} + \frac{\mu}{2} \xi_{k,m} (t - T_{n,k,m}) \right) (t - T_{n,k,m}) \right] \times P_{RC}(t - T_{n,k,m})\end{aligned} \tag{1}$$

where m = 1, 2, 3, 4 (I, II, III, and IV in Figure 79) defines which of the four different possible chirp symbols (subchirp sequences) is used,and n = 0, 1, 2 …, is the sequence number of the chirp symbols.

The $\tilde{c}_{n,k}$ of $s(t)$ in Equation (1) is the sequence of the complex data that consists of in-phase data $a_{n,k}$ and quadrature-phase data $b_{n,k}$ as the output of DQPSK coding.

The possible value of $a_{n,k}$ and $b_{n,k}$ are +1 or –1.

$$\tilde{c}_{n,k} = a_{n,k} + jb_{n,k}$$

where

n	is the sequence number of chirp symbols
k = 0,1,2, and 3	is the subchirp index
j	is $\sqrt{-1}$

$\hat{\omega}_{k,m} = 2\pi \times f_{k,n}$ are the center frequencies of the subchirp signals. This value depends on m and k = 1, 2, 3, 4, which defines the subchirp number in the subchirp sequence.

$T_{n,k,m}$ defines the starting time of the actual subchirp signal to be generated. It is determined by T_{chirp}, which is the average duration of a chirp symbol, and by T_{sub}, which is the duration of a subchirp signal.

$$T_{n,k,m} = \left(k + \frac{1}{2}\right) T_{sub} + nT_{chirp} - (1 - (-1)^n)\tau_m$$

The constant μ defines the characteristics of the subchirp signal. A value of $\mu = 2\pi \times 7.3158 \times 10^{12}$ [rad/s^2] shall be used.

The function P_{RC}, which is defined in 13.3.4, is a windowing function that is equal to zero at the edges and outside of the subchirp centered at time zero.

The constant τ_m is either not added or added twice and thus determines (but is not identical to) the time gap that was applied between two subsequent subchirp sequences as shown in Figure 80.

Table 93 shows the values for the subband center frequencies, Table 94 the subchirp directions, and Table 95 the timing parameters in Equation (1). It should be noted that these time and frequency parameters are assumed to be derived from a reference crystal in a locked manner. In other words, any relative errors in chirp subband center frequencies, chirp rate, and time gaps are equal.

Table 93—Subband center frequencies, $f_{k,m}$ (MHz, numerical parameters)

*m**k*	1	2	3	4
1	$f_c - 3.15$	$f_c + 3.15$	$f_c + 3.15$	$f_c - 3.15$
2	$f_c + 3.15$	$f_c - 3.15$	$f_c - 3.15$	$f_c + 3.15$
3	$f_c - 3.15$	$f_c + 3.15$	$f_c + 3.15$	$f_c - 3.15$
4	$f_c + 3.15$	$f_c - 3.15$	$f_c - 3.15$	$f_c + 3.15$

Table 94—Subchirp directions, $\zeta_{k,m}$, numerical parameters

*m**k*	1	2	3	4
1	+1	+1	–1	–1
2	+1	–1	+1	–1
3	–1	–1	+1	+1
4	–1	+1	–1	+1

Table 95—Timing parameters for baseband signal

Symbol	Value	Multiple of 1/32MHz
T_{chirp}	6 μs	192
T_{sub}	1.1875 μs	38
τ_1	468.75 μs	15
τ_2	312.5 ns	10
τ_4	156.25 ns	5
τ_4	0 ns	0

13.3.4 Raised cosine window for chirp pulse shaping

The raised-cosine time-window described here shall be used to shape the subchirp. The raised cosine window $P_{RC}(t)$ is applied to every subchirp signal in the time domain, as illustrated in Figure 81.

$$P_{RC}(t) = \begin{cases} 1 & |t| \le \frac{(1-\alpha)}{(1+\alpha)}\frac{T_{sub}}{2} \\ \frac{1}{2}\left[1 + \cos\left(\frac{(1+\alpha)\pi}{\alpha T_{sub}}\left(|t| - \frac{(1-\alpha)}{(1+\alpha)}\frac{T_{sub}}{2}\right)\right)\right] & \frac{(1-\alpha)}{(1+\alpha)}\frac{T_{sub}}{2} < |t| \le \frac{T_{sub}}{2} \\ 0 & |t| > \frac{T_{sub}}{2} \end{cases}$$

where

$\alpha = 0.25$

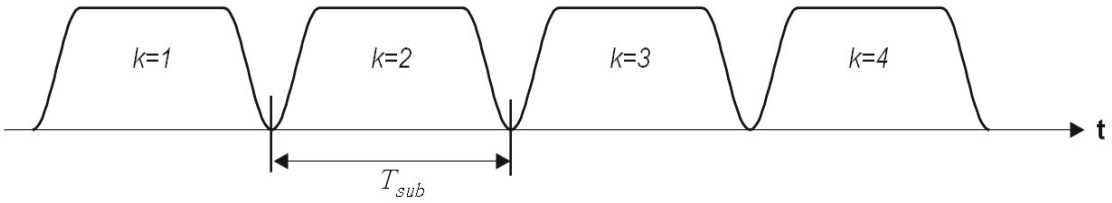


Figure 81—Subchirp time-domain pulse shaping

13.3.5 Subchirp transmission order

During each chirp symbol period, subchirp 1 ($k = 1$) is transmitted first, and subchirp 4 ($k = 4$) is transmitted last.

13.3.6 Example of CSK signal generation

An example for the modulation of one chirp symbol is provided in this subclause to illustrate each step from DEMUX to the output of the reference modulator as shown in Figure 78. The scenario parameters are as follows:

— The initial values of all four feedback memory stages of the differential encoder is set $e^{(j\pi)/4}$.

— The data bit rate is 1 Mb/s.

Input binary data:

0 1 0 1 1 0

Demux:

I-path: 0 0 1

Q-path: 1 1 0

Serial-to-parallel mapping:

I-path: {1 0 0}

Q-path: {1 1 0}

Bi-orthogonal mapping ($r = 3/4$):

I-path: 1 –1 1 –1

Q-path: –1 –1 1 1

Parallel-to-serial and QPSK symbol mapping:

Mapper input: $(1 - j)$, $(-1 - j)$, $(1 + j)$, $(-1 + j)$

QPSK output phase: $-\pi/2$, π, 0, $\pi/2$

D-QPSK coding:

Initial phase of four feedback memory for D-QPSK: all π/4

D-QPSK coder output phase: –π/4, –3π/4, π/4, 3π/4

D-QPSK–to–D-QCSK modulation output and subchirp sequence of D-QCSK output:

[exp(–jπ/4) × subchirp(k = 1), exp(–j3π/4) × subchirp(k = 2), exp(jπ/4) × subchirp(k = 3), exp(j3π/4) × subchirp(k = 4)]

13.4 CSS RF requirements

In addition to meeting regional regulatory requirements, CSS devices operating in the 2450 MHz band shall also meet the requirements in this subclause.

13.4.1 Transmit power spectral density (PSD) mask and signal tolerance

The transmitted spectral power density of a CSS signal $s(t)$ shall be within the relative limits specified in the template shown in Figure 82. The average spectral power shall be made using 100 kHz resolution bandwidth and a 1 kHz video bandwidth. For the relative limit, the reference level shall be the highest average spectral power measured within ±11 MHz of the carrier frequency. Specifically, the normalized frequency spectrum to the peak value in the signal bandwidth $|f - f_c| \leq 7$ MHz shall be less than or equal to –30 dB in the stop band 11 MHz $\leq |f - f_c| \leq 22$ MHz and shall be less than or equal to –50 dB in the stop band $|f - f_c| > 22$ MHz. For testing the transmitted spectral power density, a $2^{15} - 1$ pseudo-random binary sequence (PRBS) shall be used as input data.

As additional criteria for the compliance of a CSS signal, the mean square error shall be used. Let $s(t)$ be the baseband CSS signal that is given in Equation (1). Then the implemented signal, $s_{impl}(t)$, shall satisfy the following equation:

$$\text{mmse} = \min_{A,\, \tau_d,\, \varphi} \left(\frac{\int_0^T \left| s^m(t) - A \times s^m_{impl}(t - \tau_d) e^{j\varphi} \right|^2 dt}{\int_0^T \left| s^m(t) \right|^2 dt} \right) \leq 0.005$$

where the constants A, τ_d, and φ ϕ are used to minimize the mean squared error. The constant T_{chirp} is the period of the CSS symbol. The $c_{n,k}$ of $s(t)$ in Equation (1) is the constant data $(1 + j)$ for the measurement for all n and k.

13.4.2 Symbol rate

The 2450 MHz PHY DQCSK symbol rate shall be 166.667 ksymbol/s (1/6 Msymbol/s) ± 40 ppm.

13.4.3 Receiver sensitivity

Under the conditions specified in 6.1.6, a compliant device shall be capable of achieving a receiver sensitivity of –85 dBm or better for 1 Mb/s and –91 dBm or better for 250 kb/s.

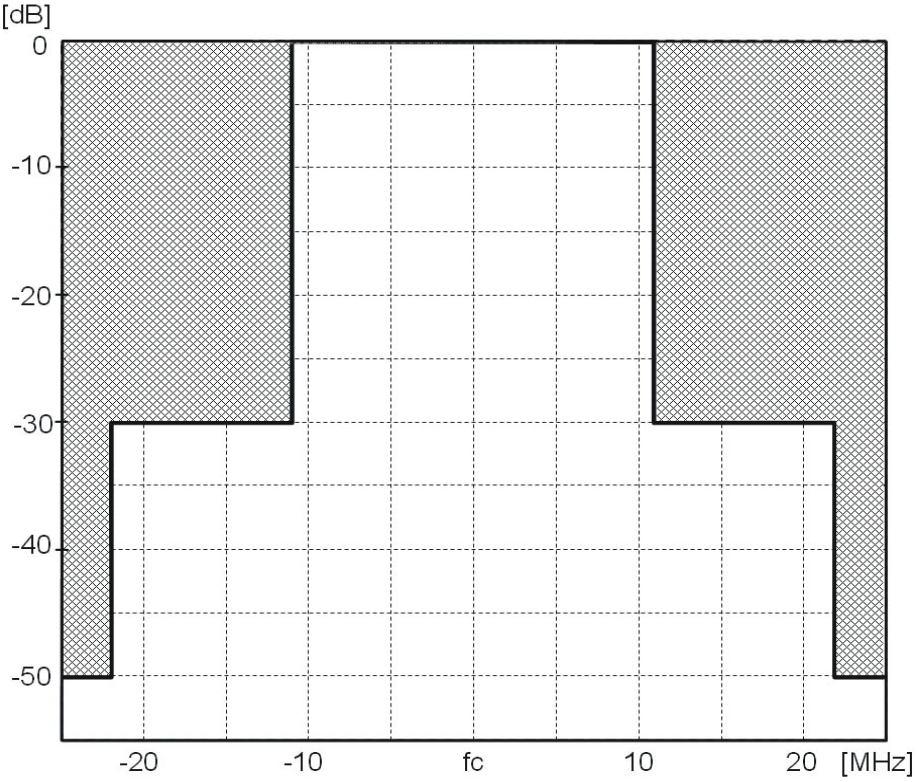


Figure 82—Transmit PSD mask

13.4.4 Receiver interference rejection

Table 96 gives minimum receiver interference rejection levels. A nonoverlapping adjacent channel is defined to have a center frequency offset of 25 MHz. A nonoverlapping alternate channel is defined to have a center frequency offset of 50 MHz. The adjacent channel rejection shall be measured as follows: The desired signal shall be a compliant 2450 MHz CSS signal of pseudo-random data. The desired signal is input to the receiver at a level 3 dB above the maximum allowed receiver sensitivity given in 13.4.3. In the adjacent or the alternate channel, a CSS signal of the same or a different subchirp sequence as the victim device is input at the relative level specified in Table 96. The test shall be performed for only one interfering signal at a time. The receiver shall meet the error rate criteria defined in 6.1.6 under these conditions.

Table 96—Minimum receiver interference rejection levels for 2450 MHz CSS PHY

Data rate	Nonoverlapping adjacent channel rejection (25 MHz offset) (dB)	Nonoverlapping alternate channel rejection (50 MHz offset) (dB)
1 Mb/s	34	48
250 kb/s (optional)	38	52

13.4.5 TX-to-RX turnaround time

The TX-to-RX turnaround time shall be less than or equal to *aTurnaroundTime*, as defined in 9.2.

The TX-to-RX turnaround time is defined as the shortest time possible at the air interface from the trailing edge of the last chirp (of the last symbol) of a transmitted PPDU to the leading edge of the first chirp (of the first symbol) of the next received PPDU.

The TX-to-RX turnaround time shall be less than or equal to the RX-to-TX turnaround time.

13.4.6 RX-to-TX turnaround time

The RX-to-TX turnaround time shall be less than or equal to *aTurnaroundTime*, as defined in 9.2.

The RX-to-TX turnaround time is defined as the shortest time possible at the air interface from the trailing edge of the last chirp (of the last symbol) of a received PPDU to the leading edge of the first chirp (of the first symbol) of the next transmitted PPDU.

13.4.7 Transmit center frequency tolerance

The CSS PHY transmit center frequency tolerance shall be ±40 ppm maximum.

13.4.8 Transmit power

The CSS PHY shall be capable of transmitting at a power level of least –3 dBm.

13.4.9 Receiver maximum input level of desired signal

The CSS PHY shall have a receiver maximum input level greater than or equal to –20 dBm using the measurement defined in 8.2.4.

13.4.10 Receiver ED

The CSS PHY shall provide the receiver ED measurement as described in 8.2.5.

13.4.11 Link quality indicator (LQI)

The CSS PHY shall provide the LQI measurement as described in 8.2.6.

13.4.12 Clear channel assessment (CCA)

The CSS PHY– shall use the one of the CCA modes as described in 8.2.7.

14. UWB PHY

14.1 General

The UWB PHY waveform is based upon an impulse radio signaling scheme using band-limited data pulses. The UWB PHY supports three independent bands of operation:

— The sub-gigahertz band, which consists of a single channel and occupies the spectrum from 249.6 MHz to 749.6 MHz
— The low band, which consists of four channels and occupies the spectrum from 3.1 GHz to 4.8 GHz
— The high band, which consists of 11 channels and occupies the spectrum from 6.0 GHz to 10.6 GHz

Within each channel, there is support for at least two complex channels that have unique length 31 SHR preamble codes. The combination of a channel and a preamble code is termed a *complex channel*. A compliant device shall implement support for at least one of the channels (0, 3, or 9) in Table 107. In addition, each device shall support the two unique length 31 preamble codes for the implemented channels as defined in Table 102. Support for the other channels listed in Table 107 is optional.

A combination of burst position modulation (BPM) and binary phase-shift keying (BPSK) is used to support both coherent and noncoherent receivers using a common signaling scheme. The combined BPM-BPSK is used to modulate the symbols, with each symbol being composed of an active burst of UWB pulses. The various data rates are supported through the use of variable-length bursts. Figure 83 shows the sequence of processing steps used to create and modulate a UWB PPDU. The sequence of steps indicated here for the transmitter is used as a basis for explaining the creation of the UWB PHY waveform specified in the PHY of this standard. Note that the receiver portion of Figure 83 is informative and meant only as a guide to the essential steps that any compliant UWB receiver needs to implement in order to successfully decode the transmitted signal.

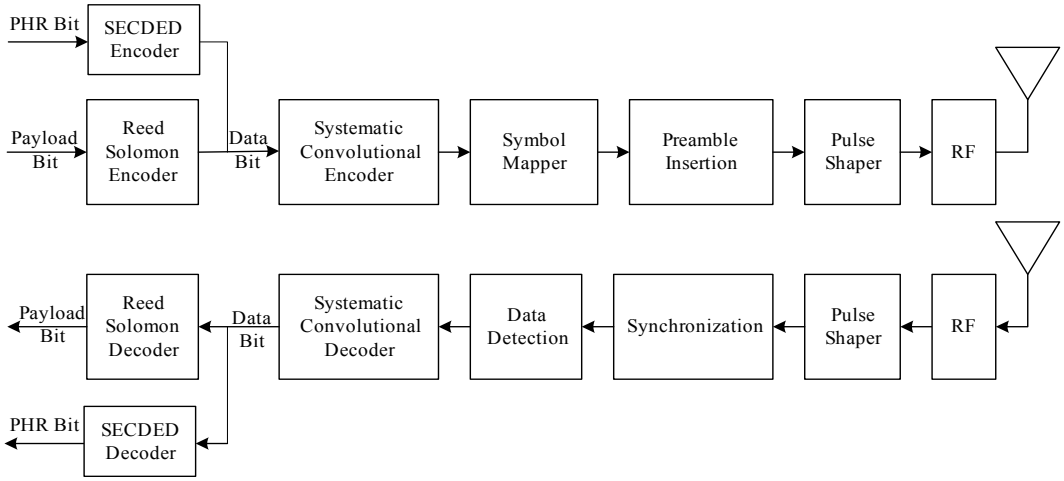


Figure 83—UWB PHY signal flow

14.2 UWB PPDU format

Figure 84 shows the format for the UWB PPDU. For convenience, the PPDU structure is presented so that the leftmost field as written in this standard shall be transmitted or received first. Unless otherwise specified, all multiple octet fields shall be transmitted or received least significant octet first, and each octet shall be transmitted or received LSB first. Unless otherwise specified, all multi-bit strings shall be transmitted LSB first.

Note that each UWB-compliant device shall support the length 31 preamble codes specified in Table 102 and that two base rates corresponding to the two mandatory PRFs result for this code length. The mandatory SHR preamble base rates are, therefore, 1.01 Msymbol/s and 0.25 Msymbol/s as indicated in Table 100.

The PHR is sent at 850 kb/s for all data rates greater than or equal to 850 kb/s and at 110 kb/s for the data rate of 110 kb/s. The PSDU is sent at the desired information data rate as defined in Table 99.

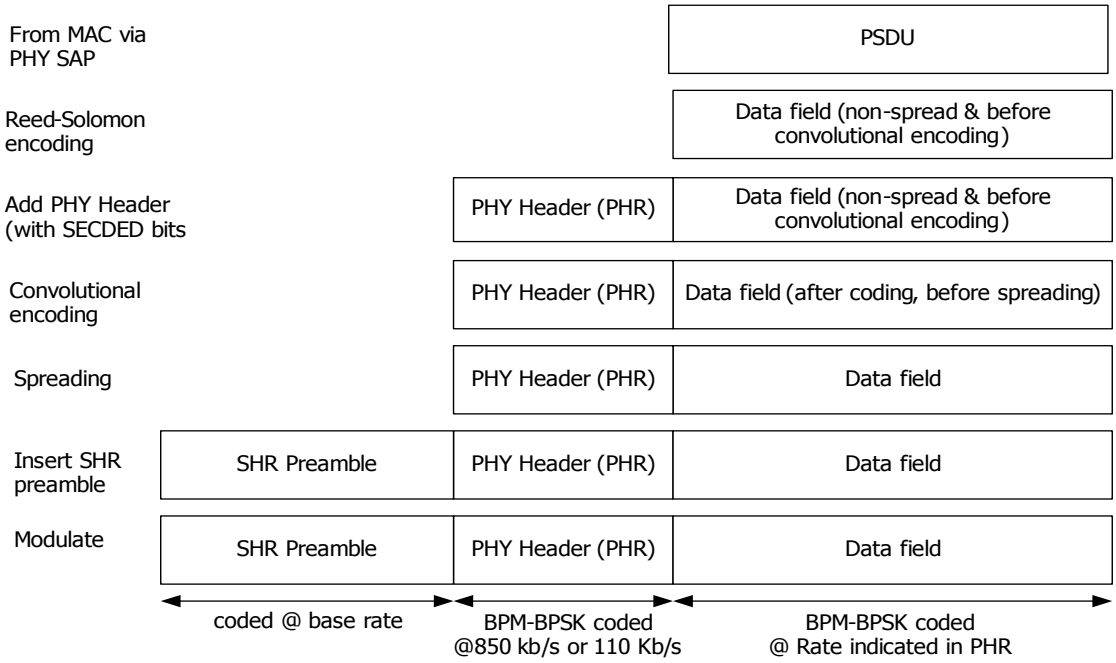


Figure 84—UWB PPDU encoding process

14.2.1 PPDU encoding process

The encoding process is composed of many steps, as illustrated in Figure 84. The details of these steps are fully described in later subclauses, as noted in the following list:

a) Perform Reed-Solomon encoding on PSDU as described in 14.3.3.1.

b) Produce the PHR as described in 14.2.6.1.

c) Add SECDED check bits to PHR as described in 14.2.6.2 and prepend to the PSDU.

d) Perform further convolutional coding as described in 14.3.3.2. Note that in some instances at the 27 Mb/s data rate, the convolutional encoding of the data field is effectively bypassed and two data bits are encoded per BPM-BPSK symbol.

e) Modulate and spread PSDU according to the method described in 14.3.1 and 14.3.2 The PHR is modulated using BPM-BPSK at either 850 kb/s or 110 kb/s and the data field is modulated at the rate specified in the PHR.

f) Produce the SHR preamble field from the SYNC field (used for AGC convergence, diversity selection, timing acquisition, and coarse frequency acquisition) and the SFD field (used to indicate the start of frame). The SYNC and SFD fields are described in 14.2.5.1 and 14.2.5.2, respectively.

Table 97 and Table 98 show how the 19 header bits (H_0–H_{18}), N data bits (D_0–D_{N-1}), and two tail bits (T_0–T_1) are mapped onto the symbols. In these tables, the polarity bit column operation is an XOR. The tables also show when the transition from the header bit rate to the data bit rate takes place. Note that the delay line of the convolutional code is initialized to zero. For this reason, the position bit of Symbol 0 shall always be zero. This means that Symbol 0 is always transmitted in the first half of the first header symbol.

Table 97—Mapping of header bits, data bits, and tail bits onto symbols with Viterbi rate 0.5

Symbol #	Input data	Position bit	Polarity bit	
0	H_0	0	H_0	21 symbols of header at 850 kb/s
1	H_1	H_0	H_1	
2	H_2	H_1	$H_0 \oplus H_2$	
3	H_3	H_2	$H_1 \oplus H_3$	
…	…	…	…	
16	H_{16}	H_{15}	$H_{14} \oplus H_{16}$	
17	H_{17}	H_{16}	$H_{15} \oplus H_{17}$	
18	H_{18}	H_{17}	$H_{16} \oplus H_{17}$	
19	D_0	H_{18}	$H_{17} \oplus D_0$	
20	D_1	D_0	$H_{18} \oplus D_1$	
21	D_2	D_1	$D_0 \oplus D_2$	N symbols of data at data rate, e.g., 6.8 Mb/s
…	…	…	…	
N+17	D_{N-2}	D_{N-3}	$D_{N-4} \oplus D_{N-2}$	
N+18	D_{N-1}	D_{N-2}	$D_{N-3} \oplus D_{N-1}$	
N+19	T_0	D_{N-1}	$D_{N-2} \oplus T_0$	
N+20	T_1	T_0	$D_{N-1} \oplus T_1$	

14.2.2 UWB PHY symbol structure

In the BPM-BPSK modulation scheme, a UWB PHY symbol is capable of carrying two bits of information: one bit is used to determine the position of a burst of pulses, while an additional bit is used to modulate the phase (polarity) of this same burst.

The structure and timing of a UWB PHY symbol is illustrated in Figure 85. Each symbol shall consist of an integer number of possible chip positions, N_c, each with duration T_c. The overall symbol duration denoted by T_{dsym} is given by $T_{dsym} = N_c T_c$. Furthermore, each symbol is divided into two BPM intervals each with duration $T_{BPM} = T_{dsym}/2$, which enables binary position modulation.

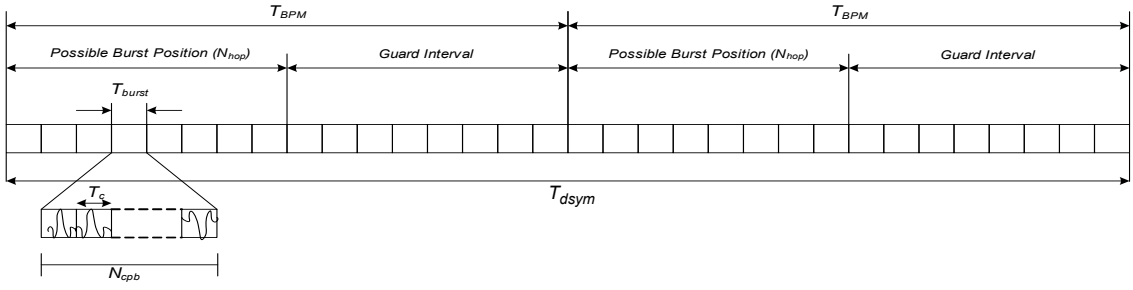


Figure 85—UWB PHY symbol structure

Table 98—Mapping of header bits, data bits and tail bits onto symbols with Viterbi rate 1

Symbol #	Input data	Position bit	Polarity bit	
0	H_0	0	H_0	21 symbols of header at 850 kb/s
1	H_1	H_0	H_1	
2	H_2	H_1	$H_0 \oplus H_2$	
3	H_3	H_2	$H_1 \oplus H_3$	
…	…	…	…	
16	H_{16}	H_{15}	$H_{14} \oplus H_{16}$	
17	H_{17}	H_{16}	$H_{15} \oplus H_{17}$	
18	H_{18}	H_{17}	$H_{16} \oplus H_{17}$	
19	T_0	H_{18}	$H_{17} \oplus T_0$	
20	T_1	T_0	$H_{18} \oplus T_1$	
21	D_0, D_1	D_0	D_1	1/2 N symbols of data at data rate, e.g., 6.8 Mb/s
…	D_2, D_3	D_2	D_3	
…	…	…	…	
1/2 N+19	D_{N-6}, D_{N-5}	D_{N-6}	D_{N-5}	
1/2 N+20	D_{N-4}, D_{N-3}	D_{N-4}	D_{N-3}	
1/2 N+21	D_{N-2}, D_{N-1}	D_{N-2}	D_{N-1}	

A burst is formed by grouping N_{cpb} consecutive chips and has duration $T_{burst} = N_{cpb}T_c$. The location of the burst in either the first half or the second half of the symbol indicates one bit of information. Additionally, the phase of the burst (either –1 or +1) is used to indicate a second bit of information.

In each UWB PHY symbol interval, a single burst event shall be transmitted. The fact that burst duration is typically much shorter than the BPM duration, i.e., $T_{burst} << T_{BPM}$, provides for some multi-user access interference rejection in the form of time hopping. The total number of burst durations per symbol, N_{burst}, is given by $N_{burst} = T_{dsym}/T_{burst}$. In order to limit the amount of inter-symbol interference caused by multipath, only the first half of each T_{BPM} period shall contain a burst. Therefore, only the first $N_{hop} = N_{burst}/4$ possible burst positions are candidate hopping burst positions within each BPM interval. Each burst position can be varied on a symbol-to-symbol basis according to a time hopping code as described in 14.3.

14.2.3 PSDU timing parameters

The PSDU rate-dependent parameters and timing-related parameters are summarized in Table 99. Within each UWB channel {0:15}, the peak PRF shall be 499.2 MHz. This rate corresponds to the highest frequency at which a compliant transmitter shall emit pulses. Additionally, the mean PRF is defined as the total number of pulses emitted during a symbol period divided by the length of the symbol duration. During the SHR preamble portion of a UWB frame, the peak and mean PRFs are essentially the same since pulses are emitted uniformly during each preamble symbol. During the data portion of a PPDU, however, the peak and mean PRFs differ due to the grouping of pulses into consecutive chip durations.

There are two possible preamble code lengths (31 or 127) and three possible mean PRFs (15.6 MHz, 3.90 MHz, and 62.4 MHz). A compliant device shall implement support for the preamble code length of 31

Table 99—UWB PHY rate-dependent and timing-related parameters

Channel Number	Peak PRF MHz	Bandwidth MHz	Preamble Code Length	Modulation & Coding			Data Symbol Structure						Data		
				Viterbi Rate	RS Rate	Overall FEC Rate	#Burst Positions per Symbol N_{burst}	# Hop Bursts N_{hop}	# Chips Per Burst N_{cpb}	#Chips Per Symbol	Burst Duration T_{burst} (ns)	Symbol Duration T_{dsym} (ns)	Symbol Rate (MHz)	Bit Rate Mb/s	Mean PRF (MHz)
{0:3, 5:6, 8:10, 12:14}	499.2	499.2	31	0.5	0.87	0.44	32	8	128	4096	256.41	8205.13	0.12	0.11	15.60
	499.2	499.2	31	0.5	0.87	0.44	32	8	16	512	32.05	1025.64	0.98	0.85	15.60
	499.2	499.2	31	0.5	0.87	0.44	32	8	2	64	4.01	128.21	7.80	6.81	15.60
	499.2	499.2	31	1	0.87	0.87	32	8	1	32	2.00	64.10	15.60	27.24	15.60
{0:3, 5:6, 8:10, 12:14}	499.2	499.2	31	0.5	0.87	0.44	128	32	32	4096	64.10	8205.13	0.12	0.11	3.90
	499.2	499.2	31	0.5	0.87	0.44	128	32	4	512	8.01	1025.64	0.98	0.85	3.90
	499.2	499.2	31	0.5	0.87	0.44	128	32	2	256	4.01	512.82	1.95	1.70	3.90
	499.2	499.2	31	1	0.87	0.87	128	32	1	128	2.00	256.41	3.90	6.81	3.90
{0:3, 5:6, 8:10, 12:14}	499.2	499.2	127	0.5	0.87	0.44	8	2	512	4096	1025.64	8205.13	0.12	0.11	62.40
	499.2	499.2	127	0.5	0.87	0.44	8	2	64	512	128.21	1025.64	0.98	0.85	62.40
	499.2	499.2	127	0.5	0.87	0.44	8	2	8	64	16.03	128.21	7.80	6.81	62.40
	499.2	499.2	127	0.5	0.87	0.44	8	2	2	16	4.01	32.05	31.20	27.24	62.40
{4, 11}	499.2	1331.2	31	0.5	0.87	0.44	32	8	128	4096	256.41	8205.13	0.12	0.11	15.60
	499.2	1331.2	31	0.5	0.87	0.44	32	8	16	512	32.05	1025.64	0.98	0.85	15.60
	499.2	1331.2	31	0.5	0.87	0.44	32	8	2	64	4.01	128.21	7.80	6.81	15.60
	499.2	1331.2	31	1	0.87	0.87	32	8	1	32	2.00	64.10	15.60	27.24	15.60
{4, 11}	499.2	1331.2	127	0.5	0.87	0.44	8	2	512	4096	1025.64	8205.13	0.12	0.11	62.40
	499.2	1331.2	127	0.5	0.87	0.44	8	2	64	512	128.21	1025.64	0.98	0.85	62.40
	499.2	1331.2	127	0.5	0.87	0.44	8	2	8	64	16.03	128.21	7.80	6.81	62.40
	499.2	1331.2	127	0.5	0.87	0.44	8	2	2	16	4.01	32.05	31.20	27.24	62.40
7	499.2	1081.6	31	0.5	0.87	0.44	32	8	128	4096	256.41	8205.13	0.12	0.11	15.60
	499.2	1081.6	31	0.5	0.87	0.44	32	8	16	512	32.05	1025.64	0.98	0.85	15.60
	499.2	1081.6	31	0.5	0.87	0.44	32	8	2	64	4.01	128.21	7.80	6.81	15.60
	499.2	1081.6	31	1	0.87	0.87	32	8	1	32	2.00	64.10	15.60	27.24	15.60
7	499.2	1081.6	127	0.5	0.87	0.44	8	2	512	4096	1025.64	8205.13	0.12	0.11	62.40
	499.2	1081.6	127	0.5	0.87	0.44	8	2	64	512	128.21	1025.64	0.98	0.85	62.40
	499.2	1081.6	127	0.5	0.87	0.44	8	2	8	64	16.03	128.21	7.80	6.81	62.40
	499.2	1081.6	127	0.5	0.87	0.44	8	2	2	16	4.01	32.05	31.20	27.24	62.40
15	499.2	1354.97	31	0.5	0.87	0.44	32	8	128	4096	256.41	8205.13	0.12	0.11	15.60
	499.2	1354.97	31	0.5	0.87	0.44	32	8	16	512	32.05	1025.64	0.98	0.85	15.60
	499.2	1354.97	31	0.5	0.87	0.44	32	8	2	64	4.01	128.21	7.80	6.81	15.60
	499.2	1354.97	31	1	0.87	0.87	32	8	1	32	2.00	64.10	15.60	27.24	15.60
15	499.2	1354.97	127	0.5	0.87	0.44	8	2	512	4096	1025.64	8205.13	0.12	0.11	62.40
	499.2	1354.97	127	0.5	0.87	0.44	8	2	64	512	128.21	1025.64	0.98	0.85	62.40
	499.2	1354.97	127	0.5	0.87	0.44	8	2	8	64	16.03	128.21	7.80	6.81	62.40
	499.2	1354.97	127	0.5	0.87	0.44	8	2	2	16	4.01	32.05	31.20	27.24	62.40

and shall also support both the 15.6 MHz and 3.90 MHz mean PRFs for the PSDU as depicted in Table 99. The use of the length 127 code is optional; when implemented, the mean PRF of the PSDU shall be 62.4 MHz.

UWB channels {4, 7, 11, 15} are all optional channels and are differentiated from other UWB channels by the larger bandwidth (>500 MHz) of the transmitted signals. These channels overlap the existing lower bandwidth channels. The larger bandwidth enables devices operating in these channels to transmit at a higher power (for fixed PSD constraints), and thus they may achieve a longer communication range. The larger bandwidth pulses offer enhanced multipath resistance. Additionally, larger bandwidth leads to more accurate range estimates. The admissible data rates, preamble code lengths, PRFs, and modulation timing parameters are listed in Table 99.

Each UWB channel allows for several data rates that are obtained by modifying the number of chips within a burst, while the total number of possible burst positions remains constant. Therefore, the symbol duration, T_{dsym}, changes to obtain the stated symbol rate and bit rates.

Each row in Table 99 completely describes all timing parameters shown in Figure 85 for each permitted combination of channel number, preamble code length, and PRF.

The channel number parameter column identifies the UWB PHY channel numbers where the remaining PSDU timing parameters in the current row are valid. Association between channel number and center frequency is given in Table 107.

The peak PRF states the highest frequency in megahertz at which a compliant transmitter shall emit pulses. The peak PRF is also used to derive the chip duration T_c by the formula $T_c = 1/(peakPRF)$. The value of T_c is approximately 2 ns.

The bandwidth denotes the 3 dB bandwidth of the UWB pulses. Note that the bandwidth is not necessarily the inverse of the chip duration T_c. Pulse shape and bandwidth are further defined in 14.4.5.

The preamble code length parameter denotes the length of the preamble code length to be used during the SHR portion of a data frame. The code length together with the channel number defines a complex channel. Individual codes to be used on each channel are given in Table 102 (length 31) and Table 103 (length 127).

This Viterbi rate parameter determines the rate of the convolutional code applied to the PSDU data bits. A value of 1 indicates that no convolutional coding is applied, while a value of 0.5 indicates that a rate 1/2 code as described in 14.3.3.2 is applied to the PSDU data bits.

The RS rate parameters indicates the (63,55) Reed-Solomon code rate, which is approximately 0.87. The Reed-Solomon code is applied to all the PSDU data bits that are transmitted by the UWB PHY. Reed-Solomon encoding is further described in 14.3.3.1.

The overall FEC rate is determined by the product of the Viterbi rate and the Reed-Solomon rate and has either a value of 0.44 or 0.87.

The burst positions per symbol parameter is the total number of possible burst positions in a data symbol duration. N_{burst} has been chosen so that for each mean PRF a data symbol consists of a fixed number of burst durations.

The hop bursts parameter is the number of burst positions that may contain an active burst, that is, a burst containing UWB pulses. The value is computed as $N_{hop} = N_{burst}/4$.

The chips per burst parameter is the number of chip T_c durations within each burst period T_{burst}. Each burst consists of a multiple number of consecutive chips, as illustrated in Figure 85. Depending on the data rate to be used in the transmission of the PSDU, the number of chips in a burst varies, e.g., for low data rates, the burst consists of more chip periods than for high data rates. Particular, values of N_{cpb} have been selected so that the following is a valid data rate: (2 × Overall FEC Rate)/($N_{cpb} \times N_{burst} \times T_c$).

The burst duration parameter is simply the duration of a burst and is computed as $T_{burst} = N_{cpb} \times T_c$.

The symbol duration parameter is a the duration of a modulated and coded PSDU symbol on the air and is computed as follows: $T_{dsym} = N_{burst} \times T_{burst}$.

The symbol rate parameter is the inverse of the PSDU symbol duration $1/T_{dsym}$.

The bit rate parameter is the user information rate considering FEC and is computed as follows:

Bit Rate = 2 × (Overall FEC Rate)/T_{dsym}

The mean PRF parameter is the average PRF during the PSDU portion of a PHY frame and is computed as follows:

Mean PRF = N_{cpb}/T_{dsym}

14.2.4 Preamble timing parameters

Due to the variability in the preamble code length and the PRF, there are several admissible values for the timing parameters of a preamble symbol. These values are summarized in Table 100. In this subclause, a preamble symbol is defined as the waveform consisting of one whole repetition of the modulated preamble code (either length 31 or 127). Details on the construction of the preamble symbol for various code lengths and PRFs are given in 14.2.5. For each target PRF, the preamble is constructed from a preamble code, C_i, by inserting a number of chip durations between code symbols. The number of chip durations to insert is denoted by δ_L, values for each code length and PRF are given in Table 100, and the chip insertion is detailed in 14.2.5.1.

Table 100 presents the timing parameters during the SHR portion of a UWB PHY frame, while Table 99 presents the timing parameters for the PSDU portion of the frame. First, note that the preamble is sent at a slightly higher mean PRF than the data, as defined in Table 99. This is due to the fact that length 31 or 127 ternary codes are being used within the SHR, and the number of chips within the SHR is no longer a power of 2. For example, for the two mandatory PRFs in channels {0:3, 5:6, 8:10, 12:14), the peak PRFs during the preamble are 31.2 MHz and 7.8 MHz, respectively, and the corresponding mean PRFs during the preamble are 16.10 MHz and 4.03 MHz, respectively. The corresponding mean PRFs during the PSDU are 15.60 MHz and 3.90 MHz, respectively. The remaining peak and mean PRF values for other optional UWB channels and the optional length 127 code are listed in Table 100.

Table 100—UWB PHY preamble parameters

Bands	Preamble						
Channel Number	C_i Code Length	Peak PRF (MHz)	Mean PRF (MHz)	Delta Length δ_L	#Chips Per Symbol	Symbol Duration T_{psym}(ns)	Base Rate Msymbol/s
{0:15}	31	31.20	16.10	16	496	993.59	1.01
{0:3, 5:6, 8:10, 12:14}	31	7.80	4.03	64	1984	3974.36	0.25
{0:15}	127	124.80	62.89	4	508	1017.63	0.98

The base symbol rate is defined as the rate at which the preamble symbols are sent. The base rates corresponding to the two mandatory mean PRFs of 16.10 MHz and 4.03 MHz are 1 Msymbol/s and 0.25 Msymbol/s, respectively, and are listed in the column with the heading "Base Rate" in Table 100. These symbol rates correspond to a preamble symbol duration, T_{psym}, of 993.59 ns and 3974.36 ns for the two mandatory PRFs.

Finally, for each UWB frame consisting of the SHR, SFD, PHR, and a data field, there are four possible durations of the SHR. This is due to the four possible lengths of SYNC field in the SHR, as described in 14.2.5. The SYNC field consists of repetitions of the preamble symbol. The number of preamble symbol repetitions are 16, 64, 1024, and 4096. These different SYNC field lengths yield different time durations of the UWB frame. The relationship between SYNC field length and frame duration is shown in Table 101. For each UWB channel, the number of chips in an individual preamble symbol is shown in the row titled "N_c." N_c is a function of the PRF used within the channel and, therefore, has either two or three values. For each value of N_c, the admissible preamble symbol durations T_{psym} are defined, and the duration of the SYNC portion of the SHR for each length (16, 64, 1024, or 4096) is denoted as T_{sync}. After the insertion of the SFD (the SFD may be either 8 or 64 preamble symbols long), the total length (in preamble symbols) of the SHR may any of the N_{pre} values shown in Table 101, and this in turn leads to the possible SHR durations denoted as T_{pre}. After creation of the SHR, the frame is appended with the PHR whose length, N_{hdr}, is 16 symbols

and duration is denoted as T_{hdr}. The values of the frame duration parameters are shown in Table 101 for each of the UWB channels.

Table 101—UWB PHY frame-dependent parameters

Parameter	Description		Value		
Channel	UWB PHY channel number		{0:15}		{0:3, 5:6, 8:10, 12:14}
PRF_{mean}	Mean PRF (MHz)		16.10	62.89	4.03
N_c	Number of chips per preamble symbol		496	508	1984
T_{psym}	Preamble symbol duration (ns)		993.6	1017.6	3974.4
N_{sync}	Number of symbols in the packet sync sequence	Short	16		
		Default	64		
		Medium	1024		
		Long	4096		
T_{sync}	Duration of the packet sync sequence (μs)	Short	15.9	16.3	63.6
		Default	63.6	65.1	254.4
		Medium	1017.4	1042.1	4069.7
		Long	4069.7	4168.2	N/A[a]
N_{sfd}	Number of symbols in the SFD		8 or 64		
T_{sfd}	Duration of the SFD (μs)		7.9 or 63.6	8.1 or 65.1	31.8 or 254.4
N_{pre}	Number of symbols in the SHR preamble	Short	24 or 80		
		Default	72 or 128		
		Medium	1032 or 1088		
		Long	4104 or 4160		
T_{pre}	Duration of the SHR preamble (μs)	Short	23.8 or 79.5	24.4 or 270.6	95.4 or 128.7
		Default	71.5 or 127.2	73.3 or 319.5	286.2 or 319.5
		Medium	1025.4 or 081.0	1050.2 or 1296.4	4101.5 or 4134.9
		Long	4077.7 or 4133.3	4176.3 or 4422.6	N/A[a]
N_{CCA_PHR}	Number of multiplexed preamble symbols in PHR		4 or 32		
N_{CCA_data}	Number of multiplexed preamble symbols in the data field		$T_{pre}/(4 \times T_{dsym/M})$		

[a]The use of the long SYNC sequence is not allowed when operating at a mean PRF of 4.03 MHz.

The motivation for two different PRFs stems from the fact that the devices will operate in environments with widely varying delay spreads. The low PRF is mainly intended for operation in environments with high delay spreads, particularly if the receiver uses energy detection. For coherent reception in high-delay-spread

environments and for any receivers in low-delay-spread environments, operation with the higher PRF is preferable.

The higher PRF allows the use of lower peak voltages for a fixed output power; note, however, that there is no requirement for the transmitter to transmit with maximum admissible (by the frequency regulators) output power. Thus, a transmitter can—if so desired by the designer—operate the low PRF with the same peak voltage as for the high-PRF case. Note that this case is still beneficial for noncoherent receivers as it reduces the intersymbol interference (compared to the high-PRF case).

Finally, it is noteworthy that the implementation of a dual PRF does not lead to a significant increase in the complexity of either transmitter or receiver since the PRFs are integer multiples of each other. The transmit signal corresponding to a low PRF can thus be generated with the same pulse generator as for the high PRF; the generator is simply excited less frequently. Similarly, the receive signal corresponding to the low PRF can be obtained by subsampling of the signal corresponding to the high PRF. Thus, even the synchronization procedure (when the PRF is unknown) can use the same sampler and ADC and just search both the fast-sampled (i.e., for high PRF) signal and (possibly in parallel) a subsampled version that is obtained from the fast-sampled signal by retaining only every fourth sample.

14.2.5 SHR preamble

An SHR preamble shall be added prior to the PHR to aid receiver algorithms related to AGC setting, antenna diversity selection, timing acquisition, coarse and fine frequency recovery, packet and frame synchronization, channel estimation, and leading edge signal tracking for ranging.

Four mandatory preambles are defined: a default preamble, a short preamble, a medium preamble, and a long preamble. The preamble to be used in the transmission of the current frame is determined by the value of the UWBPreambleSymbolsRepetitions parameter in the MCPS-DATA.request primitive.

Figure 86 shows the structure of the SHR preamble, which is subdivided into the SYNC field, as described in 14.2.5.1, and the SFD field, as described in 14.2.5.2. The duration of these fields is given in Table 101.

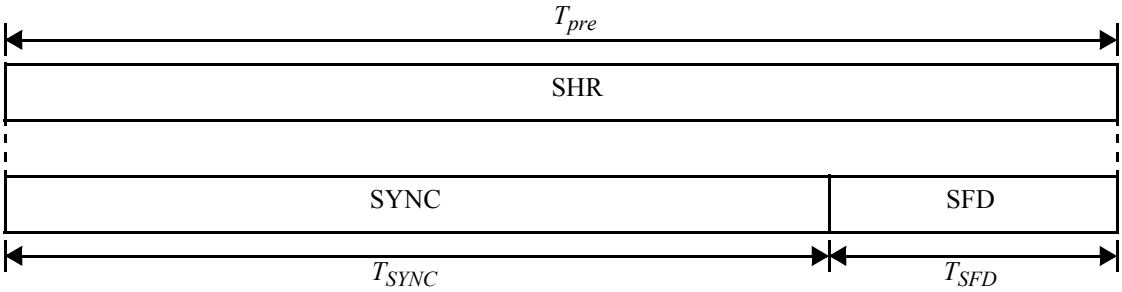


Figure 86—SHR preamble structure

14.2.5.1 SHR SYNC field

Each PAN operating on one of the UWB PHY channels {0–15} is also identified by a preamble code. The preamble code is used to construct symbols that constitute the SYNC portion of the SHR preamble as shown in Figure 86. The UWB PHY supports two lengths of preamble code: a length 31 code and an optional length 127 code. Each preamble code is a sequence of code symbols drawn from a ternary alphabet {–1,0,1} and selected for use in the UWB PHY because of their perfect periodic autocorrelation properties. The length 31 code sequences are shown in Table 102, while the length 127 code sequences are shown in Table 103 where they are indexed from 1–24 ($\boldsymbol{C_i}$ i = 1,2,...24). The first 8 codes (index 1–8) are length 31, while the remaining 16 (index 9–24) are length 127. Which codes may be used in each of the UWB PHY channels is restricted, and the particular code assignments are made in Table 102 and Table 103.

Specifically, the last column in each table indicates the set of UWB channel numbers that permit use of the code. This restriction of codes is to ensure that codes with the lowest cross-correlation are used in the same UWB PHY channel. Additionally, 8 of the length 127 codes are reserved for use with the private ranging protocol only and are not used during normal WPAN operation. This restriction is indicated in the third column of Table 103 as well.

Table 102—Length 31 ternary codes

Code index	Code sequence	Channel number[a]
1	-0000+0-0+++0+-000+-+++00-+0-00	0, 1, 8, 12
2	0+0+-0+0+000-++0-+---00+00++000	0, 1, 8, 12
3	-+0++000-+-++00++0+00-0000-0+0-	2, 5, 9, 13
4	0000+-00-00-++++0+-+000+0-0++0-	2, 5, 9, 13
5	-0+-00+++-+000-+0+++0-0+0000-00	3, 6, 10, 14
6	++00+00---+-0++-000+0+0-+0+0000	3, 6, 10, 14
7	+0000+-0+0+00+000+0++---0-+00-+	4, 7, 11, 15
8	0+00-0-0++0000--+00-+0++-++0+00	4, 7, 11, 15

[a]Note that codes indexed 1 through 6 may also be used for UWB channels 4, 7, 11, and 15 (i.e., channels whose bandwidth is wider that 500 MHz) if interchannel communication is desired.

Table 103—Optional length 127 ternary codes

Code index	Code sequence	Channel number[a]
9	+00+000-0--00--+0+0+00-+-++0+0000++-000+00-00--0-+0+0--0-+++0++000+-0+00-0++-0+++00-+00+0+0-0++-+--+000000+00000-+0000-0-000--+	0–4, 5, 6, 8–10, 12–14
10	++00+0-+00+00+000000-000-00--000-0+-+0-0+-0-+00000+-00++0-0+00--+00++-+0+-0+0000-0-0-0-++-+0+00+0+000-+0+++000----+++0000+++0--	0–4, 5, 6, 8–10, 12–14
11	-+-0000+00--00000-0+0+0+-0+00+00+0-00-+++00+000-+0+0-0000+++++-+0+--0+-0++--0-000+0-+00+0+----000-000000-+00+-0++000++-00++-0-0	0–4, 5, 6, 8–10, 12–14
12	-+0++000000-0+0-+0---+-++00-+0++0+0+0+000-00-00-+00+-++000-+-0-++0-0++++0-00-0++00+0+00++-00+000+-000-0--+0000-0000--0+00000+--	0–4, 5, 6, 8–10, 12–14
13	+000--0000--++0-++++0-0++0+0-00-+0++00++-0++0+-+0-00+00-0--000-+-00+0000-0++-00000+-0-000000-00-+-++-+000-0+0+0+++-00--00+0+000	0–15; DPS only
14	+000++0-0+0-00+-0-+0-00+0+0000+0+-0000++00+0+++++-+0-0+-0--+0++--000---0+000+0+0-+-000000+-+-0--00++000-00+00++-00--++-00-00000	0–15; DPS only
15	0+-00+0-000-++0000---++000+0+-0-+00-+000--0-00--0--+++-+0-++00+-++0+00000+0-0+++-00+00+000-0000+00--+0++0+0+0-00-0-+-0+0++00000	0–15; DPS only
16	++0000+000+00+--0+-++0-000--00+-0+00++000+++00+0+0-0-+-0-0+00+00+0++----+00++--+0+-0--+000000-0-0000-+0--00+00000+-++000-0-+0+0	0–15; DPS only
17	+--000-0-0000+-00000+000000+--+-++0-0+0+00+-00+++0-++0-00+0-+000++0+++-0--0+0+-0--00-00+000-++0000+0++-+-00+0+0+--00--0-000+00+	4, 7, 11, 15

Table 103—Optional length 127 ternary codes *(continued)*

Code index	Code sequence	Channel number[a]
18	--0+++0000+++----000+++0+-000+0+00+0+-++-0-0-0-0000+0-+0+-++00+--00+0-0++00-+00000+-0-+0-0+-+0-000--00-000-000000+00+00+-0+00++	4, 7, 11, 15
19	-0-++00-++000++0-+00+-000000-000----+0+00+-0+000-0--++0-+0--+0+-+++++0000-0+0+-000+00+++-00-0+00+00+0-+0+0+0-00000--00+0000-+-0	4, 7, 11, 15
20	--+00000+0--0000-0000+--0-000-+000+00-++00+0+00++0-00-0++++0-0++-0-+-000++-+00+-00-00-000+0+0+0++0+-00++-+---0+-0+0-000000++0+-	4, 7, 11, 15
21	+0+00--00-+++0+0+0-000+-++-+-00-000000-0-+00000-++0-0000+00-+-000--0-00+00-0+-+0++0-++00++0+-00-0+0++0-0++++-0++--0000--000+000	0–15; DPS only
22	0-00-++--00-++00+00-000++00--0-+-+000000-+-0+0+000+0---000--++0+--0-+0-0+-+++++0+00++0000-+0+0000+0+00-0+-0-+00-0+0-0++000+0000	0–15; DPS only
23	000++0+0-+-0-00-0+0+0++0+--00+0000-000+00+00-+++0-0+00000+0++-+00++-0+-+++--0--00-0--000+-00+-0-+0+000++---0000++-000-0+00-+000	0–15; DPS only
24	+0+-0-000++-+00000+00--0+-0000-0-000000+--0-+0+--++00+----++0+00+00+0-0-+-0-0+0+00+++000++00+0-+00--000-0++-+0--+00+000+0000++0	0–15; DPS only

[a]Note that codes indexed 9 through 13 may also be used for UWB channels 4, 7, 11, and 15 (i.e., channels whose bandwidth is wider than 500 MHz) if interchannel communication is desired.

Note that the assignment of preamble codes to channels has been done to enable interchannel communication. In other words, it is possible that a device operating on a wideband channel {4,7,11,15} may communicate with a device on a channel with which it overlaps.

For a WPAN using the ternary code indexed by i, the SYNC field shall consist of N_{sync} repetitions of the symbol $\boldsymbol{S_i}$, where $\boldsymbol{S_i}$ is the code $\boldsymbol{C_i}$ spread by the delta function δ_L of length L as shown in Table 100. The spreading operation, where code $\boldsymbol{C_i}$ is extended to the preamble symbol duration indicated in Table 100, is described mathematically by:

$$\boldsymbol{S}_i = \boldsymbol{C}_i \otimes \boldsymbol{\delta}_L(n)$$

$$\delta_L(n) = \begin{cases} 1 & n = 0 \\ 0 & n = 1, 2\ldots, L-1 \end{cases}$$

where the operator $\otimes$ indicates a Kronecker product. After the Kronecker operation, a preamble symbol is formed as depicted in Figure 87, where $L - 1$ zeros have been inserted between each ternary element of $\boldsymbol{C_i}$.

The spreading factor L, number of chips per symbol, preamble symbol duration T_{psym}, and base symbol rate for different channels are given in Table 100.

14.2.5.2 SHR SFD

An SFD shall be added to establish frame timing. The UWB PHY uses a short SFD for default and medium data rates and a long SFD for the optional low data rate of 110 kb/s as shown in Figure 86. The short SFD shall be [0 +1 0 –1 +1 0 0 –1] spread by the preamble symbol S_i, where the leftmost bit shall be transmitted first in time. The long SFD shall be obtained by spreading the sequence [0 +1 0 –1 +1 0 0 –1 0 +1 0 –1 +1 0 0 –1 –1 0 0 +1 0 –1 0 +1 0 +1 0 0 0 –1 0 –1 0 –1 0 0 +1 0 –1 –1 0 –1 +1 0 0 0 0 +1 +1 0 0 –1 –1 –1 +1 –1 +1 +1 0 0 0 0 +1 +1] by the preamble symbol S_i. Note that the long SFD is eight times longer than the short SFD

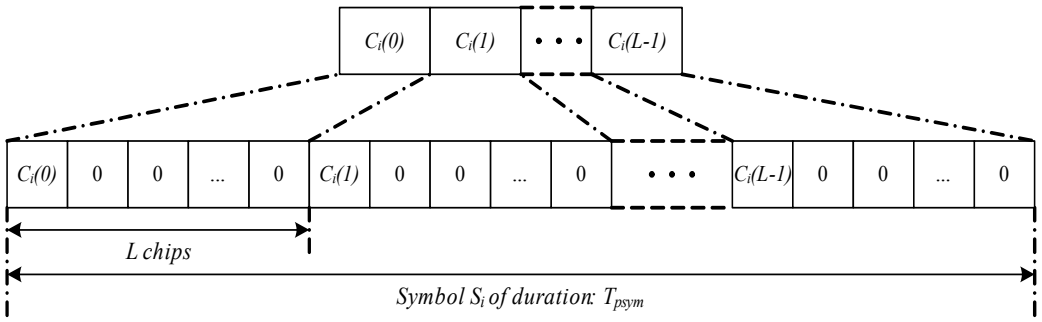


Figure 87—Construction of symbol S_i from code C_i

and consists of 64 preamble symbols, only 32 of which are active, and the other 32 are zeros. The structure of the SHR preamble and the two possible SFDs are shown in Figure 86.

14.2.6 PHY header (PHR)

A PHR, as shown in Figure 88, shall be added after the SHR preamble. The PHR consists of 19 bits and conveys information necessary for a successful decoding of the packet to the receiver. The PHR contains information about the data rate used to transmit the PSDU, the duration of the current frame's preamble, and the length of the frame payload. Additionally, six parity check bits are used to further protect the PHR against channel errors.

Bit 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
R1	R0	L6	L5	L4	L3	L2	L1	L0	RNG	EXT	P1	P0	C5	C4	C3	C2	C1	C0
Data Rate		Frame Length							Ranging Packet	Header Extension	Preamble Duration		SECDED Check Bits					

Figure 88—PHR bit assignment

The PHR shall be transmitted using the BPM-BPSK modulation outlined in 14.3.

14.2.6.1 PHR rate, length, ranging, extension, preamble duration fields

The Data Rate field indicates the data rate of the received PSDU. The bits shall be set, dependent on the mean PRF, according to Table 105. The default value of the bits shall be set to 01 as this is the only mandatory data rate that is supported by a UWB-compliant PHY implementation. Support for other data rates listed in Table 105 is optional.

The Frame Length field shall be an unsigned integer number that indicates the number of octets in the PSDU that the MAC sublayer is currently requesting the PHY to transmit.

The Ranging Packet bit, RNG, shall be set to one if the current frame is an RFRAME; otherwise, it shall be set to zero.

The Header Extension bit, EXT, is reserved for future extension of the PHR. This bit shall be set to zero.

The Preamble Duration field represents the length (in preamble symbols) of the SYNC portion of the SHR. The Preamble Duration field shall be set according to Table 104. The default setting Preamble Duration setting is 01, which corresponds to a SYNC field of length 64 preamble symbols.

Table 104—Preamble Duration field values

P1–P0	SYNC length (symbols) (Si)
00	16
01	64
10	1024
11	4096

The Preamble Duration field is intended for use during ranging operations and is used by a receiver of the PHY frame to help determine at which preamble symbol the UWB PHY acquired and began tracking the preamble. A receiver may use the Preamble Duration field to set the value of its own preamble duration based upon the received value when communicating a ranging ACK packet.

Table 105—Nominal data rates

R1–R0	Mean PRF 15.60 or 62.40 MHz	Mean PRF 3.90 MHz
00	0.11	0.11
01	0.85	0.85
10	6.81	1.70
11	27.24	6.81

14.2.6.2 PHR SECDED check bits

The SECDED (single error correct, double error detect) field is a set of six parity check bits that are used to protect the PHR from errors caused by noise and channel impairments. The SECDED bits are a simple Hamming block code that enables the correction of a single error and the detection of two errors at the receiver. The SECDED bit values depend on PHR bits 0–12 and are computed as follows:

C0 = XOR(R0, R1, L0, L2, L4, L5, EXT, P1)

C1 = XOR(R1, L2, L3, L5, L6, RNG, EXT, P0)

C2 = XOR(R0, L0, L1, L5, L6, RNG, EXT)

C3 = XOR(L0, L1, L2, L3, L4, RNG, EXT)

C4 = XOR(P0, P1)

C5 = XOR(R1, R0, L6, L5, L4, L3, L2, L1, L0, RNG, EXT, P1, P0, C4, C3, C2, C1, C0)

14.2.7 Data field

The Data field is encoded as shown in Figure 89.

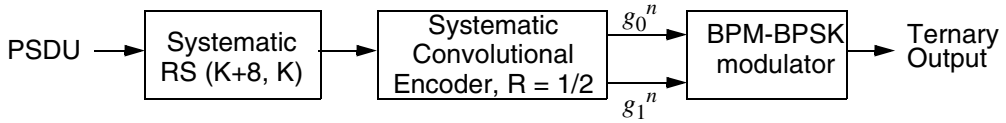


Figure 89—Data field encoding process

The data field shall be formed as follows:

— Encode the PSDU using systematic Reed-Solomon block code, which adds 48 parity bits as described in 14.3.3.1.

— Encode the output of the Reed-Solomon block code using a systematic convolutional encoder as described in 14.3.3.210.2, except in the cases where the Viterbi rate for the modulation is one, as defined in Table 99. In these cases, the convolutional encoder is bypassed.

— Spread and modulate the encoded block using BPM-BPSK modulation as described in 14.3.

14.3 UWB PHY modulation

14.3.1 UWB PHY modulation mathematical framework

The UWB PHY transmit waveform during the *k*th symbol interval may be expressed as:

$$x^{(k)}(t) = [1 - 2g_1^{(k)}] \sum_{n=1}^{N_{cpb}} [1 - 2s_{n+kN_{cpb}}] \times p(t - g_0^{(k)} T_{BPM} - h^{(k)} T_{burst} - nT_c)$$

This equation describes the time hopping with polarity scrambling, which improvesthe interference rejection capabilities of the UWB PHY. The *k*th symbol interval carries two information bits $g_0^{(k)}$ and $g_1^{(k)} \in \{0, 1\}$. Bit $g_0^{(k)}$ is encoded into the burst position, whereas bit $g_1^{(k)}$ is encoded into the burst polarity. The sequence $s_{n+kN_{cpb}} \in \{0, 1\}, n = 0, 1, \ldots, N_{cpb} - 1$ is the scrambling code used during the *k*th symbol interval, $h^{(k)} \in \{0, 1 - N_{hop} - 1\}$ is the *k*th burst hopping position, and $p(t)$ is the transmitted pulse shape at the antenna input. The burst hopping sequence $h^{(k)}$ provides for multiuser interference rejection. The chip scrambling sequence $s_{n+kN_{cpb}}$ provides additional interference suppression among coherent receivers as well as spectral smoothing of the transmitted waveform. Note that the equation defines the transmitted signal during the valid burst interval; at all other possible burst positions, no signal shall be transmitted. A reference modulator illustrating the BPM-BPSK modulation is shown in Figure 90.

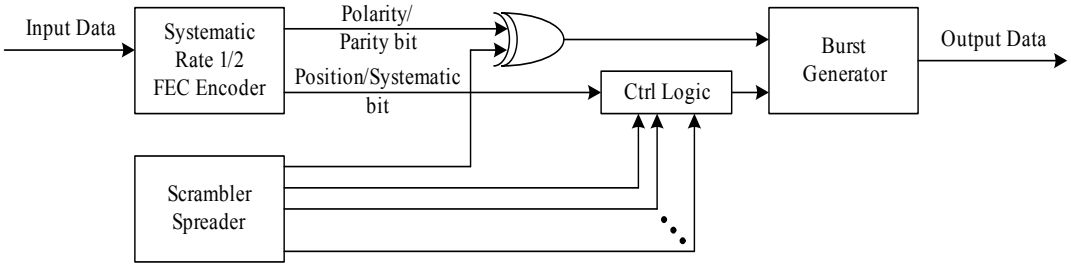


Figure 90—Reference symbol modulator

Note here that the FEC encoder is not included if the modulation Viterbi rate is one, as described in 14.2.7. In this case, the FEC encoder is replaced by a multiplexer that shall apply even bits to the position input and odd bits to the polarity input.

14.3.2 UWB PHY spreading

The time-varying spreader sequence $s_{n+kN_{cpb}}$ and the time-varying burst hopping sequence $h^{(k)}$ shall be generated from a common PRBS scrambler.

The polynomial for the scrambler generator shall be: $g(D) = 1 + D^{14} + D^{15}$

where D is a single chip delay, T_c, element. This polynomial forms not only a maximal length sequence, but also it is a primitive polynomial. By the given generator polynomial, the corresponding scrambler output is generated as:

$$s_n = s_{n-14} \oplus s_{n-15} \qquad n = 0, 1, 2, \ldots$$

where $\otimes$ denotes modulo-2 addition.

A linear feedback shift register (LFSR) realization of the scrambler is shown in Figure 91. The LFSR shall be initialized upon the transmission of bit 0 of the PHR. Note that N_{cpb} may change depending on the data rate and PRF in use during the PSDU. The LFSR shall not be reset after transmission of the PHR.

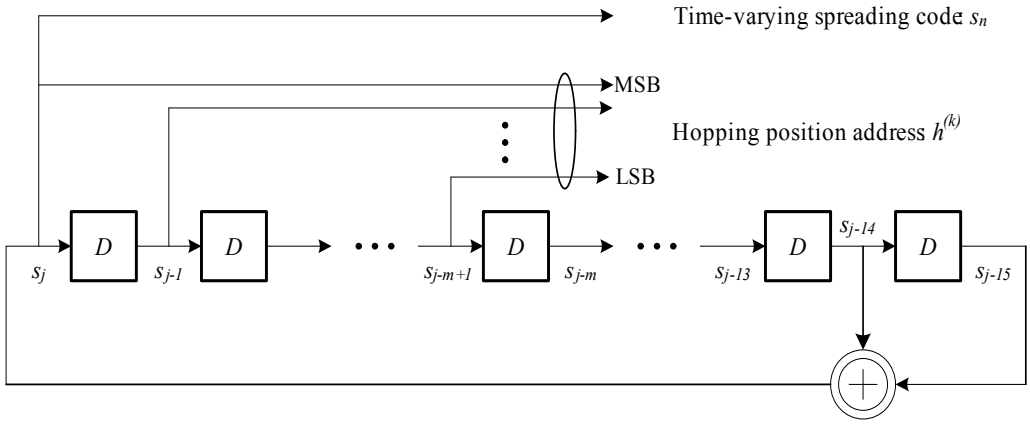


Figure 91—LFSR implementation of the scrambler

The initial state of the LFSR shall be determined from the preamble code by first removing all the zeros in the ternary code and then replacing all the negative ones with a zero. The first 15 bits of the resulting binary state shall be loaded into the LFSR. Table 106 shows an example of this procedure for preamble code, C_6 (length 31, preamble code index 6, as defined in Table 102). The table shows the initial state as well as the first 16 output bits from the scrambler.

Table 106—Example LFSR initial state for preamble code 6

Initial state (s_{-15}, s_{-14}, …, s_{-1})	**LFSR output: First 16 bits** s_0, s_1, …, s_{15} (s_0 first in time)
111000101101101	0010011101101110

Note that even though each device within a PAN uses the same initial LFSR setting, the communication in WPAN is asynchronous so that the hopping and scrambling provides interference rejection.

The LFSR shall be clocked at the peak PRF of 499.2 MHz as specified in Table 99. During the *k*th symbol interval, the LFSR shall be clocked N_{cpb} times, and the scrambler output shall be the *k*th scrambling code $s_{n+kN_{cpb}}$, $n = 0, 1, \ldots, N_{cpb} - 1$. Furthermore, the *k*th burst hopping position shall be computed as follows:

$$h^{(k)} = 2^0 s_{kN_{cpb}} + 2^1 s_{1+kN_{cpb}} + \cdots + 2^{m-1} s_{m-1+kN_{cpb}}$$

where $m = \log_2(N_{hop})$.

As shown in Table 99, the number of hopping burst N_{hop} is always a power of two, and consequently, m is always an integer. Note that for $N_{cpb} < m$, the LFSR is clocked N_{cpb} times, not m times.

For the mandatory modes with mean data PRFs of 15.60 MHz and 3.90 MHz, the numbers of hopping bursts are 8 and 32, respectively, as indicated in Table 99, and consequently, m takes on the values 3 and 5, respectively. The corresponding hopping sequences are as follows:

$$h^{(k)} = s_{kN_{cpb}} + 2s_{1+kN_{cpb}} + 4s_{2+kN_{cpb}} \quad \text{Mean PRF = 15.60 MHz}$$

$$h^{(k)} = s_{kN_{cpb}} + 2s_{1+kN_{cpb}} + 4s_{2+kN_{cpb}} + 8s_{3+kN_{cpb}} + 16s_{4+kN_{cpb}} \quad \text{Mean PRF = 3.90 MHz}$$

14.3.3 UWB PHY forward error correction (FEC)

The FEC used by the UWB PHY is a concatenated code consisting of an outer Reed-Solomon systematic block code and an inner half-rate systematic convolutional code. The inner convolutional code is not necessarily enabled at all data rates; the rows of Table 99 that have a Viterbi rate of 1 indicate that the inner convolutional code is disabled for the PSDU part of the PHY frame.

The FEC encoding of a block of M PSDU bits, $b_0, b_1, \ldots, b_{M-1}$, is shown in Figure 92. The Reed-Solomon encoder shall append 48 parity bits, $p_0, p_1, \ldots, p_{47}$, to the original block. This results in a Reed-Solomon encoded block of length $M + 48$. When the Viterbi rate is 0.5, a half-rate systematic convolutional encoder shall encode the Reed-Solomon encoded block into a systematic coded block of length $2M + 96$ bits. The convolutional systematic bits shall be used to encode the position of the burst, whereas the convolutional parity bits shall be used to encode the polarity of the pulses within a burst. When the Viterbi rate is one, even outputs of the Reed-Solomon encoder ($b_0, b_2, \ldots, b_{M-2}, p_0, p_2, \ldots, p_{46}$) shall be used to encode the position of the burst, and odd outputs ($b_1, b_3, \ldots, b_{M-1}, p_1, p_3, \ldots, p_{47}$) shall be used to encode the polarity of the pulses. Note here that M is always an even number.

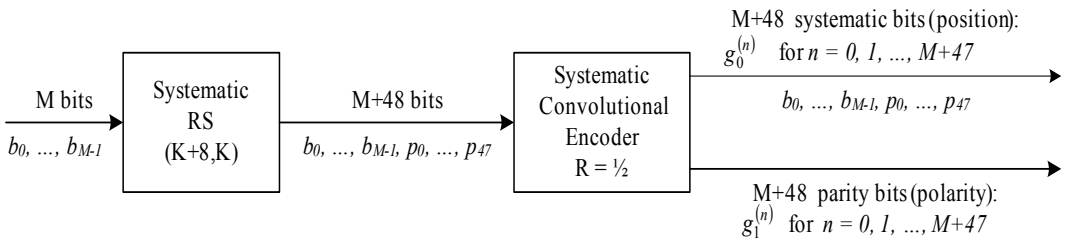


Figure 92—FEC encoding process

A noncoherent receiver cannot see the convolutional parity bits (parity bits), and consequently, a noncoherent receiver may use only a Reed-Solomon decoder to improve its performance. A coherent receiver may use either or both Reed-Solomon and convolutional decoding algorithms. Note here that since both the Reed-Solomon and the convolutional codes are both systematic, a receiver (either coherent or noncoherent) may be implemented without an FEC decoder. In this case, the information bits are simply recovered by demodulating the position of the burst. There will be additional parity check bits as a result of the Reed-Solomon encoding, but these may be simply ignored.

14.3.3.1 Reed-Solomon encoding

The systematic Reed-Solomon code is over the Galois field, GF(2^6), which is built as an extension of GF(2). The systematic Reed-Solomon code shall use the generator polynomial

$$g(x) = \prod_{k=1}^{8} (x + \alpha^k) = x^8 + 55x^7 + 61x^6 + 37x^5 + 48x^4 + 47x^3 + 20x^2 + 6x^1 + 22$$

where α = 010000 is a root of the binary primitive polynomial $1 + x + x^6$ in GF(2^6).

In Reed-Solomon encoding $RS_6(K + 8, K)$, a block of I bits (with $K = \lceil I/6 \rceil$) is encoded into a codeword of $I + 48$ bits. The Reed-Solomon encoding procedure is performed in the following steps:

a) *Addition of dummy bits*. The block of I information bits is expanded by adding $330 - I$ dummy (zero) bits to the beginning of the block. The expanded block is denoted as $\{d_0, d_1, \ldots, d_{329}\}$ where d_0 is the first in time.

b) *Bit-to-symbol conversion*. The 330 bits $\{d_0, d_1, \ldots, d_{329}\}$ are converted into 55 Reed-Solomon symbols $\{D_0, D_1, \ldots, D_{54}\}$ having the following polynomial representation:

$$D_k = \alpha^5 d_{6k+5} + \alpha^4 d_{6k+4} + \alpha^3 d_{6k+3} + \alpha^2 d_{6k+2} + \alpha d_{6k+1} + d_{6k}, \text{ where } k = 0{:}\ 54$$

Resulting 6-bit symbols are presented as $D_k = \{d_{6k+5}, d_{6k+4}, d_{6k+3}, d_{6k+2}, d_{6k+1}, d_{6k}\}$, where d_{6k+5} is the MSB and d_{6k} is the LSB.

c) *Encoding*. The information symbols $\{D_0, D_1, \ldots, D_{54}\}$ are encoded by systematic $RS_6(63,55)$ code with output symbols $\{U_0, U_1, \ldots, U_{62}\}$ ordered as follows:

$$U_k = \begin{cases} D_k & (k = 0, 1, \ldots, 54) \\ P_k & (k = 55, 56, \ldots, 62) \end{cases}$$

where P_k are parity check symbols added by $RS_6(63,55)$ encoder.

The information polynomial associated with the information symbols $\{D_0, D_1, \ldots, D_{54}\}$ is denoted as $D(x) = x^{54}D_0 + x^{53}D_1 + \ldots + xD_{53} + 54$. The parity check polynomial associated with the parity check symbols is denoted as $P(x) = x^7P_{55} + x^6P_{56} + \ldots + xP_{61} + P_{62}$. The parity check symbols are calculated as:

$$P(x) = \text{remainder}[x^8 D(x) / g(x)]$$

$$U(x) = x^8 D(x) + P(x)$$

d) *Symbol-to-bit conversion*. The output symbols $\{U_0, U_1, \ldots, U_{62}\}$ are converted into binary form with LSB coming out first, resulting in a block of 378 bits $\{u_0, u_1, \ldots, u_{377}\}$.

e) *Removal of dummy bits*. The $330 - I$ dummy bits added in the first step are removed. Only the last $I + 48$ bits are transmitted, i.e., $\{u_{330-I}, u_{331-I}, \ldots, u_{377}\}$ with u_{330-I} being first in time.

14.3.3.2 Systematic convolutional encoding

The inner convolutional encoder shall use the rate $R = ½$ code with generator polynomials $g_0 = [010]_2$ and $g_1 = [101]_2$ as shown in Figure 93. Upon transmission of each PPDU, the encoder shall be initialized to the all zero state. Additionally, the encoder shall be returned to the all zero state by appending two zero bits to the PPDU. Note that since the generator polynomials are systematic, they are also noncatastrophic.

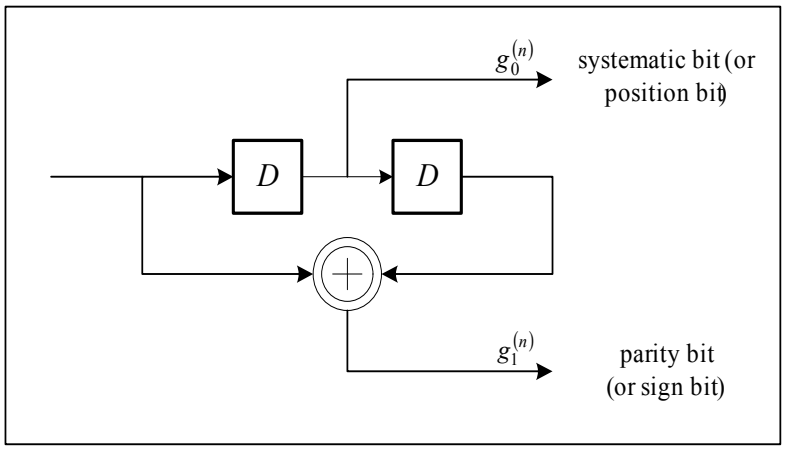


Figure 93—Systematic convolutional encoder

14.4 UWB PHY RF requirements

14.4.1 Operating frequency bands

The set of operating frequency bands is defined in Table 107. For the sub-gigahertz operation, channel 0 is defined as the mandatory channel; for the low-band operation, channel 3 is the mandatory channel; and for the high-band operation, channel 9 is the mandatory channel.

Table 107—UWB PHY band allocation

Band group[a] (decimal)	Channel number (decimal)	Center frequency, f_c (MHz)	Band width (MHz)	Mandatory/Optional
0	0	499.2	499.2	Mandatory below 1 GHz
1	1	3494.4	499.2	Optional
	2	3993.6	499.2	Optional
	3	4492.8	499.2	Mandatory in low band
	4	3993.6	1331.2	Optional

Table 107—UWB PHY band allocation *(continued)*

Band group[a] (decimal)	Channel number (decimal)	Center frequency, f_c (MHz)	Band width (MHz)	Mandatory/Optional
2	5	6489.6	499.2	Optional
	6	6988.8	499.2	Optional
	7	6489.6	1081.6	Optional
	8	7488.0	499.2	Optional
	9	7987.2	499.2	Mandatory in high band
	10	8486.4	499.2	Optional
	11	7987.2	1331.2	Optional
	12	8985.6	499.2	Optional
	13	9484.8	499.2	Optional
	14	9984.0	499.2	Optional
	15	9484.8	1354.97	Optional

[a]Note that bands indicate a sequence of adjacent UWB center frequencies: band 0 is the sub-gigahertz channel, band 1 has the low-band UWB channels, and band 2 has the high-band channels.

Figure 94 is a graphical representation of the data presented in Table 107. Each UWB PHY channel is shown as a heavy black line centered on the channel's center frequency. The length of the lines depicts the channel bandwidth.

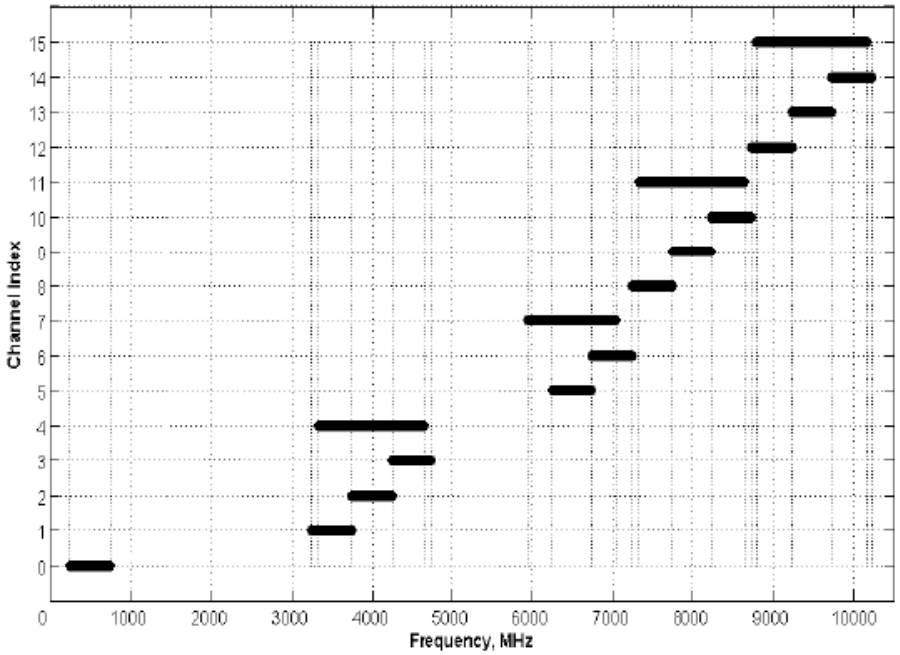


Figure 94—UWB PHY band plan

14.4.2 Channel assignments

A total of 32 complex channels are assigned for operation, two channels in each of the 16 defined operating frequency bands. A compliant implementation shall support at least the two channels for one of the mandatory bands.

14.4.3 Regulatory compliance

The maximum allowable output PSD shall be in accordance with practices specified by the appropriate regulatory bodies.

14.4.4 Operating temperature range

A conformant implementation shall meet all of the specifications in this standard for ambient temperatures from 0 °C to 40 °C.

14.4.5 Baseband impulse response

The transmitted pulse shape $p(t)$ of the UWB PHY shall be constrained by the shape of its cross-correlation function with a standard reference pulse, $r(t)$. The normalized cross-correlation between two waveforms is defined as:

$$\phi(\tau) = \frac{1}{\sqrt{E_r E_p}} \mathrm{Re} \int_{-\infty}^{\infty} r(t) p^*(t+\tau) dt$$

where E_r and E_p are the energies of $r(t)$ and $p(t)$, respectively. The reference $r(t)$ pulse used in the calculation of $|\phi(\tau)|$ is a root raised cosine pulse with a roll-off factor of $\beta = 0.5$. Mathematically this is:

$$r(t) = \frac{4\beta}{\pi\sqrt{T_p}} \frac{\cos[(1+\beta)\pi t/T_p] + \dfrac{\sin[(1-\beta)\pi t/T_p]}{4\beta(t/T_p)}}{1-(4\beta t/T_p)^2}$$

where T_p is the inverse of the chip frequency. Table 108 shows the required pulse duration for each channel.

Table 108—Required reference pulse durations in each channel

Channel number	Pulse duration, T_p (ns)	Main lobe width, T_w (ns)
{0:3, 5:6, 8:10, 12:14}	2.00	0.5
7	0.92	0.2
{4, 11}	0.75	0.2
15	0.74	0.2

In order for a UWB PHY transmitter to be compliant with this standard, the transmitted pulse $p(t)$ shall have a magnitude of the cross-correlation function $|\phi(\tau)|$ whose main lobe is greater than or equal to 0.8 for a duration of at least T_w, as defined in Table 108, and any sidelobe shall be no greater than 0.3. For the

purposes of testing a pulse for compliance, the following are defined: Let $|\phi(\tau)|$ be the magnitude of the cross-correlation of $p(t)$ and $r(t)$, and let τ_i, for $i = 1,2,...$, be a set of critical points, i.e, points at which

$$\frac{d}{d\tau}|\phi(\tau)|\Big|_{\tau = \tau_i} = 0$$

The maximum of the function occurs at one of these critical points, τ_{max}, where $|\phi(\tau_{max})| \geq |\phi(\tau)|$ for all values of τ. The requirement thus states that for some continuous set of values that contain the point τ_{max}, the function $|\phi(\tau)|$ is greater than 0.8. In addition, the second constraint on the value of sidelobes may be stated mathematically as $|\phi(\tau_i)| \leq 0.3$ for all τ_i.

Figure 95 shows an example UWB-compliant pulse, $p(t)$ (left plot), along with the root raised cosine reference pulse $r(t)$ (middle plot) with $T_p = 2.0$ ns and the magnitude of the cross-correlation $|\phi(\tau)|$ (right plot). The pulse $p(t)$ is an 8 order butterworth pulse with a 3 dB bandwidth of 500 MHz. Figure 95 is intended to show that this example pulse meets the requirements for compliance. Specifically, the main lobe is above 0.8 for nearly 1 ns, and no sidelobe is greater than 0.3 (in this case, the largest sidelobe peak is 0.2). The pulse $p(t)$ is a compliant pulse for channels {0:3, 5:6, 8:10, 12:14}.

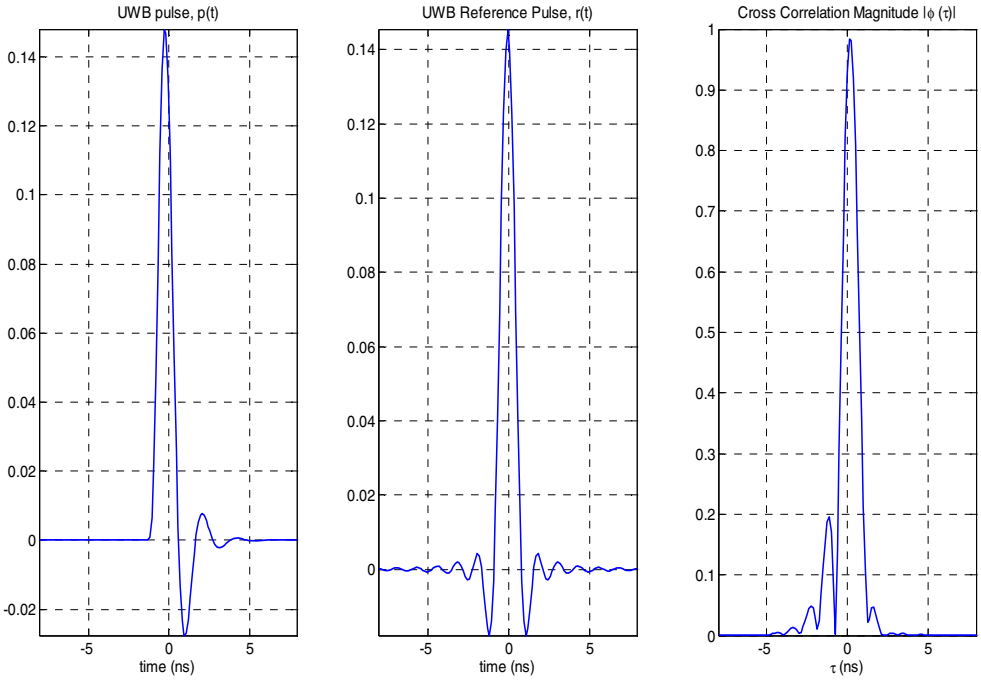


Figure 95—Compliant pulse example

Note that it is not the intention of this standard to imply that pulse shaping only occurs at baseband, but rather that the measurements described here occur on the pulse envelope if shaping is done at passband.

14.4.6 Transmit PSD mask

The transmitted spectrum shall be less than –10 dB relative to the maximum spectral density of the signal for $0.65/T_p < |f - f_c| < 0.8/T_p$ and –18 dB for $|f - f_c| > 0.8/T_p$. For example, the transmit spectrum mask for channel 4 is shown in Figure 96. The measurements shall be made using a 1 MHz resolution bandwidth and a 1 kHz video bandwidth.

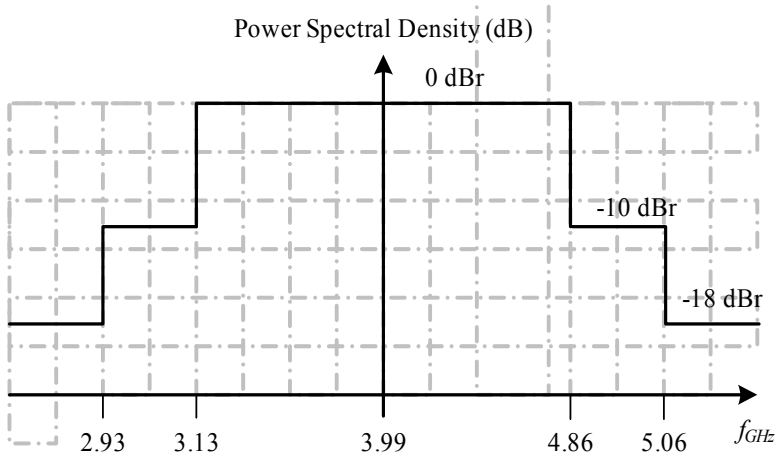


Figure 96—Transmit spectrum mask for band 4

14.4.7 Chip rate clock and chip carrier alignment

A UWB transmitter shall be capable of chipping at the peak PRF given in Table 99 with an accuracy of ±20 ppm. In addition, for each UWB PHY channel, the center of transmitted energy shall be within the values listed in Table 107 also with an accuracy of ±20 ppm. The measurements shall be made using a 1 MHz resolution bandwidth and a 1 kHz video bandwidth.

14.4.8 TX-to-RX turnaround time

The UWB PHY shall have a TX-to-RX turnaround time as defined in 8.2.1.

14.4.9 RX-to-TX turnaround time

The UWB PHY shall have a TX-to-RX turnaround time as defined in 8.2.2.

14.4.10 Transmit center frequency tolerance

The UWB PHY transmit center frequency tolerance shall be ±20 ppm. The tolerance on the chipping clock given in 14.4.7 takes precedence over this requirement.

14.4.11 Transmit power

The UWB PHY has no minimum transmit power requirement.

14.4.12 Receiver maximum input level of desired signal

The UWB PHY shall have a receiver maximum input level -45 dBm/MHz, using the measurement defined in 8.2.4.

14.4.13 Receiver ED

The UWB PHY shall provide the receiver ED measurement as described in 8.2.5. The averaging period for the receiver ED measurement is implementation specific.

The ED measurement for each channel may be performed as a series of measurements, each made at a fraction of the total channel bandwidth, in which case *phyUWBScanBinsPerChannel* specifies the number of frequency increments used. When this value is greater than one, the ED result reported using the MLME-SCAN.confirm primitive shall be a list of ED measurements, one for each frequency increment measurement. An implementation may provide multiple ED measurements, for example, to provide information to a higher layer that detects non-UWB services for the purpose of active detect and avoid (DAA) procedures as may be required in some environments.

14.4.14 Link quality indicator (LQI)

The UWB PHY shall provide the LQI measurement as described in 8.2.6.

14.4.15 Clear channel assessment (CCA)

A UWB PHY device shall support at least one of the CCA modes defined in 8.2.7. For CCA mode 6, the CCA detection time for the UWB PHY shall be equal to 40 mandatory symbol periods, which includes at least 8 (multiplexed) preamble symbols, as described in 14.6.

14.5 UWB PHY optional pulse shapes

The UWB PHY offers the capability to transmit several optional pulse types. The use of these options is controlled by the PAN coordinator and shall be limited to the nonbeacon frames. In other words, beacon frames shall be transmitted using the mandatory pulse shape as defined in 14.4.5, but all other frames may be transmitted using the optional pulse shapes if all devices in the PAN are capable of supporting the optional pulse shapes. PANs that use the optional pulse shapes shall indicate the use of a specific option via the *phyUWBCurrentPulseShape* PIB attribute. Devices choosing to join a PAN using one of the optional pulse shapes should make their decision based on the value of *phyUWBCurrentPulseShape* that is reported during the scan procedure.

14.5.1 UWB PHY optional chirp on UWB (CoU) pulses

The purpose of CoU pulses is to provide an additional dimension (besides frequency and DS codes) to support simultaneously operating piconets. Because CoU is an optional mode of pulse shapes in addition to the mandatory pulse shape, all modulation specifications shall be the same as they are for the mandatory pulse shape except those defined for the CoU pulses when a device implements the CoU option.

A mathematical representation of a CoU pulse at baseband is given by Equation (2):

$$p_{CoU}(t) = \begin{cases} p(t)\exp\left(-j\frac{\pi\beta t^2}{2}\right) & -\frac{T}{2} \le t \le \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where

$p(t)$ denotes a mandatory pulse shape that satisfies constraints in 14.4.5

$\beta = B/T$ is the chirping rate (chirping slope). Moreover, B and T are the bandwidth and time duration of the CoU pulse, respectively.

A graphical example of CoU pulse is shown in Figure 97.

The CoU is an operation added to the mandatory pulse. When a CoU pulse is transmitted, the receiver needs to perform a matched de-chirp operation to demodulate the signal.

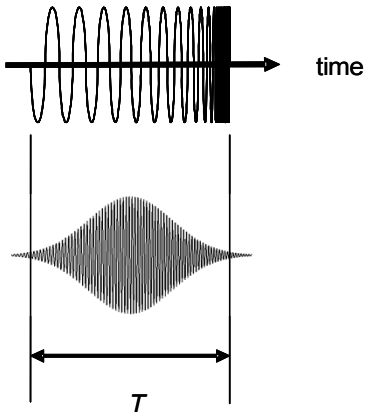


Figure 97—Graphical view of a CoU pulse

The optional CoU pulses are admitted with two slopes per each DS code per each 500 MHz bandwidth. The chirp slopes are denoted as CCh.1 and CCh.2. Within channels 4, 7, 11, and 15, there are chirp slopes admitted per each DS code. These are denoted as CCh.3 through CCh.6. The values for each chirp slope are listed in Table 109.

Table 109—CoU channel slopes

CoU number	β (slopes)
CCh.1	500 MHz/2.5 ns
CCh.2	–500 MHz/2.5 ns
CCh.3	1 GHz/5 ns
CCh.4	–1 GHz/5 ns
CCh.5	1 GHz/10 ns
CCh.6	–1 GHz/10 ns

14.5.2 UWB PHY optional continuous spectrum (CS) pulses

This subclause specifies optional CS pulses. A CS pulse is obtained by passing the mandatory pulse through an all-passing CS filter. The CS filter introduces controlled group delays to the input pulse. The purpose of the optional CS pulses is to reduce the interference level between different PANs.

Since CS is an optional mode of pulse shapes in addition to the mandatory pulse shape, all modulation specifications shall be the same as they are for the mandatory pulse shape except those defined for the CS pulses when a device implements the CS option.

An optional CS pulse $p_{CS}(t)$ is defined by Equation (3):

$$p_{CS}(t) = \int P(f)\exp[-j2\pi f(t-(\tau \times f))]df \tag{3}$$

where

τ represents the group delay (s/Hz)

$P(f)$ represents the Fourier transform of $p(t)$, where $p(t)$ is any pulse shape that meets the requirements defined in 14.4.5

The Fourier transform is defined by Equation (4):

$$P(f) = \int p(t) \exp[-j2\pi ft] dt \tag{4}$$

The CS filtering is an operation added to the mandatory pulse. When a CS pulse is transmitted, the receiver needs to perform an inverse CS filtering (CS^{-1}) operation to demodulate the signal.

Some examples of CS pulses are shown in Figure 98.

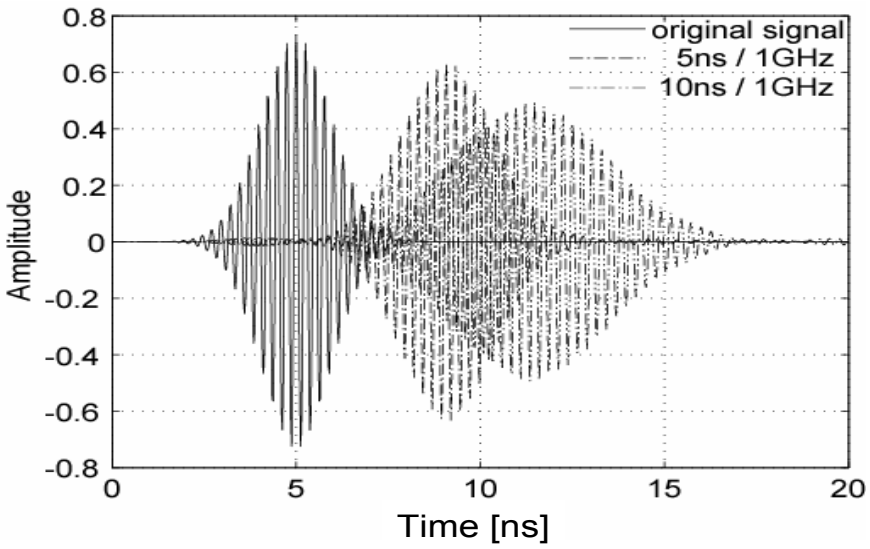


Figure 98—Examples of CS pulses

Each 500 MHz band shall use No.1 or No.2 pulses, while each 1.5 GHz band shall use one of No.3 through No.6 pulses, as defined in Table 110.

Table 110—CS group delays

CS pulse number	τ (Group delay)
No.1	2 ns/500 MHz
No.2	–2 ns/500 MHz
No.3	5 ns/1 GHz
No.4	–5 ns/1 GHz
No.5	10 ns/1 GHz
No.6	–10 ns/1 GHz

14.5.3 UWB PHY linear combination of pulses (LCP)

LCP is an optional pulse shape that can be used in regulatory regions where DAA schemes are required by regulators. Using LCP pulses enables a PAN to limit interference to incumbent wireless systems. The pulse shape for LCP is denoted $p_{LCP}(t)$ and is the sum of N weighted and delayed pulses $p(t)$ as follows:

$$p_{LCP}(t) = \sum_{i=1}^{N} a_i p(t - \tau_i)$$

where $p(t)$ is any pulse that satisfies the cross-correlation constraints outlined in 14.4.5.

The number of pulses N that can be combined is set to four (although smaller number of pulses can be realized by setting the amplitudes of some of the pulses to zero). The values of the pulse delays shall be limited to $0 \le \tau_i \le 4$ ns. The value of τ_1 is assumed to be zero, and thus, the remaining delays are considered as relative delay time with respect to the nominal pulse location. The values for these delays are stored as the PIB values *phyUWBLCPDelay2*, *phyUWBLCPDelay3*, and *phyUWBLCPDelay4*. The values for the amplitudes a_i are stored as the PIB values *phyUWBLCPweight1* through *phyUWBLCPDelay4*, as defined in Table 71. The amplitudes, a_i, shall be selected so that the energy in the combined pulse is the same as the energy in the mandatory pulse. The numerical values of the delays and amplitudes of the pulses shall be transmitted following the general framework of optional pulse shapes, as defined in 14.5. The method to compute the weights and delay values is outside the scope of this standard.

14.6 Extended preamble for optional UWB CCA mode

The PHY may provide the capability to perform the optional UWB CCA mode 6, as defined in 8.2.7. This CCA mode shall be supported by the modified frame structure where preamble symbols are multiplexed with the data symbols in the PHR and the PSDU of a frame.

Figure 99 shows the modified frame structure with multiplexed preamble symbols. One preamble symbol is inserted after each PHR and PSDU segment. The inserted preamble symbol shall be the same as the symbol used in the SHR, as described in 14.2.5.1, of the same frame. The time interval between two neighboring inserted preamble symbols, which is also the time duration of each PHR or PSDU segment, is independent of the current data rate. The PIB attribute *phyUWBInsertedPreambleInterval* defines a constant time interval based on the data rate of 850 kb/s for all operation bands. The data rate of 850 kb/s is listed in Table 105 with the data rate field index R1 – R0 = 01. The value of the PIB attribute *phyUWBInsertedPreambleInterval* is fixed to four. Distinguished from the CCA in narrowband systems, which is used to detect the energy of carrier waveforms, the UWB CCA based on the frame with multiplexed preamble is used to detect the presence of preamble symbols. The processing gain can be enhanced by exploiting the spreading characteristics and repetition of the preamble symbols.

The PAN coordinator of a PAN shall coordinate all nodes in the PAN before the UWB CCA mode 6 is enabled. The modified frame structure with multiplexed preamble shall be applied to a data frame and a MAC frame in the CAP only when the PHY PIB attribute *phyCCAmode* indicates the UWB CCA mode 6.

The CCA detection time shall be equivalent to 40 data symbol periods, T_{dsym}, for a nominal 850 kb/s, or equivalently, at least 8 (multiplexed) preamble symbols should be captured in the CCA detection time.

In addition to enabling the UWB CCA mode 6, the multiplexed preamble symbols can help to improve ranging accuracy or assist data demodulation. This function is similar to that of the pilot tone in narrowband systems.

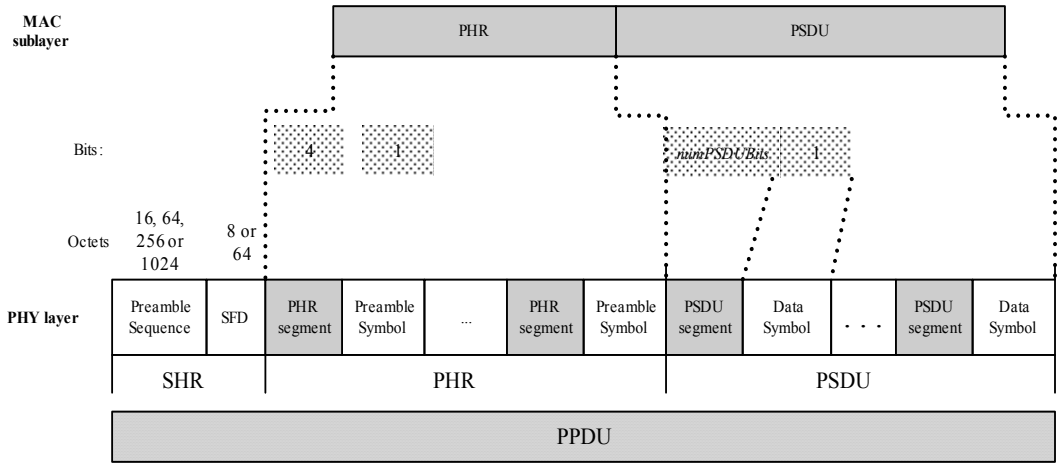


Figure 99—Illustration of the modified frame structure with multiplexed preamble

14.7 Ranging

Support for ranging is optional. A PHY that supports ranging is called a ranging-capable device (RDEV), and it has optional and mandatory capabilities. An RDEV shall support the ranging counter described in 14.7.1 and the FoM described in 14.7.3. An RDEV may support optional crystal characterization described in 14.7.2 and the optional DPS, as described in E.1.8.2.

RDEVs produce results, called timestamp reports, that are used by higher layers to compute the ranges between devices. An RDEV timestamp report shall consist of a 4-octet ranging counter start value, 4-octet ranging counter stop value, 4-octet ranging tracking interval, 3-octet ranging tracking offset, and 1-octet ranging FoM. These numbers are always reported together in the same primitive and remain together for their entire processing lifetime. It is not acceptable to have any pipelining of the individual results where (for example) in a timestamp report the ranging tracking offset and ranging tracking interval might be associated with the ranging counter value of the previous timestamp report and the ranging FoM might be associated with the ranging counter value of the timestamp report before that.

14.7.1 Ranging counter

The ranging counter supported by an RDEV is a set of behavioral properties and capabilities of the RDEV that produce ranging counter values. A ranging counter value is a 32-bit unsigned integer. The LSB of the counter value shall represent 1/128 of a chip time at the mandatory chipping rate of 499.2 MHz.

14.7.2 Crystal characterization

An RDEV that implements optional crystal characterization shall produce a tracking offset value and a tracking interval value for every timestamp report that is produced. The tracking offset and the tracking interval are computed from measurements taken during an interval that includes the interval bounded by the ranging counter start value and the ranging counter stop value. Note that crystal characterization is relevant only if it is characterizing the crystal that affects the ranging counter.

14.7.2.1 Ranging tracking offset

The UWB ranging tracking offset is a signed magnitude integer. The integer magnitude part of the number shall be 19 bits. The LSB of the integer represents a “part.” The sign bit of the signed magnitude integer shall be equal to zero when the oscillator at the transmitter is a higher frequency than the oscillator at the

receiver, and the sign bit shall be 1 when the oscillator at the receiver is a higher frequency than the transmitter. The value of the integer shall be a number that represents the difference in frequency between the receiver's oscillator and the transmitter's oscillator after the tracking offset integer is divided by the ranging tracking interval integer of 14.7.2.2. For example, if the difference between the oscillators is 10 ppm, then an acceptable value of the ranging tracking offset would be 10 when the ranging tracking interval is 1 million. Another acceptable value for the ranging tracking offset is 15 when the ranging tracking interval is 1.5 million.

14.7.2.2 Ranging tracking interval

The UWB ranging tracking interval shall be a 32-bit unsigned integer. The LSB of the ranging tracking interval represents a "part" that shall be exactly equal to the "part" in the LSB of the ranging tracking offset of 14.7.2.1. The size of the "part" is a time period that shall be smaller than or equal to a chip time at the mandatory chipping rate of 499.2 MHz. Use of smaller "parts" for the LSB is encouraged, as described in E.1.5.4.

14.7.3 Ranging FoM

An RDEV shall produce a ranging FoM for every ranging counter value that is produced. The UWB ranging FoM shall be formatted as shown in Figure 100. The FoM Confidence Level field is defined in Table 111. The confidence level is the probability that the leading edge of the pulse will arrive during the confidence interval. The FoM Confidence Interval field is defined in Table 112. The confidence interval width in Table 112 is the entire interval width, not a plus or minus number. The FoM Confidence Interval Scaling Factor field is defined in Table 113. Thus, the overall confidence interval is obtained according to the formula *overall confidence interval = confidence interval × confidence interval scaling field*. The MSB of the FoM octet is the extension bit. When the extension bit is set to zero, the fields have the normal meanings given in Table 111, Table 112, and Table 113. When the extension bit is 1, the FoM has the meaning given in Table 114.

Bit 7	6	5	4	3	2	1	0
Extension	Confidence Interval Scaling Factor field		Confidence Interval field		Confidence Level field		

Figure 100—Ranging FoM

Table 111—Confidence Level field

Confidence level	Bit 2	Bit 1	Bit 0
No FoM	0	0	0
20%	0	0	1
55%	0	1	0
75%	0	1	1
85%	1	0	0
92%	1	0	1
97%	1	1	0
99%	1	1	1

Table 112—Confidence Interval field

Confidence interval	Bit 1	Bit 0
100 ps	0	0
300 ps	0	1
1 ns	1	0
3 ns	1	1

Table 113—Confidence Interval Scaling Factor field

Confidence interval scaling factor	Bit 1	Bit 0
Confidence interval × 1/2	0	0
Confidence interval × 1	0	1
Confidence interval × 2	1	0
Confidence interval × 4	1	1

Table 114—FoM values with the extension bit set

	Bit 7	6	5	4	3	2	1	0
UWBRangingStart is uncorrected	1	0	0	0	0	0	0	0
Reserved	1	Any nonzero value						

The FoM characterizes the accuracy of the PHY estimate of the arrival time of the leading edge of the first pulse of the header at the antenna. The FoM in a particular timestamp report shall characterize the accuracy of the first pulse of the header that corresponds to the timer counter value in the same timestamp report.

The FoM value of 0x80 is specifically used to signal the upper layer that the RangingCounterStart value is not correct and the upper layer should use the sounding primitives. The FoM value of 0x00 is special and means “no FoM.” No FoM means that there simply is no information about the quality of a ranging measurement. That is different from reporting a very low quality measurement, but it is known that the measurement cannot be trusted. The FoM value 0x00 is not used to report untrustworthy measurements. The most untrustworthy measurement reportable is 0x79.

15. GFSK PHY

15.1 PPDU formats

The GFSK PHY shall use the PPDU formats described in 10.1, except that the preamble is 32 symbols (4 octets) and the bits in each octet shall be "01010101".

15.2 Modulation

The GFSK PHY does not employ any spreading technology.

15.2.1 GFSK PHY data rates

The data rate of the GFSK PHY shall be 100 kb/s.

15.2.2 Reference modulator diagram

The functional block diagram in Figure 101 is provided as a reference for specifying the GFSK PHY modulation and spreading functions. Each bit in the PPDU shall be processed through the data whitening and modulation functions in octet-wise order, beginning with the Preamble field and ending with the last octet of the PSDU. Within each octet, the LSB, b_0, is processed first and the MSB, b_7, is processed last.

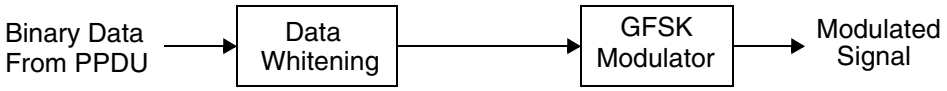


Figure 101—GFSK modulation and data whitening functions

15.2.3 Data whitening

Data whitening shall be the exclusive OR of the PPDU data (without SHR) with the PN9 sequence. This shall be performed by the transmitter and is given by:

$$E_n = R_n \oplus \mathrm{PN9}_n$$

where

R_n is the raw data bit being whitened
E_n is the whitened bit
$\mathrm{PN9}_n$ is the PN9 sequence

Index n starts after the SFD from 0 and is increased by one every symbol. For each packet transmitted, R_0 is the PHR first raw data bit after the SFD. Conversely, the decoding process, as performed at the receiver, can be described by:

$$R_n = RE_n \oplus \mathrm{PN9}_n$$

For each packet received, R_0 is the PHR first raw data bit.

The PN generator is defined by the schematic in Figure 102.

The seed in the PN9 shall be all ones: "1 1 1 1 1 1 1 1 1". The PN9 shall be reinitialized to the seed after each packet (either transmit or receive).

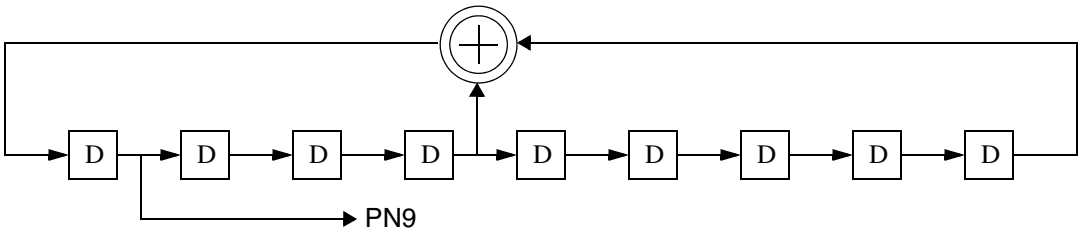


Figure 102—Schematic of the PN generator

The preamble and the SFD are not whitened. After the SFD, the PN9 generator is clocked starting from the seed. For example, the first 30 bits out of the PN9, once it is enabled, would be as follows:

$$PN9_n = 0_0\ 0_1\ 0_2\ 0_3\ 1_4\ 1_5\ 1_6\ 1_7\ 0_8\ 1_9\ 1_{10}\ 1_{11}\ 0_{12}\ 0_{13}\ 0_{14}\ 0_{15}\ 1_{16}\ 0_{17}\ 1_{18}\ 1_{19}\ 0_{20}\ 0_{21}\ 1_{22}\ 1_{23}\ 0_{24}$$

$$1_{25}\ 1_{26}\ 0_{27}\ 1_{28}\ 1_{29}$$

15.2.4 GFSK modulation

The bit sequences are modulated onto the carrier using GFSK with a modulation index of one where the Gaussian filter BT is 0.5, where a bit value of one is transmitted by shifting the frequency higher than the channel center frequency and a bit value of zero is transmitted by shifting the frequency lower than the current channel center frequency.

The nominal frequency deviation shall be 50 kHz. The deviation shall be between 70% and 130% of the nominal deviation. For the sequence 0101, the deviation shall be between 70% and 110% of the nominal deviation. for the sequence 00001111, the deviation shall be between 80% and 130% of the nominal deviation.

15.3 GFSK PHY RF requirements

15.3.1 Operating frequency range

The GFSK PHY operates in the 950.8–955.8 MHz frequency band.

15.3.2 Transmit PSD mask

The PSD mask for the GFSK PHY is specified as:

— The average power measured within ±100 kHz of the frequency 300 kHz apart from the center frequency shall be –26 dBm or less for a 1 mW device or –18 dBm or less for a 10 mW device.

— The average power measured with a 100 kHz resolution bandwidth in the frequency band from 950 MHz to 956 MHz except for the frequency band within ±300 kHz of the carrier frequency, f_c, shall be less than –39 dBm

NOTE—The PSD has to comply with Japanese regulations.

15.3.3 Symbol rate

The GFSK PHY symbol rate shall be 100 ksymbol/s with an accuracy of ±40 ppm.

15.3.4 Receiver sensitivity

Under the conditions specified in 8.1.7, a compliant GFSK PHY device shall be capable of achieving a sensitivity of –85 dBm or better.

15.3.5 Receiver interference rejection

The minimum receiver interference rejection levels are given in Table 115. The adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel, and the alternate channel is one more removed from the adjacent channel. For example, when channel 15 is the desired channel, channel 14 and channel 6 are the adjacent channels, and channel 13 and channel 17 are the alternate channels.

Table 115—Minimum receiver interference rejection requirements for GFSK PHY

Adjacent channel rejection	Alternate channel rejection
0 dB	24 dB

The adjacent channel rejection shall be measured as follows: the desired signal shall be a compliant GFSK PHY signal, as defined by 15.2, of pseudo-random data. The desired signal is input to the receiver at a level 3 dB greater than the maximum allowed receiver sensitivity given in 15.3.4.

In either the adjacent or the alternate channel, a compliant signal, as defined by 15.2, is input at the level specified in Table 115 relative to the desired signal. The test shall be performed for only one interfering signal at a time. The receiver shall meet the error rate criteria defined in 8.1.7 under these conditions.

15.3.6 TX-to-RX turnaround time

The GFSK PHY shall have a TX-to-RX turnaround time as defined in 8.2.1.

15.3.7 RX-to-TX turnaround time

The GFSK PHY shall have an RX-to-TX turnaround time as defined in 8.2.2.

15.3.8 Transmit center frequency tolerance

The GFSK PHY transmit center frequency tolerance shall be ±40 ppm maximum.

15.3.9 Transmit power

The GFSK PHY shall be capable of transmitting at a power level of least –3 dBm.

15.3.10 Receiver maximum input level of desired signal

The GFSK PHY shall have a receiver maximum input level greater than or equal to –20 dBm using the measurement defined in 8.2.4.

15.3.11 Receiver ED

The GFSK PHY shall provide the receiver ED measurement as described in 8.2.5.

15.3.12 Link quality indicator (LQI)

The GFSK PHY shall provide the LQI measurement as described in 8.2.6.

15.3.13 Clear channel assessment (CCA)

The GFSK PHY shall use the one of the CCA modes as described in 8.2.7.

Annex A

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry—Key Agreement and Key Transport Using Elliptic Curve Cryptography.[7]

[B2] ARIB STD-T96, 950 MHz-Band Telemeter, Telecontrol and Data Transmission Radio Equipment for Specified Low Power Radio Station, 2010.7.15 (H22.7.15) Version 1.1.

[B3] Biswas, P,. and Ye, Y., "Semidefinite programming for ad hoc wireless sensor network localization," *IEEE Conference on Information Processing in Sensor Networks*, pp. 46–54, Apr. 2004.

[B4] Coexistence analysis of IEEE Std 802.15.4 with other IEEE standards and proposed standards, Doc. IEEE 15-10-0808-00-0000, Sept. 2010.

[B5] Doherty, L., Pister, K. S. J., and. El Ghaoui, L., "Convex position estimation in wireless sensor networks," *IEEE Conference on Information Theory and Communications*, pp. 1655–1663, Apr. 2001.

[B6] English translation of ARIB STD-T96, 950 MHz-Band Telemeter, Telecontrol and Data Transmission Radio Equipment for Specified Low Power Radio Station, 2008.6.6 (H20.6.6) Version 1.0.

[B7] Gentile, C., "Distributed sensor location through linear programming with triangle inequality constraints," *IEEE Conference on Communications*, June 2006.

[B8] Gentile, C., "Sensor location through through linear programming with triangle inequality constraints," *IEEE Conference on Communications*, pp. 3192–3196, May 2005.

[B9] Guvenc, I., and Sahinoglu, Z., "Threshold-based TOA estimation for impulse radio UWB systems," *IEEE International Conference on Ultra Wideband Systems and Technologies*, pp. 420–425, Sept. 2005.

[B10] Guvenc, I., and Sahinoglu, Z., "Threshold selection for UWB TOA estimation based on Kurtosis analysis," *IEEE Communication Letters,* vol. 9, no. 12, pp. 1025–1027, Dec. 2005.

[B11] Hach, R., "Symmetric double sided two-way ranging," IEEE P802.15 Working Group for Wireless Personal Area Networks (WPAN), Doc. IEEE P.802.15-05-0334-00-004a, June 2005.Jonsson, J., "On the security of CTR + CBC-MAC, in Proceedings of Selected Areas in Cryptography—SAC 2002," Nyberg, K., Heys, H., eds., lecture notes in *Computer Science*, vol. 2595, pp. 76–93, Berlin: Springer, 2002.

[B12] Jonsson, J., "On the security of CTR + CBC-MAC, NIST mode of operation," additional CCM documentation.[8]

[7]ANSI publications are available from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[8]Publication is available at http://csrc.nist.gov/encryption/modes/proposedmodes/.

[B13] Lee, J.-Y., and Scholtz, R. A., "Ranging in sense multipath environment using an UWB radio link," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 9, Dec. 2002.

[B14] Niculescu, D., and Nath, B., "Ad hoc positioning system (APS)," *IEEE Conference on Global Communications*, pp. 2926–2931, Nov. 2001.

[B15] NIST Pub 800-38C, Recommendation for Block Cipher Modes of Operation—The CCM Mode for Authentication and Confidentiality, U.S. Department of Commerce/N.I.S.T., May 12, 2004.

[B16] Qi, Y., Kobayashi, H., and Suda, H., "On time-of-arrival positioning in a multipath environment," *IEEE Transactions on Vehicular Technology*, 2006.

[B17] Qi, Y., and Kohno, R., "Mitigation of sampling-induced errors in delay estimation," *Proceedings of the IEEE International Conference on UWB 2005 (ICU2005)*, Zurich, Switzerland, Sept. 2005.

[B18] Rogaway, P., and Wagner, D., "A critique of CCM," IACR ePrint Archive 2003-070, Apr. 13, 2003.[9]

[B19] Saverese, C., Rabaey, J. M., and Beutel, J., "Location in distributed ad-hoc networks," *IEEE Conference on Acoustics, Speech, and Signal Processing*, pp. 2037–2040, May 2001.

[B20] Savvides, A., Park, H., and Srivastava, M. B., "The bits and flops of the N-hop multilateration primitive for node localization problems," *ACM Conference on Wireless Sensor Networks and Applications*, pp. 112–121, Sept. 2002.

[B21] Shang, Y., Rumi, W,. Zhang, Y., and Fromherz, M. P. J., "Localization from mere connectivity," *ACM Conference on Mobile Ad Hoc Networking and Computing*, pp. 201–212, June 2003.

[9]Publication is available from http://www.iacr.org/.

Annex B

(normative)

CCM* mode of operation

B.1 Introduction

CCM* is a generic combined encryption and authentication block cipher mode. The CCM* mode coincides with the original specification for the combined counter with CBC-MAC (cipher block chaining message authentication code) mode of operation (known as CCM) (ANSI X9.63-2001 [B1], Appendix A of NIST Pub 800-38C [B15]) for messages that require authentication and, possibly, encryption, but also offers support for messages that require only encryption. Moreover, it can be used in implementation environments for which the use of variable-length authentication tags, rather than fixed-length authentication tags only, is beneficial.

B.2 Notation and representation

B.2.1 Strings and string operations

A string is a sequence of symbols over a specific set (e.g., the binary alphabet {0,1} or the set of all octets). The length of a string is the number of symbols it contains (over the same alphabet). The empty string is the string of length 0. The right-concatenation of two strings x and y (over the same alphabet) of length m and n, respectively (notation: $x \parallel y$), is the string z of length $m + n$ that coincides with x on its leftmost (most significant) m symbols and with y on its rightmost (least significant) n symbols. An octet is a symbol string of length 8. In the context of this annex, all octets are strings over the binary alphabet.

B.2.2 Integers, octets, and their representation

Throughout this annex, the representation of integers as octet strings and of octet strings as binary strings shall be fixed. All integers shall be represented as octet strings in most-significant-octet-first order. All octets shall be represented as bit strings of length eight in most-significant-bit-first order.

For example, the 32-bit integer 0x12345678 is represented as an octet string of {0x12, 0x34, 0x56, 0x78}, and the first octet of that octet string is represented as a bit string of {0,0,0,1,0,0,1,0}.

B.3 Symmetric-key cryptographic building blocks

The symmetric-key cryptographic primitives and mechanisms are defined for use with all security processing operations specified in this standard.

B.3.1 Block cipher

The block cipher used in this standard shall be the advanced encryption standard (AES)-128, as specified in FIPS Pub 197.[10] This block cipher shall be used with symmetric keys with the same size as that of the block cipher: 128 bits. The generation of these keys is outside the scope of this standard.

B.3.2 Mode of operation

The block cipher mode of operation used in this standard shall be the generic CCM* mode of operation, as specified in B.4, with the following instantiations:

a) Each entity shall use the block cipher *E* as specified in B.3.1.
b) All integers shall be represented as octet strings as specified in B.2.2.
c) All octets shall be represented as binary strings as specified in B.2.2.
d) The parameter *L* shall have the integer value 2.
e) The parameter *M* shall have one of the following integer values: 0, 4, 8, or 16.

B.4 Specification of generic CCM* mode of operation

Prerequisites:

The following are the prerequisites for the operation of the generic CCM* mode:

a) A block cipher encryption function *E* shall have been chosen, with a 128-bit block size. The length in bits of the keys used by the chosen encryption function is denoted by keylen.
b) A fixed representation of integers as octet strings shall have been chosen (e.g., most-significant-octet-first order or least-significant-octet-first order).
c) A fixed representation of octets as binary strings shall have been chosen (e.g., most-significant-bit-first order or least-significant-bit-first order).
d) The length *L* of the message Length field, in octets, shall have been chosen. Valid values for *L* are the integers 2, 3, …, 8 (the value $L = 1$ is reserved).
e) The length *M* of the Authentication field, in octets, shall have been chosen. Valid values for *M* are the integers 0, 4, 6, 8, 10, 12, 14, and 16 (the value $M = 0$ corresponds to disabling authenticity because then the Authentication field is the empty string).

B.4.1 CCM* mode encryption and authentication transformation

Inputs:

The CCM* mode forward transformation takes the following as inputs:

a) A bit string Key of length keylen bits to be used as the key. Each entity shall have evidence that access to this key is restricted to the entity itself and its intended key sharing group member(s).
b) A nonce *N* of $15 - L$ octets. Within the scope of any encryption key Key, the nonce value shall be unique.
c) An octet string *m* of length $l(m)$ octets, where $0 \leq l(m) < 2^{8L}$.

[10]Information on references can be found in Clause 2.

d) An octet string a of length $l(a)$ octets, where $0 \le l(a) < 2^{64}$.

The nonce N shall encode the potential values for M so that the actual value of M can be uniquely determined from N.

Actions:

The CCM* mode forward transformation involves the execution, in order, of an input transformation, as defined in B.4.1.1, an authentication transformation, as defined in B.4.1.2, and an encryption transformation, as defined in B.4.1.3.

B.4.1.1 Input transformation

This step involves the transformation of the input strings a and m to the strings AuthData and PlainTextData, to be used by the authentication transformation and the encryption transformation, respectively.

This step involves the following steps, in order:

a) Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string a, as follows:
   1) If $l(a) = 0$, then $L(a)$ is the empty string.
   2) If $0 < l(a) < 2^{16} - 2^8$, then $L(a)$ is the 2-octets encoding of $l(a)$.
   3) If $2^{16} - 2^8 \le l(a) < 2^{32}$, then $L(a)$ is the right-concatenation of the octet 0xff, the octet 0xfe, and the 4-octet encoding of $l(a)$.
   4) If $2^{32} \le l(a) < 2^{64}$, then $L(a)$ is the right-concatenation of the octet 0xff, the octet 0xff, and the 8-octet encoding of $l(a)$.

b) Right-concatenate the octet string $L(a)$ with the octet string a itself. Note that the resulting string contains $l(a)$ and a encoded in a reversible manner.

c) Form the padded message AddAuthData by right-concatenating the resulting string with the smallest non-negative number of all-zero octets so that the octet string AddAuthData has length divisible by 16.

d) Form the padded message PlaintextData by right-concatenating the octet string m with the smallest non-negative number of all-zero octets so that the octet string PlaintextData has length divisible by 16.

e) Form the message AuthData consisting of the octet strings AddAuthData and PlaintextData: AuthData = AddAuthData || PlaintextData.

B.4.1.2 Authentication transformation

The data AuthData that was established in B.4.1.1 shall be tagged using the tagging transformation as follows:

a) Form the 1-octet Flags field consisting of the 1-bit Reserved field, the 1-bit Adata field, and particular 3-bit representations of the integers M and L, as follows:

   Flags = Reserved || Adata || M || L.

   Here, the 1-bit Reserved field is reserved for future expansions and shall be set to '0'. The 1-bit Adata field is set to '0' if $l(a) = 0$ and set to '1' if $l(a) > 0$. The M field is the 3-bit representation of the integer $(M - 2)/2$ if $M > 0$ and of the integer 0 if $M = 0$, in most-significant-bit-first order. The L field is the 3-bit representation of the integer $L - 1$, in most-significant-bit-first order.

b) Form the 16-octet B_0 field consisting of the 1-octet Flags field defined in step a) in this subclause, the $(15 - L)$-octet Nonce field N, and the L-octet representation of the Length field $l(m)$, as follows:

   B_0 = Flags || Nonce N || $l(m)$.

c) Parse the message AuthData as $B_1 \| B_2 \| \ldots \| B_t$, where each message block B_i is a 16-octet string.

d) The CBC-MAC value X_{t+1} is defined by:

$X_0 := 0^{128}$; $X_{i+1} := E(\text{Key}, X_i \oplus B_i)$ for i = 0, …, t.

Here, E(K, x) is the cipher text that results from encryption of the plaintext x, using the established block cipher encryption function E with key K; the string 0^{128} is the 16-octet all-zero bit string.

e) The authentication tag T is the result of omitting all but the leftmost M octets of the CBC-MAC value X_{t+1} thus computed.

B.4.1.3 Encryption transformation

The data PlaintextData that was established in B.4.1.1 (step d) and the authentication tag T that was established in B.4.1.2 (step e) shall be encrypted using the encryption transformation as follows:

a) Form the 1-octet Flags field consisting of two 1-bit Reserved fields, and particular 3-bit representations of the integers 0 and L, as follows:

Flags = Reserved || Reserved || 0 || L.

Here, the two 1-bit Reserved fields are reserved for future expansions and shall be set to '0'. The '0' field is the 3-bit representation of the integer 0, in most-significant-bit-first order. The L field is the 3-bit representation of the integer L – 1, in most-significant-bit-first order.

b) Define the 16-octet A_i field consisting of the 1-octet Flags field defined in step a) in this subclause, the (15 – L)-octet Nonce field N, and the L-octet representation of the integer i, as follows:

A_i = Flags || Nonce N || Counter i, for i = 0, 1, 2, …

Note that this definition ensures that all the A_i fields are distinct from the B_0 fields that are actually used, as those have a Flags field with a nonzero encoding of M in the positions where all A_i fields have an all-zero encoding of the integer 0, as described in B.4.1.2, step b).

c) Parse the message PlaintextData as M1 || … $\| M_t$, where each message block M_i is a 16-octet string.

d) The cipher text blocks $C_1, \ldots, C_t$ are defined by:

$C_i := E(\text{Key}, A_i) \oplus M_i$ for i = 1, 2, …, t.

e) The string Ciphertext is the result of omitting all but the leftmost $l(m)$ octets of the string $C_1 \| \ldots \| C_t$.

f) Define the 16-octet encryption block S_0 by:

$S_0 := E(\text{Key}, A_0)$.

g) The encrypted authentication tag U is the result of XOR-ing the string consisting of the leftmost M octets of S_0 and the authentication tag T.

Output:

If any of the preceding operations has failed, then output “invalid.” Otherwise, output the right-concatenation c of the encrypted message Ciphertext and the encrypted authentication tag U.

B.4.2 CCM* mode decryption and authentication checking transformation

Inputs:

The CCM* mode inverse transformation takes the following as inputs:

a) A bit string Key of length keylen bits to be used as the key. Each entity shall have evidence that access to this key is restricted to the entity itself and its intended key-sharing group member(s).

b) A nonce N of $15 - L$ octets. Within the scope of any encryption key Key, the nonce value shall be unique.

c) An octet string c of length $l(c)$ octets, where $0 \le l(c) - M < 2^{8L}$.

d) An octet string a of length $l(a)$ octets, where $0 \le l(a) < 2^{64}$.

Actions:

The CCM* mode inverse transformation involves the execution, in order, of a decryption transformation, as defined in B.4.2.1, and an authentication checking transformation, as defined in B.4.2.2.

B.4.2.1 Decryption transformation

The decryption transformation involves the following steps, in order:

a) Parse the message c as $C \| U$, where the rightmost string U is an M-octet string. If this operation fails, output "invalid" and stop. U is the purported encrypted authentication tag. Note that the leftmost string C has length $l(c) - M$ octets.

b) Form the padded message CiphertextData by right-concatenating the string C with the smallest non-negative number of all-zero octets so that the octet string CiphertextData has length divisible by 16.

c) Use the encryption transformation in B.4.1.3, with as inputs the data CipherTextData and the tag U.

d) Parse the output string resulting from applying this transformation as $m \| T$, where the rightmost string T is an M-octet string. T is the purported authentication tag. Note that the leftmost string m has length $l(c) - M$ octets.

B.4.2.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

a) Form the message AuthData using the input transformation in B.4.1.1, with as inputs the string a and the octet string m that was established in B.4.2.1 (step d).

b) Use the authentication transformation in B.4.1.2, with as input the message AuthData.

c) Compare the output tag MACTag resulting from this transformation with the tag T that was established in B.4.2.1 (step d). If MACTag = T, output "valid"; otherwise, output "invalid" and stop.

Output:

If any of the preceding verifications has failed, then output "invalid," and reject the octet strings a and m. Otherwise, accept the octet strings a and m, and accept one of the key sharing group member(s) as the source of a and m.

B.4.3 Restrictions

All implementations shall limit the total amount of data that is encrypted with a single key. The CCM* encryption and authentication transformation shall invoke not more than 2^{61} block cipher encryption function invocations with the same key in total.

The CCM* decryption and authentication checking transformation shall not expose any information if any verification check fails. The only information that may be exposed in this case is that the authenticity verification transformation failed; all other information, such as the purported plaintext, shall be destroyed.

NOTE 1—With regard to security of the CCM* mode of operation, the CCM* mode coincides with the original CCM mode specification (ANSI X9.63-2001 [B1]) for messages that require authentication and, possibly, encryption, but also

offers support for messages that require only encryption. Moreover, it can be used in implementation environments for which the use of variable-length authentication tags, rather than fixed-length authentication tags only, is beneficial. As with the CCM mode, the CCM* mode requires only one key. The CCM* specification differs from the CCM specification, as follows:

— The CCM* mode allows the length of the Authentication field M to be zero as well (the value $M = 0$ corresponding to disabling authenticity because then the Authentication field is the empty string).
— The CCM* mode imposes a further restriction on the nonce N: it shall encode the potential values for M so that one can uniquely determine from N the actually used value of M.

As a result, if M is fixed and the value $M = 0$ is not allowed, then there are no additional restrictions on N, in which case the CCM* mode reduces to the CCM mode. In particular, the proof of the CCM mode applies (Jonsson [B11] and [B12]).

For fixed-length authentication tags, the CCM* mode is equally secure as the original CCM mode. For variable-length authentication tags, the CCM* mode completely avoids, by design, the vulnerabilities that do apply to the original CCM mode.

For fixed-length authentication tags, the security proof of the original CCM mode carries over to that of the CCM* mode (also for $M = 0$), by observing that the proof of the original CCM mode relies on the following properties, which slightly relax those stated in Jonsson [B11] and [B12] (relaxed property indicated in italics):

— The B_0 field uniquely determines the value of the nonce N.
— The authentication transformation operates on input strings $B_0 \| B_1 \| B_2 \| \ldots \| B_t$ from which one can uniquely determine the input strings a and m (as well as the nonce N). In fact, for any two input strings corresponding to distinct triples (N, m, a), neither one is a prefix string of the other.
— All the A_i fields are distinct from the B_0 fields *that are actually used* (over the lifetime of the key), as those have a Flags field with a nonzero encoding of M in the positions where all A_i fields have an all-zero encoding of the integer 0.

Hence, if M is fixed, then the CCM* mode offers the same security properties as the original CCM mode: confidentiality over the input string m and data authenticity over the input strings a and m, relative to the length of the authentication tag. Obviously, if $M = 0$, then no data authenticity is provided by the CCM* mode itself (but may be provided by an external mechanism).

For variable-length authentication tags, the original CCM mode is known to be vulnerable to specific attacks (e.g., Section 3.4 of Rogaway and Wagner [B18]). These attacks may arise with the original CCM mode because the decryption transformation does not depend on the length of the authentication tag itself. The CCM* mode avoids these attacks altogether by requiring that one shall be able to uniquely determine the length of the applicable authentication tag from the A_i fields (i.e., from the counters blocks).

NOTE 2—With regard to the interoperability between CCM mode and CCM* mode of operation, the CCM* mode reduces to the CCM mode in all implementation environments where the length of the authentication tag is fixed and where the value $M = 0$ (encryption-only) is not allowed. In particular, the CCM* mode is compatible with the CCM mode, as specified in IEEE Std 802.11™-2007 (for WLANs), IEEE Std 802.15.3™-2003 (for WPANs), and IEEE Std 802.15.4-2003 (for older WPANs).

NOTE 3—Test vectors for cryptographic building blocks are given in Annex C.

Annex C

(informative)

Test vectors for cryptographic building blocks

With regard to the CCM* mode of operation, as described in Annex B, this annex provides sample test vectors for the IEEE 802.15.4 community, aimed at assisting in building interoperable security implementations.

C.1 AES block cipher

FIPS Pub 197 provides sample test vectors for the block cipher specified in B.3.1.

C.2 Mode of operation

This subclause provides sample test vectors for the mode of operation as specified in B.3.2, illustrated in the context of different MAC frame types.

C.2.1 MAC beacon frame

C.2.1.1 Description

The example below illustrates security processing of a beacon frame that is transmitted by the coordinator using its extended source address. In this example, the Superframe Specification field is set to 0xCF55 (e.g., the beacon order and superframe order have integer value 5, while the final CAP slot has integer value 15), and there are no pending addresses. This example uses source address 0xacde480000000001, PAN identifier 0x4321, and beacon payload 0x51 0x52 0x53 0x54; the frame counter has integer value 5. The security level is set to 0x02 (MIC-64, or 64-bit data authenticity).

For simplicity, all frames in this example are shown without the FCS field (because security processing is independent of it).

Secured beacon frame:

08 D0 84 21 43 01 00 00 00 00 48 DE AC || 02 05 00 00 00 || 55 CF 00 00 51 52 53 54 22 3B C1 EC 84 1A B5 53.

Corresponding unsecured beacon frame:

00 C0 84 21 43 01 00 00 00 00 48 DE AC || 55 CF 00 00 51 52 53 54.

Prerequisite:

For this mode of operation, the parameter *M* has the integer value eight.

C.2.1.2 CCM* mode encryption and authentication transformation

Inputs:

The inputs to the CCM* mode forward transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:
   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.
b) The nonce N of $15 – L = 13$ octets to be used:
   Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 02.
c) The octet string m of length $l(m) = 0$ octets to be used:
   m = (empty string).
d) The octet string a of length $l(a) = 26$ octets to be used:
   a = 08 D0 84 21 43 01 00 00 00 00 48 DE AC || 02 || 05 00 00 00 || 55 CF 00 00 51 52 53 54.

Actions:

The CCM* mode forward transformation involves the execution, in order, of an input transformation, as defined in C.2.1.2.1, an authentication transformation, as defined in C.2.1.2.2, and an encryption transformation, as defined in C.2.1.2.3.

C.2.1.2.1 Input transformation

This step involves the transformation of the input strings a and m to the strings AuthData and PlainTextData, to be used by the authentication transformation and the encryption transformation, respectively.

a) Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string a:
   $L(a)$ = 00 1A.
b) Right-concatenate the octet string $L(a)$ with the octet string a itself:
   $L(a)$ || a =00 1A || 08 D0 84 21 43 01 00 00 00 00 48 DE AC 02 05 00 00 00 55 CF 00 00 51 52 53 54.
c) Form the padded message AddAuthData by right-concatenating the resulting string with the smallest non-negative number of all-zero octets so that the octet string AddAuthData has length divisible by 16:
   AddAuthData =00 1A 08 D0 84 21 43 01 00 00 00 00 48 DE AC 02 05 00 00 00 55 CF 00 00 51 52 53 54 00 00 00 00.
d) Form the padded message PlaintextData by right-concatenating the octet string m with the smallest non-negative number of all-zero octets so that the octet string PlaintextData has length divisible by 16:
   PlaintextData = (empty string).
e) Form the message AuthData consisting of the octet strings AddAuthData and PlaintextData:
   AuthData =00 1A 08 D0 84 21 43 01 00 00 00 00 48 DE AC 02 05 00 00 00 55 CF 00 00 51 52 53 54 00 00 00 00.

C.2.1.2.2 Authentication transformation

The data AuthData that was established in C.2.1.2.1 is tagged using the tagging transformation as follows:

a) Form the 1-octet Flags field:
   Flags = 59.
b) Form the 16-octet B_0 field:
   B_0 = 59 || AC DE 48 00 00 00 00 01 00 00 00 05 02 || 00 00.
c) Parse the message AuthData as B_1 || B_2, where each message block B_i is a 16-octet string.

d) The CBC-MAC value X_3 is computed as shown in Table C.1.

Table C.1—Computation of CBC-MAC value

i	B_i	X_i
0	59 AC DE 48 00 00 00 00 01 00 00 00 05 02 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1	00 1A 08 D0 84 21 43 01 00 00 00 00 48 DE AC 02	C4 A4 D0 BD 70 73 7E 32 11 2E 51 9A CA A2 01 F1
2	05 00 00 00 55 CF 00 00 51 52 53 54 00 00 00 00	A9 70 2C 6E E1 7E DE E0 C7 32 88 0A 40 41 7F 9C
3	—	AB 6B 19 E7 5B 75 2D 9A 6E F0 CC 13 09 98 EB D0

e) The authentication tag *T* is the result of omitting all but the leftmost $M = 8$ octets of the CBC-MAC value X_3:

T = AB 6B 19 E7 5B 75 2D 9A.

C.2.1.2.3 Encryption transformation

The data PlaintextData that was established in C.2.1.2.1 (step d) and the authentication tag *T* that was established in C.2.1.2.2 (step e) are encrypted using the encryption transformation as follows:

a) Form the 1-octet Flags field:

Flags = 01.

b) Define the 16-octet A_i field as shown in Table C.2.

Table C.2—A_i fields

i	A_i
0	01 AC DE 48 00 00 00 00 01 00 00 00 05 02 00 00

c) Define the 16-octet encryption block S_0:

S_0:= *E*(Key, A_0) = 89 50 D8 0B DF 6F 98 C9 63 F2 D5 A1 08 A1 55 C7.

d) The encrypted authentication tag *U* is the result of XOR-ing the string consisting of the leftmost $M = 8$ octets of S_0 and the authentication tag *T*:

U = 22 3B C1 EC 84 1A B5 53.

Output:

The octet string *c* = 22 3B C1 EC 84 1A B5 53.

C.2.1.3 CCM* mode decryption and authentication checking transformation

Inputs:

The inputs to the CCM* mode inverse transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:

Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

b) The nonce *N* of 15 – *L* = 13 octets to be used:

Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 02.

c) The octet string *c* of length *l*(*c*) = 8 octets to be used:

c = 22 3B C1 EC 84 1A B5 53.

d) The octet string *a* of length *l*(*a*) = 26 octets to be used:

a = 08 D0 84 21 43 01 00 00 00 00 48 DE AC || 02 || 05 00 00 00 || 55 CF 00 00 51 52 53 54.

Actions:

The CCM* mode inverse transformation involves the execution, in order, of a decryption transformation, as defined in C.2.1.3.1, and an authentication checking transformation, as defined in C.2.1.3.2.

C.2.1.3.1 Decryption transformation

The decryption transformation involves the following steps, in order:

a) Parse the message *c* as *C* ||*U*, where the rightmost string *U* is an 8-octet string:

C = (empty string);
U = 22 3B C1 EC 84 1A B5 53.

b) Form the 1-octet Flags field:

Flags = 01.

c) Define the 16-octet A_i field as shown in Table C.3.

Table C.3—A_i fields

i	A_i
0	01 AC DE 48 00 00 00 00 01 00 00 00 05 02 00 00

d) Define the 16-octet encryption block S_0:

S_0 = *E*(Key, A_0) = 89 50 D8 0B DF 6F 98 C9 63 F2 D5 A1 08 A1 55 C7.

e) The purported authentication tag *T* is the result of XOR-ing the string consisting of the leftmost *M* = 8 octets of S_0 and the octet string *U*:

T = AB 6B 19 E7 5B 75 2D 9A.

C.2.1.3.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

a) Form the message AuthData using the input transformation in C.2.1.2.1, with as inputs the string *a* and the octet string *m* = (empty string):

AuthData =00 1A 08 D0 84 21 43 01 00 00 00 00 48 DE AC 02 05 00 00 00 55 CF
00 00 51 52 53 54 00 00 00 00.

b) Use the authentication transformation in C.2.1.2.2, with as input the message AuthData to compute the authentication tag MACTag:

MACTag = AB 6B 19 E7 5B 75 2D 9A.

c) Compare the output tag MACTag resulting from this transformation with the tag *T* that was established in C.2.1.3.1 (step e):

T = AB 6B 19 E7 5B 75 2D 9A = MACTag.

Output:

Because MACTag = *T*, output "valid," accept the octet strings *a* and *m*, and accept one of the key sharing group member(s) as the source of *a* and *m*.

C.2.2 MAC data frame

C.2.2.1 Description

The example below illustrates security processing of a data frame that is transmitted using extended addresses, with PAN identifier compression and acknowledgment enabled. This example uses source address 0xacde480000000001, destination address 0xacde480000000002, PAN identifier 0x4321, and data payload 0x61 0x62 0x63 0x64; the frame counter has integer value five. The security level is set to 0x04 (ENC, or data confidentiality without data authenticity).

For simplicity, all frames in this example are shown without the FCS field (because security processing is independent of it).

Secured data frame:

69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC || 04 05 00 00 00 || D4 3E 02 2B.

Corresponding unsecured data frame:

61 CC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC || 61 62 63 64.

Prerequisite:

For this mode of operation the parameter *M* has the integer value zero.

C.2.2.2 CCM* mode encryption and authentication transformation

Inputs:

The inputs to the CCM* mode forward transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:

Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

b) The nonce *N* of 15 – *L* = 13 octets to be used:

Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 04.

c) The octet string *m* of length *l*(*m*) = 4 octets to be used:

m = 61 62 63 64.

d) The octet string *a* of length *l*(*a*) = 26 octets to be used:

a = 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC || 04 || 05 00 00 00.

Actions:

The CCM* mode forward transformation involves the execution, in order, of an input transformation, as defined in C.2.2.2.1, an authentication transformation, as defined in C.2.2.2.2, and an encryption transformation, as defined in C.2.2.2.3.

C.2.2.2.1 Input transformation

This step involves the transformation of the input strings *a* and *m* to the strings AuthData and PlainTextData, to be used by the authentication transformation and the encryption transformation, respectively.

a) Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string *a*:

   $L(a)$ = 00 1A.

b) Right-concatenate the octet string $L(a)$ and the octet string *a* itself:

   $L(a) \| a$ = 00 1A || 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC 04 05 00 00 00.

c) Form the padded message AddAuthData by right-concatenating the resulting string with the smallest non-negative number of all-zero octets so that the octet string AddAuthData has length divisible by 16:

   AddAuthData = 00 1A 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC 04 05 00 00 00 00 00 00 00.

d) Form the padded message PlaintextData by right-concatenating the octet string *m* with the smallest non-negative number of all-zero octets so that the octet string PlaintextData has length divisible by 16:

   PlaintextData = 61 62 63 64 00 00 00 00 00 00 00 00 00 00 00 00.

e) Form the message AuthData consisting of the octet strings AddAuthData and PlaintextData:

   AuthData = 00 1A 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC 04 05 00 00 00 00 00 00 00 61 62 63 64 00 00 00 00 00 00 00 00 00 00 00 00.

C.2.2.2.2 Authentication transformation

T = (empty string).

C.2.2.2.3 Encryption transformation

The data PlaintextData is encrypted using the encryption transformation as follows:

a) Form the 1-octet Flags field:

   Flags = 01.

b) Define the 16-octet A_i field as shown in Table C.4.

Table C.4—A_i fields

i	A_i
1	01 AC DE 48 00 00 00 00 01 00 00 00 05 04 00 01

c) Parse the message PlaintextData as M_1, where each message block M_i is a 16-octet string.

d) The ciphertext block C_1 is computed as shown in Table C.5.

Table C.5—Computation of ciphertext

i	AES(Key, A_i)	C_i = AES(Key, A_i) ⊕ M_i
1	B5 5C 61 4F A6 8B 7E E0 CB 77 37 EB A8 1D 33 41	D4 3E 02 2B A6 8B 7E E0 CB 77 37 EB A8 1D 33 41

e) The string Ciphertext is the result of omitting all but the leftmost $l(m) = 4$ octets of the string C_1:
CipherText = D4 3E 02 2B.

Output:

c = D4 3E 02 2B.

C.2.2.3 CCM* mode decryption and authentication checking transformation

Inputs:

The inputs to the CCM* mode inverse transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:
Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

b) The nonce N of $15 – L = 13$ octets to be used:
Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 04.

c) The octet string c of length $l(c) = 4$ octets to be used:
c = D4 3E 02 2B.

d) The octet string a of length $l(a) = 26$ octets to be used:
a = 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC || 04 || 05 00 00 00.

Actions:

The CCM* mode inverse transformation involves the execution, in order, of a decryption transformation, as defined in C.2.2.3.1, and an authentication checking transformation, as defined in C.2.2.3.2.

C.2.2.3.1 Decryption transformation

The decryption transformation involves the following steps, in order:

a) Parse the message c as $C \| U$, where the rightmost string U is a 0-octet string:
C = D4 3E 02 2B.
U = (empty string).

b) Form the padded message CiphertextData by right-concatenating the string C with the smallest non-negative number of all-zero octets so that the octet string CiphertextData has length divisible by 16:
CipherTextData = D4 3E 02 2B 00 00 00 00 00 00 00 00 00 00 00 00.

c) Form the 1-octet Flags field:
Flags = 01.

d) Define the 16-octet A_i field as shown in Table C.6.

Table C.6—A_i fields

i	A_i
1	01 AC DE 48 00 00 00 00 01 00 00 00 05 04 00 01

e) Parse the message CiphertextData as C_1, where each message block C_i is a 16-octet string.

f) The plaintext block P_1 is computed as shown in Table C.7.

Table C.7—Computation of plaintext

i	AES(Key, A_i)	P_i = AES(Key, A_i) ⊕ C_i
1	B5 5C 61 4F A6 8B 7E E0 CB 77 37 EB A8 1D 33 41	61 62 63 64 00 00 00 00 00 00 00 00 00 00 00 00

g) The octet string *m* is the result of omitting all but the leftmost $l(m) = 4$ octets of the string P_1:

m = 61 62 63 64.

h) The purported authentication tag *T* is the empty string:

T = (empty string).

C.2.2.3.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

a) Form the message AuthData using the input transformation in C.2.2.2.1, with as inputs the string *a* and the octet string *m* that was established in C.2.2.3.1 (step g):

AuthData = 00 1A 69 DC 84 21 43 02 00 00 00 00 48 DE AC 01 00 00 00 00 48 DE AC 04 05 00 00 00 00 00 00 00 61 62 63 64 00 00 00 00 00 00 00 00 00 00 00 00.

b) Use the authentication transformation in C.2.2.2.2, with as input the message AuthData to compute the authentication tag MACTag:

MACTag = (empty string).

c) Compare the output tag MACTag resulting from this transformation with the tag *T* that was established in C.2.2.3.1 (step h):

T = (empty string) = MACTag.

Output:

Because MACTag = *T*, output "valid," accept the octet strings *a* and *m*, and accept one of the key sharing group member(s) as the source of *a* and *m*.

C.2.3 MAC command frame

C.2.3.1 Description

The example below illustrates security processing of an association request command frame that is transmitted by a FFD using extended addresses, with acknowledgment enabled. In this example, the Capability field is set to 0xCE. This example uses source address 0xacde480000000001, destination address 0xacde480000000002, PAN identifier 0x4321, and command payload 0xCE; the frame counter has integer value five. The security level is set to 0x06 (ENC-MIC-64, or data confidentiality with 64-bit data authenticity).

For simplicity, all frames in this example are shown without the FCS field (because security processing is independent of it).

Secured command frame:

2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF 01 00 00 00 00 48 DE AC || 06 05 00 00 00 || 01 D8 4F DE 52 90 61 F9 C6 F1.

Corresponding unsecured command frame:

23 CC 84 21 43 02 00 00 00 00 48 DE AC FF FF 01 00 00 00 00 48 DE AC || 01 CE.

Prerequisite:

For this mode of operation the parameter M has the integer value eight.

C.2.3.2 CCM* mode encryption and authentication transformation

Inputs:

The inputs to the CCM* mode forward transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:

   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

b) The nonce N of $15 - L = 13$ octets to be used:

   Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 06.

c) The octet string m of length $l(m) = 1$ octets to be used:

   m = CE.

d) The octet string a of length $l(a) = 29$ octets to be used:

   a = 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF 01 00 00 00 00 48 DE AC || 06 || 05 00 00 00 || 01.

Actions:

The CCM* mode forward transformation involves the execution, in order, of an input transformation, as defined in C.2.3.2.1, an authentication transformation, as defined in C.2.3.2.2, and an encryption transformation, as defined in C.2.3.2.3.

C.2.3.2.1 Input transformation

This step involves the transformation of the input strings *a* and *m* to the strings AuthData and PlainTextData, to be used by the authentication transformation and the encryption transformation, respectively.

a) Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string *a*:

$L(a)$ = 00 1D.

b) Right-concatenate the octet string $L(a)$ and the octet string *a* itself:

$L(a) \| a$ = 00 1D || 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF
01 00 00 00 00 48 DE AC 06 05 00 00 00 01.

c) Form the padded message AddAuthData by right-concatenating the resulting string with the smallest non-negative number of all-zero octets so that the octet string AddAuthData has length divisible by 16.

AddAuthData = 00 1D 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF
01 00 00 00 00 48 DE AC 06 05 00 00 00 01 00.

d) Form the padded message PlaintextData by right-concatenating the octet string *m* with the smallest non-negative number of all-zero octets so that the octet string PlaintextData has length divisible by 16:

PlaintextData = CE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

e) Form the message AuthData consisting of the octet strings AddAuthData and PlaintextData:

AuthData = 00 1D 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF 01 00 00 00 00 48 DE AC 06 05 00 00 00 01 00 CE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

C.2.3.2.2 Authentication transformation

The data AuthData that was established in C.2.3.2.1 is tagged using the tagging transformation as follows:

a) Form the 1-octet Flags field:

Flags = 59.

b) Form the 16-octet B_0 field:

B_0 = 59 || AC DE 48 00 00 00 00 01 00 00 00 05 06 || 00 01.

c) Parse the message AuthData as $B_1 \| B_2 \| B_3$, where each message block B_i is a 16-octet string.

d) The CBC-MAC value X_4 is computed as shown in Table C.8.

Table C.8—Computation of CBC-MAC value

i	B_i	X_i
0	59 AC DE 48 00 00 00 00 01 00 00 00 05 06 00 01	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1	00 1D 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF	1C E4 F7 E4 FC 48 74 6D 0C 22 20 5D E8 DB B9 B0
2	FF 01 00 00 00 00 48 DE AC 06 05 00 00 00 01 00	16 EC 61 6D 5A C1 1A A0 4B 30 89 09 5D D5 7F 89
3	CE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	49 C3 1D 64 A5 A0 12 58 5B 07 78 B8 CD FE CE A8
4	—	D6 06 7B B5 9B 57 03 9C 00 98 0A 5D B3 63 BB 80

e) The authentication tag *T* is the result of omitting all but the leftmost M = 8 octets of the CBC-MAC value X_4:

T = D6 06 7B B5 9B 57 03 9C.

C.2.3.2.3 Encryption transformation

The data PlaintextData is encrypted using the encryption transformation as follows:

a) Form the 1-octet Flags field:
   Flags = 01.
b) Define the 16-octet A_i fields as shown in Table C.9.

Table C.9—A_i fields

i	A_i
0	01 ‖ AC DE 48 00 00 00 00 01 00 00 00 05 06 ‖ 00 00
1	01 ‖ AC DE 48 00 00 00 00 01 00 00 00 05 06 ‖ 00 01

c) Parse the message PlaintextData as M_1, where each message block M_i is a 16-octet string.
d) The cipher text block C_1 is computed as shown in Table C.10.

Table C.10—Computation of cipher text

i	AES(Key, A_i)	C_i = AES(Key, A_i) ⊕ M_i
1	16 A9 67 B4 0F F9 72 DE B1 CB 46 E7 09 FD EB FF	D8 A9 67 B4 0F F9 72 DE B1 CB 46 E7 09 FD EB FF

e) The string Ciphertext is the result of omitting all but the leftmost $l(m) = 1$ octet of the string C_1:
   CipherText = D8.
f) Define the 16-octet encryption block S_0:
   $S_0 = E(\text{Key}, A_0)$ = 99 D8 29 25 FA AE C5 6D 17 93 04 21 3B 88 69 35.
g) The encrypted authentication tag *U* is the result of XOR-ing the string consisting of the leftmost $M = 8$ octets of S_0 and the authentication tag *T*:
   U = 4F DE 52 90 61 F9 C6 F1.

Output:

c = D8 || 4F DE 52 90 61 F9 C6 F1.

C.2.3.3 CCM* mode decryption and authentication checking transformation

Inputs:

The inputs to the CCM* mode inverse transformation are as follows:

a) The key Key of size keylen = 128 bits to be used:
   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

b) The nonce N of $15 - L = 13$ octets to be used:

Nonce = AC DE 48 00 00 00 00 01 || 00 00 00 05 || 06.

c) The octet string c of length $l(c) = 9$ octets to be used:

c = D8 4F DE 52 90 61 F9 C6 F1.

d) The octet string a of length $l(a) = 29$ octets to be used:

a = 2B DC 84 21 43 02 00 00 00 00 48 DE AC FF FF || 01 00 00 00 00 48 DE AC || 06 || 05 00 00 00 || 01.

Actions:

The CCM* mode inverse transformation involves the execution, in order, of a decryption transformation, as defined in C.2.3.3.1 and an authentication checking transformation as defined in C.2.3.3.2.

C.2.3.3.1 Decryption transformation

The decryption transformation involves the following steps, in order:

a) Parse the message c as $C \| U$, where the rightmost string U is an 8-octet string:

C = D8;
U = 4F DE 52 90 61 F9 C6 F1.

b) Form the padded message CiphertextData by right-concatenating the string C with the smallest non-negative number of all-zero octets so that the octet string CiphertextData has length divisible by 16:

CipherTextData = D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

c) Form the 1-octet Flags field:

Flags = 01.

d) Define the 16-octet A_i fields as shown in Table C.11.

Table C.11—A_i fields

i	A_i
0	01 ‖ AC DE 48 00 00 00 00 01 00 00 00 05 06 ‖ 00 00
1	01 ‖ AC DE 48 00 00 00 00 01 00 00 00 05 06 ‖ 00 01

e) Parse the message CiphertextData as C_1, where each message block C_i is a 16-octet string.

f) The plaintext block P_1 is computed as shown in Table C.12.

Table C.12—Computation of plaintext

i	AES(Key, A_i)	P_i = AES(Key, A_i) ⊕ C_i
1	16 A9 67 B4 0F F9 72 DE B1 CB 46 E7 09 FD EB FF	CE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

g) The octet string m is the result of omitting all but the leftmost $l(m) = 1$ octet of the string P_1:

m = CE.

h) Define the 16-octet encryption block S_0:

$S_0 = E(\text{Key}, A_0)$ = 99 D8 29 25 FA AE C5 6D 17 93 04 21 3B 88 69 35.

i) The purported authentication tag *T* is the result of XOR-ing the string consisting of the leftmost *M* = 8 octets of S_0 and the octet string *U*:

T = D6 06 7B B5 9B 57 03 9C.

C.2.3.3.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

a) Form the message AuthData using the input transformation in C.2.3.2.1, with as inputs the string *a* and the octet string *m* that was established in C.2.3.3.1 (step g):

AuthData = 00 1D DC 84 21 43 02 00 00 00 00 48 DE AC FF FF 01 00 00 00 00 48 DE AC 06 05 00 00 00 01 00 CE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00.

b) Use the authentication transformation in C.2.3.2.2 with as input the message AuthData to compute the authentication tag MACTag:

MACTag = D6 06 7B B5 9B 57 03 9C.

c) Compare the output tag MACTag resulting from this transformation with the tag *T* that was established in C.2.3.3.1 (step i):

T = D6 06 7B B5 9B 57 03 9C = MACTag.

Output:

Because MACTag = *T*, output valid," accept the octet strings *a* and *m*, and accept one of the key sharing group member(s) as the source of *a* and *m*.

Annex D

(informative)

Protocol implementation conformance statement (PICS) proforma[11]

D.1 Introduction

To evaluate the conformance of a particular implementation, it is necessary to have a statement of which capabilities and options have been implemented for a given standard. Such a statement is called a protocol implementation conformance statement (PICS).

D.1.1 Scope

This annex provides the PICS proforma for this standard in compliance with the relevant requirements.

D.1.2 Purpose

The supplier of a protocol implementation claiming to conform to this standard shall complete the following PICS proforma and accompany it with the information necessary to identify fully both the supplier and the implementation.

The PICS of a protocol implementation is a statement of which capabilities and options of the protocol have been implemented. The statement is in the form of answers to a set of questions in the PICS proforma. The questions in a proforma consist of a systematic list of protocol capabilities and options as well as their implementation requirements. The implementation requirement indicates whether implementation of a capability is mandatory, optional, or conditional depending on options selected. When a protocol implementor answers questions in a PICS proforma, the implementor indicates whether an item is implemented and provides explanations if an item is not implemented.

D.2 Abbreviations and special symbols

Notations for requirement status:

M Mandatory
O Optional
O.n Optional, but support of at least one of the group of options labeled O.n is required.
N/A Not applicable
X Prohibited
“item”: Conditional, status dependent upon the support marked for the “item”

For example, FD1: O.1 indicates that the status is optional but at least one of the features described in FD1 and FD2 is required to be implemented, if this implementation is to follow the standard to which this PICS proforma is part.

[11]Copyright release for PICS proformas: Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

D.3 Instructions for completing the PICS proforma

If it is claimed to conform to this standard, the actual PICS proforma to be filled in by a supplier shall be technically equivalent to the text of the PICS proforma in this annex and shall preserve the numbering, naming, and ordering of the PICS proforma.

A PICS that conforms to this annex shall be a conforming PICS proforma completed in accordance with the instructions for completion given in this annex.

The main part of the PICS is a fixed-format questionnaire, divided into tables. Answers to the questionnaire are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (such as Yes or No) or by entering a value or a set or range of values.

D.4 Identification of the implementation

Implementation under test identification

Implementation under test name: _____

Implementation under test version: _____

_____

System under test identification

System under test name _____

Hardware configuration: _____

_____

_____

Operating system: _____

Product supplier

Name: _____

Address: _____

_____

Telephone number: _____

Facsimile number: _____

Email address: _____

Additional information: _____

Client

Name: _____

Address: _____

_____

Telephone number: _____

Facsimile number: _____

Email address: _____

Additional information: _____

PICS contact person

Name: _____

Address: _____

_____

Telephone number: _____

Facsimile number: _____

Email address: _____

Additional information: _____

PICS/System conformance statement

Provide the relationship of the PICS with the system conformance statement for the system: _____

_____

_____

D.5 Identification of the protocol

This PICS proforma applies to IEEE Std 802.15.4-2011.

D.6 Global statement of conformance

The implementation described in this PICS proforma meets all of the mandatory requirements of the referenced standard.

[ ] Yes

[ ] No

NOTE—Answering ”No” indicates nonconformance to the specified protocol standard. Nonsupported mandatory capabilities are to be identified in the following tables, with an explanation by the implementor explaining why the implementation is nonconforming.

The supplier will have fully complied with the requirements for a statement of conformance by completing the statement contained in this subclause. However, the supplier may find it helpful to continue to complete the detailed tabulations in the subclauses that follow.

D.7 PICS proforma tables

The following tables are composed of the detailed questions to be answered, which make up the PICS proforma.

D.7.1 Functional device types

The requirements for the functional device types are described in Table D.1.

Table D.1—Functional device types

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
FD1	Is this a full function device (FFD)	5.1	O.1			
FD2	Is this a reduced function device (RFD)	5.1	O.1			
FD3	Support of 64 bit IEEE address	5.2.1.1.6	M			
FD4	Assignment of short network address (16 bit)	5.1.3.1	FD1: M			
FD5	Support of short network address (16 bit)	5.2.1.1.6	M			
O.1 At least one of these features shall be supported.						

D.7.2 Major capabilities for the PHY

The requirements for the major PHY capabilities are given in this subclause.

D.7.2.1 PHY functions

The requirements for the PHY functions are described in Table D.2.

Table D.2—PHY functions

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
PLF1	Energy detection (ED)	8.2.5	FD1: M O			
PLF2	Link quality indication (LQI)	8.2.6	M			
PLF3	Channel selection	8.1.2	M			
PLF4	Clear channel assessment (CCA)	8.2.7	M			
PLF4.1	Mode 1	8.2.7	O.2			
PLF4.2	Mode 2	8.2.7	O.2			
PLF4.3	Mode 3	8.2.7	O.2			
PLF4.4	Mode 4	14.4.15	RF4: O.6			
PLF4.5	Mode 5	14.4.15	RF4: O.6			
PLF4.6	Mode 6	14.4.15, 14.6	RF4: O.6			
PLF5	Ranging	14.7	RF4: O			
PLF5.1	Crystal characterization	14.7.2	O			
PLF5.2	Dynamic preamble selection (DPS)	5.1.8.3	O			
PLF6	Default UWB pulse shape	14.3	RF4: M			
PLF6.1	Chirp on UWB (CoU)	14.5.1	O			
PLF6.2	Continuous spectrum (CS)	14.5.2	O			
PLF6.3	Linear combination of pulses (LCP)	14.5.3	O			
O.2 At least one of these features shall be supported. O.6 At least one of these features shall be supported.						

D.7.2.2 Radio frequency (RF)

The requirements for the PHY RF capabilities are described in Table D.3.

D.7.2.3 Channel capabilities for UWB PHY

The UWB channel requirements are described in Table D.4.

Table D.3—Radio frequency (RF)

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
RF1	BPSK PHY	Clause 11	O.3			
RF1.1	868 MHz band	Table 66	M			
RF1.2	915 MHz band	Table 66	M			
RF1.3	ASK PHY	Table 66, Clause 12	O			
RF1.4	O-QPSK PHY in 868 MHz or 915 MHz bands	Table 66, Clause 10	O			
RF2	2450 MHz O-QPSK PHY	Table 66, Clause 10	O.3			
RF3	CSS PHY	Clause 13, Table 66, Table 67	O.3			
RF4	UWB PHY	Clause 14, Table 66, Table 68	O.3			
RF4.1	250–750 MHz UWB PHY	Table 66, 14.4	O.5			
RF4.2	3244–4742 MHz UWB PHY	Table 66, 14.4	O.5			
RF4.3	5944–10 234 MHz UWB PHY	Table 66, 14.4	O.5			
RF5	780 MHz band	Clause 10, Annex G	O.3			
RF5.1	O-QPSK PHY in 780 MHz band	Clause 10	RF5: O.7			
RF5.2	MPSK PHY in 780 MHz band	Annex G	RF5: O.7			
RF6	950 MHz band	Clause 11, Clause 15, Table 66	O.3			
RF6.1	BPSK PHY in 950 MHz band	Clause 11	RF6: O.8			
RF6.2	GFSK PHY in 950 MHz band	Clause 15	RF6: O.8			
RF7	Supports CSS PHY 1 Mb/s	Clause 13	RF3: M			
RF7.1	Supports CSS PHY 250 kb/s	Clause 13	O			
RF8	Supports UWB PHY 850 kb/s (rate 01)	14.2.3, 14.2.6.1	RF4: M			

Table D.3—Radio frequency (RF) *(continued)*

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
RF8.1	Supports rate 00	14.2.3, 14.2.6.1	O			
RF8.2	Supports rate 02	14.2.3, 14.2.6.1	O			
RF8.3	Supports rate 03	14.2.3, 14.2.6.1	O			
RF8.4	Supports rate 04	14.2.3, 14.2.6.1	O			
RF9	Support 950 MHz band channels 14 and 17	8.1.2.3	RF6: O			
O.3 At least one of these features shall be supported. O.5 At least one of these features shall be supported. O.7 At least one of these features shall be supported. O.8 At least one of these features shall be supported.						

Table D.4—UWB channels

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
PCH1	Channel number 0	Table 68	RF4.1: M			
PCH2	Channel number 1	Table 68	RF4.2: O			
PCH3	Channel number 2	Table 68	RF4.2: O			
PCH4	Channel number 3	Table 68	RF4.2: M			
PCH5	Channel number 4	Table 68	RF4.2: O			
PCH6	Channel number 5	Table 68	RF4.3: O			
PCH7	Channel number 6	Table 68	RF4.3: O			
PCH8	Channel number 7	Table 68	RF4.3: O			
PCH9	Channel number 8	Table 68	RF4.3: O			
PCH10	Channel number 9	Table 68	RF4.3: M			
PCH11	Channel number 10	Table 68	RF4.3: O			
PCH12	Channel number 11	Table 68	RF4.3: O			
PCH13	Channel number 12	Table 68	RF4.3: O			
PCH14	Channel number 13	Table 68	RF4.3: O			
PCH15	Channel number 14	Table 68	RF4.3: O			
PCH16	Channel number 15	Table 68	RF4.3: O			

D.7.3 Major capabilities for the MAC sublayer

The major capabilities for the MAC sublayer are described in this subclause.

D.7.3.1 MAC sublayer functions

The MAC sublayer function requirements are described in Table D.5.

Table D.5—MAC sublayer functions

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
MLF1	Transmission of data	6.3	M			
MLF1.1	Purge data	6.3.4, 6.3.5	FD1: M FD2: O			
MLF2	Reception of data	6.3	M			
MLF2.1	Promiscuous mode	5.1.6.5	FD1: M FD2: O			
MLF2.2	Control of PHY receiver	6.2.9	O			
MLF2.3	Timestamp of incoming data	6.3.2	O			
MLF3	Beacon management	Clause 5	M			
MLF3.1	Transmit beacons	Clause 5, 5.1.2.4	FD1: M FD2: O			
MLF3.2	Receive beacons	Clause 5, 6.2.4	M			
MLF4	Channel access mechanism	Clause 5, 5.1.1	M			
MLF5	Guaranteed time slot (GTS) management	Clause 5, 6.2.6, 5.3.9, 5.1.7	O			
MLF5.1	GTS management (allocation)	Clause 5, 6.2.6, 5.3.9, 5.1.7	O			
MLF5.2	GTS management (request)	Clause 5, 6.2.6, 5.3.9, 5.1.7	O			
MLF6	Frame validation	6.3.3, 5.2, 5.1.6.2	M			
MLF7	Acknowledged frame delivery	Clause 5, 6.3.3, 5.2.1.1.4, 5.1.6.4	M			
MLF8	Association and disassociation	Clause 5, 6.2.2, 6.2.3, 5.1.3	M			
MLF9	Security	Clause 7	M			
MLF9.1	Unsecured mode	Clause 7	M			
MLF9.2	Secured mode	Clause 7	O			

Table D.5—MAC sublayer functions *(continued)*

Item number	Item description	Reference	Status	Support		
				N/A	Yes	No
MLF9.2.1	Data encryption	Clause 7	O.4			
MLF9.2.2	Frame integrity	Clause 7	O.4			
MLF10.1	ED	5.1.2.1, 5.1.2.1.1	FD1: M FD2: O			
MLF10.2	Active scanning	5.1.2.1.2	FD1: M FD2: O			
MLF10.3	Passive scanning	5.1.2.1.2	M			
MLF10.4	Orphan scanning	5.1.2.1, 5.1.2.1.3	M			
MLF11	Control/define/ determine/declare superframe structure	5.1.1.1	FD1: O			
MLF12	Follow/use superframe structure	5.1.1.1	O			
MLF13	Store one transaction	5.1.5	FD1: M			
MLF14	Ranging	5.1.8	RF4: O			
MLF14.1	DPS	5.1.8.3, 6.2.15	O			
O.4 At least one of these features shall be supported.						

D.7.3.2 MAC frames

The MAC frame requirements are described in Table D.6.

Table D.6—MAC frames

Item number	Item description	Reference	Transmitter		Receiver	
			Status	Support N/A Yes No	Status	Support N/A Yes No
MF1	Beacon	5.2.2.1	FD1: M		M	
MF2	Data	5.2.2.2	M		M	
MF3	Acknowledgment	5.2.2.3	M		M	
MF4	Command	5.2.2.4	M		M	
MF4.1	Association request	5.2.2.4, 5.3.1	M		FD1: M	
MF4.2	Association response	5.2.2.4, 5.3.2	FD1: M		M	

Table D.6—MAC frames *(continued)*

Item number	Item description	Reference	Transmitter		Receiver	
			Status	Support N/A Yes No	Status	Support N/A Yes No
MF4.3	Disassociation notification	5.2.2.4, 5.3.3	M		M	
MF4.4	Data request	5.2.2.4, 5.3.4	M		FD1: M	
MF4.5	PAN identifier conflict notification	5.2.2.4, 5.3.5	M		FD1: M	
MF4.6	Orphaned device notification	5.2.2.4, 5.3.6	M		FD1: M	
MF4.7	Beacon request	5.2.2.4, 5.3.7	FD1: M		FD1: M	
MF4.8	Coordinator realignment	5.2.2.4, 5.3.8	FD1: M		M	
MF4.9	GTS request	5.2.2.4, 5.3.9	MLF5: O		MLF5: O	

Annex E

(informative)

Location topics

E.1 Overview

Ranging capability is achieved through support of a number of specific PHY capabilities as well as defined MAC behaviors and protocols.

E.1.1 Two-way ranging

UWB devices that have implemented ranging support are called ranging-capable devices (RDEVs). UWB PHYs have a bit in the PHY header (PHR) called the ranging bit that serves to signal to the receiver that this particular frame is intended for ranging. A UWB frame with the ranging bit set in the PHR is called a ranging frame (RFRAME). There is nothing else (beyond the ranging bit set in the PHR) that makes an RFRAME unique. RFRAMEs can carry data, RFRAMEs can be acknowledgments, and RFRAMEs do not even (for the case of one-way ranging) necessarily require an acknowledgment. As far as ranging is concerned, the critical instant in a frame is the first pulse of the PHR. The first pulse of the PHR is the ranging marker (RMARKER). This standard is primarily structured to support the two-way time-of-flight computation of distance between two RDEVs. Figure E.1 illustrates the complete sequence for two-way ranging.

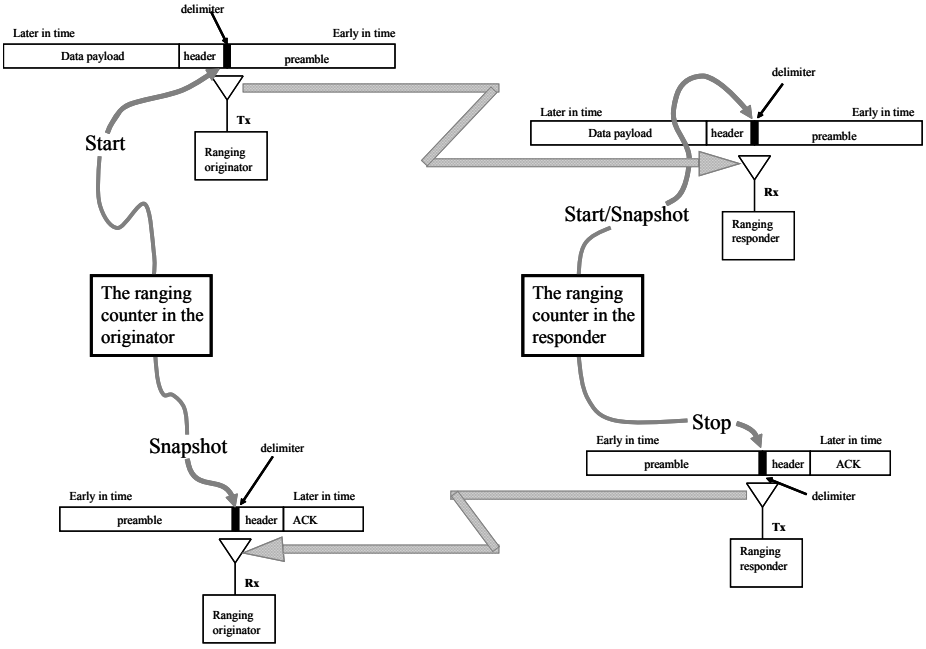


Figure E.1—The complete two-way ranging technique

The first frame of the two-way exchange is an RFRAME that is sent from the originating device to the responding device, as shown in the top half of Figure E.1. A ranging counter start value is captured in the originator device upon the RMARKER departure from the originator, and a ranging counter start value is

captured in the responding device upon RMARKER arrival at the responder. The RFRAME has the acknowledge request bit set in the MHR. In the most general case, the counter in the responder PHY will have already started running when a previous RFRAME arrived, but the previous RFRAME was not intended for this device and thus did not get an acknowledgment from this device. In Figure E.1, the counter activity is labeled “start/snapshot” from the PHY perspective. For the PHY, the counter function is “start” for the first arriving frame and “snapshot” for subsequent frames. Snapshot means that the value of the counter is captured and stored at the instant of the snapshot, but the counter continues to count as if the snapshot had not happened. The responder PHY initiates the counter snapshot values for all arriving RFRAMEs. The responder MAC discards the snapshot values that are for RFRAMEs not intended for the responder device. At the end of the first frame transmission in Figure E.1, the counters are running in both devices.

The RFRAME shown in the bottom half of Figure E.1 is an acknowledgment sent from the responding device to the originating device. The ranging counter stop value is a snapshot in the responding device upon RMARKER departure from the responder. The responder PHY is now in transmit mode, and the counter is still running. Because the PHY is in transmit mode, it will not be receiving any frames or taking any counter snapshots. Leaving the counter running in the responder at this point in the algorithmic flow only serves to deplete the battery of a mobile device. In Figure E.1, the counter action is labeled stop, not because it really is stopped (it is not), but because the algorithmic flow is done with it and because it will appear to the application as if it has stopped because it will generate no more snapshots. The counter stop value in the originator device is snapshot and saved upon RMARKER arrival at the originator. The originator MAC verifies that the frame was from the responder, and ultimately the application will then stop the counter using MLME primitives.

When the application in the responding device learned that the acknowledgment had been sent, it stopped the ranging counter in the PHY. When the application at the originator device discovered that the acknowledgment frame was for the originator device, it stopped the counter in the PHY. Thus, in Figure E.1, all the counters have stopped, and the values are located in the respective devices.

The discussion in this subclause risks confusion because it includes the general case of arriving frames not intended for the devices. While that behavior is important for algorithmic robustness, for understanding basic ranging, it is a distraction. In Figure E.1, the ranging pair holds two different sets of counter values, with a start value and a stop value in each set. Even if some counter snapshots have been discarded by the application due to frames destined for other devices, what remains at the end of the exchange are pairs of counter values that when subtracted represent the elapsed times between the arrival and the departure of the intended frames.

At the system state represented by Figure E.1, the necessary information required to compute the range between the devices is known. However, the information is still distributed in the system; and before the ranging computation can be accomplished, the data are brought to a common compute node. The difference of the counter start and stop values in the originator device represents the total elapsed time from the departure of the first message to the arrival of the acknowledgment. The difference of the counter start and stop values in the responder represents the total elapsed time from arrival of the data message to the departure of the acknowledgment. After these values are all brought together at a common compute node, they are subtracted, the difference is divided by two, and the time of flight (and thus the inferred range) is known.

While management of the ultimate disposition of the counter values is outside the scope of this standard, the most immediately obvious resolution is for the application at the responder device to send the counter value to the originator device in a data frame. The originator device started the exchange; therefore, it might be assumed that it is the point at which the application desires to have the range information. The responder device was just in radio contact with the originator device; therefore, a communication channel is very likely to be available.

The obvious solution suggested in the previous paragraph is often the wrong solution. This standard supports applications that bring a large number of ranging measurements together at a single computation device, and there the application solves not just for the ranges between individual devices but also the two- or three-dimensional relative location of the devices. A discussion of typical activities at a central compute node is included in Annex E.

E.1.2 Position awareness through one-way transmissions

The primary intent of the ranging support in this standard is to support ranging through two-way time-of-flight measurements. To establish that capability, this standard defines a whole suite of capabilities and behaviors by which two-way ranging is enabled. The capabilities required to accomplish one-way ranging are sufficiently similar so that this standard allows operation in that mode as well. One-way ranging requires an infrastructure of RDEVs and some means to establish a common notion of time across those devices. The protocol to establish the common notion of time is outside the scope of this standard. What this standard does provide is a bit in the PHR that serves to signal an RDEV that location awareness is desired. To operate in a one-way environment, any UWB device (not necessarily an RDEV) simply sends an RFRAME having an appropriate preamble length. As described in E.1.1, after the PHY counter is running in an infrastructure RDEV, the MAC initiates an MCPS-DATA.indication primitive to the next higher layer for all arriving RFRAMEs. By this mechanism, the application acquires a list of counter values representing the arrival times of all one-way ranging messages received at all infrastructure devices. As with the case of two-way ranging, nothing useful can happen with the lists of counter values until they are brought together at a common compute node; and as with two-way ranging, that bringing together activity is beyond the scope of this standard. After the counter values have been brought together, the computation of the relative locations proceeds as shown in E.5.2.

The PHY will initiate an MCPS-DATA.indication for all arriving RFRAMEs. That can be thought of as a great goal; and if the RFRAMEs arrive infrequently, it is a very achievable goal. However, the PHY does not get to choose how quickly a sequence of RFRAMEs might arrive; and in real-world applications, it is possible that the RFRAMEs arrive more quickly than the PHY can deal with them. The processing time for an RFRAME is very dependent on the implementation of the PHY. While the PHY has no control over the inter-arrival time of RFRAMEs, the application very well might. The application can discover the RFRAME processing time by reading the PHY PAN information base (PIB) attribute *phyRFRAMEProcessingTime*. The intended use of this attribute is that if an application discovers the processing capabilities of the devices in the network, it can structure the traffic so that devices are not overrun. Because the only purpose is to help prevent overruns, there is no need for high precision in expressing this value.

E.1.3 The ranging counter

At the most fundamental level, ranging capability as described in this standard is enabled by the ranging counter shown in the center boxes of Figure E.1. The ranging counter has the capability of assigning values to the precise instant that RMARKERS are transmitted and received from the device. Once that counter and the ability for it to precisely snapshot a timestamp are in place, then conceptually, the computation of the time of flight is simple.

An actual physical counter that exhibits the behavior attributed to the “ranging counter” does not need to exist in any PHY implementation. The ranging counter described is simply an abstraction that is used to specify the required PHY behavior.

The LSB of the ranging counter represents a time interval so small that an actual physical counter would have to run at a nominal 64 GHz to produce values with the required resolution. It is unlikely that a device intended for low-cost, battery-powered operation would implement a counter running at 64 GHz.

The lack of an actual physical realization does not in any way preclude the use of the ranging counter as an abstraction in this standard to visualize and specify the behavior of an RDEV. From an algorithmic and computational viewpoint, the RDEV will appear to an application as if it possesses a 64 GHz counter with the ability to start and stop based on the state of bits before they arrive.

The implementation of the ranging counter is beyond the scope of this standard; however, the following ideas are suggested:

— Take snapshots of the counter at every event when counter value might be required and then discard unneeded snapshots after the future becomes known.
— Do not build a 64 GHz counter, but rather generate the less significant bits of the ranging counter values using computational techniques like those described in E.2.

E.1.4 Accounting for signal arrival time

The start counter values represent the time of arrival of the first pulse of the first symbol of the header of a PPDU. That necessary task is not trivial when the signal arrives in a channel with significant multipath. The first pulse of the header arrives at the antenna once for every multipath reflection in the environment. The result is that in an indoor environment, the "first pulse" can seem to arrive at a multitude of different times. Accurate ranging requires discriminating the leading edge of the cluster of signal arrivals that accounts for the first pulse of the header. The technique for achieving this discrimination is outside of the scope of this standard, but it is helpful to discuss a typical approach. In a typical approach, the counter value is a snapshot relative to a position on the arriving waveform where the PHY-tracking loop has achieved and is maintaining a "signal lock spot." Then the offset from the lock spot to the leading edge of the pulse energy is determined. After a time offset from the UWB signal lock spot to the leading edge of the energy cluster is found, the rest is very straightforward: that offset is just subtracted from the counter value exactly as the other correction factors are. The time offset to the leading edge is discovered by sampling the energy ahead of the acquisition lock spot over multiple different offsets to discover the earliest point with discernible energy. To achieve the required precision, the sample values in the vicinity of the leading edge could be further refined using techniques like those shown in E.2. For those computations (and other techniques, generally known as upsampling), it is critical that the noise in the samples be well suppressed. This standard supports this leading edge activity by allowing a very long acquisition preamble keeping the signal steady and data free for a protracted time.

The necessary characterization of the channel multipath response is generically called a channel sounding. The techniques that accomplish that task can be numerically intense. A system designer could accomplish that task in the PHY. However, a sounding mechanism involving the MLME primitives has been defined. This allows the PHY to present raw data to a higher layer should the PHY lack significant computational capability and the system designer wishes to employ numerically intensive channel sounding algorithms. The raw data can then be moved to whatever device has sufficient computational resources to support the desired algorithms. The primitives supplied are very simple and assume that both the MAC and the application are well behaved and give priority attention to sounding activities. For example, the primitives do not include tags to associate a particular sounding with a particular packet. The application is responsible for making sure that sounding activities are conducted in a timely way so that the sounding information is associated with the last packet received. The actual handling of the raw data after the sounding operation facilitates uploading to a next higher layer is beyond the scope of this standard.

E.1.4.1 Leading edge search during the acquisition preamble

Upon acquisition of the signal, the PHY is not aware of how much time remains in the preamble before the delimiter. A reasonable goal is to do the best possible job of bracketing the leading edge with whatever time is available and then reporting how well the leading edge was bracketed by way of the ranging FoM. In a typical implementation, if the delimiter arrives very quickly after the acquisition threshold was satisfied,

then the leading edge equipment will still be using coarse steps to characterize the energy. The PHY will make the best judgment it can about the leading edge based on the coarse steps and then report a FoM value appropriate for coarse steps. If the leading edge search engine has ample time before the delimiter arrives, then not only can it have progressed to using a very fine search step, but it can also have integrated many samples to drive down the noise in the computation. In this case, the correction representing the leading edge offset is applied to the counter value (and it might have been the very same correction value as was applied in the previous case when the search time was short), but this time the FoM value is reported for a very small characterization bin and very high confidence that the leading edge truly was in that bin. Again note that the counter value returned with a good FoM can have the same value as the counter value returned with a bad FoM; i.e., the counter value is independent of the FoM.

E.1.4.2 FoM for bad times

If the PHY performs a short leading edge search (as will happen after recovering from an acquisition false alarm, for example), it still makes its best guess for a leading edge correction and goes on with the ranging algorithm. Even when the final counter value represents a known error-prone measurement, the PHY should not return a FoM of zero. Zero means "no FoM," which is neither correct nor useful. An appropriate FoM to report for the most error-prone cases is 0x79. That value decodes to tell the application that even if the other RDEV calculated its half of the measurement perfectly, given the expected error just due to this RDEV's measurement alone, the PHY is 80% confident that the computed range will be wrong by more than 2 m.

E.1.4.3 Other opportunities for leading edge search refinement

The previous discussion was framed as if all channel characterization had to stop upon the arrival of the delimiter. In fact, PHYs can do additional things after the delimiter to further refine the estimate of the leading edge offset. The UWB CCA pulses described in 14.6, if used, offer additional opportunities to look at the signal in a known state after the delimiter. A very sophisticated PHY can perform additional characterization during the time that data are on the air if the algorithm designer is willing to "back out" the effects of the data after demodulation (and thus after the data are "known"). From the application's point of view, all the application sees for the difference between a very capable PHY and a sloppy, mediocre PHY is a difference in the FoM values being reported.

E.1.4.4 Managing the preamble length for leading edge search

One of the most distinguishing traits of ranging UWB radio transmissions is the long preambles. This standard allows the application to specify preambles that are either 1024 or 4096 symbol repetitions long. The selection is a function of the channel multipath, the signal-to-noise ratio (SNR) in the link, and the capability of the receiving PHY. It is theoretically possible that a very capable PHY that does leading edge refinement using the data could do ranging accurately with a preamble that is 16 symbols long. It is also possible (likely, in fact) that a PHY with a poorly designed search engine will not do a good job in a heavy multipath even with a 4096 symbol preamble. The upper layers are responsible for picking the preamble length. It is suggested that the application start ranging operations using the 1024 symbol preamble and keep a history of how the FoMs are reported. The FoMs are the critical feedback information that tells the application how the various PHYs are doing, and the application can make future adjustments to the preamble length based on that history.

E.1.4.5 PHY deferral of the computations for leading edge search

This standard provides a mechanism to allow the PHY to pass the computational burden of leading edge processing to a higher layer. If the computations are not done in the PHY, then the value in the timestamp report for RangingCounterStart is not corrected for the leading edge. The RangingFOM is used to signal this condition to the higher layer, which will have to compute a correction based on data acquired using the sounding primitives, as described in 14.7.3. The higher layer issues a MLME-SOUNDING.request primitive. The associated MLME-SOUNDING.confirm response returns a list SoundingPoints where each

SoundingPoint is a pair of integers representing data taken by the PHY at time offsets from the point on the waveform represented by the uncorrected value in RangingCounterStart. A time of zero in the list designates an amplitude value taken at the point indicated by RangingCounterStart. Positive time values indicate amplitudes that occurred earlier in time than the zero point. The amplitude values do not represent any particular voltage. They are only meaningful in a relative sense and in the context of each other. The values are linear (not logarithmic). The amplitude values are all consistent with each other. For example, it is acceptable for an automatic gain control (AGC) circuit to change the gain during the measurement of the amplitudes, if the numbers are corrected so that the effect of the gain change is removed and the numbers returned in a SOUNDING.confirm primitive are the values that would have been measured had the gain been perfectly stable and unchanged for all measurements. The list of measurements returned by the SOUNDING.confirm primitive begins with the size of the list. The maximum size that can be represented is 65 K value pairs. That large value is only because a single octet would not be adequate to represent lists larger than 255 pairs. In practical systems, lists larger than 255 pairs can occur. Two octets would be the next choice to represent the list size, but that does not mean that lists approaching 65 K pairs would be appropriate. There is no particular acceptable or unacceptable list size. Generally, a larger list is superior, as described in E.1.4.1. In the case where the PHY is deferring the leading edge computation to an upper layer, the PHY does not assign a FoM to the timestamp report. That does not mean that a FoM is unneeded by algorithms at the higher layers; it just means that the PHY will not be the source. In cases where the PHY defers computation, the upper layer will typically compute a FoM for itself based on the size and quality of the list returned with the SOUNDING.confirm primitive.

E.1.4.6 PHY deferral of the computations for self-calibration

The sounding primitives provide a mechanism to offload from the PHY the computational burden of analyzing a channel sounding for the leading edge of an arriving signal. A very similar problem arises in the self-calibration of a ranging UWB PHY. One excellent technique for self-calibration is the “sounding” of a loopback path in the radio. In this technique, the PHY actually transmits to itself through reflections associated with the transmission path (like a transmit/receive switch or the antenna itself). When using this technique, the issue comes up again that it can be computationally intense to discover the moment of arrival of the (often small) amplitude disturbances associated with elements in the path to the antenna. Implementers can choose to achieve this computational effort in the PHY. In an alternate implementation, the sounding mechanism can be used to offload this computational burden to a higher layer. When performing a sounding for leading edge computation, the instant associated with time zero is the point associated with signal tracking. Sounding for calibration is slightly different in that the time associated with zero is the launching of the pulse event from logic at the level of the ranging counter.

E.1.5 Management of crystal offsets

The numbers that will be subtracted in the range computation will typically represent times on the order of 5 ms. The time of flight for a 10 m link is about 30 ns. The expected difference of the counter values will be twice the time of flight, or something like 60 ns, in this example.

When a subtraction of values representing 5 ms is supposed to yield a meaningful answer on the order of 60 ns, even small percentage errors in the relative measurement of the 5 ms numbers yield large errors in the difference. The root cause of these errors is the fact that each of the individual 5 ms measurements was made with different crystals in different devices.

The crystals’ oscillators in different devices can generate frequency errors of 20 ppm. A 20 ppm error in a 5 ms number can account for 100 ns. This 100 ns error is disconcerting considering the 60 ns time-of-flight result.

Management of the errors due to crystal differences is essential to ranging. Correcting for a crystal difference algebraically at the time of the subtraction computation is straightforward if the difference is precisely known. The crux of the problem is to determine the crystal difference.

The mechanisms to characterize the crystal difference will be present and functioning in typical UWB PHY implementations. This crystal characterization equipment is the signal tracking loop in the receiver. A UWB pulse occupying 500 MHz has a nominal envelope width of 2 ns. The receiver tracking loop in a UWB PHY will stay "locked on" to this envelope for the duration of a packet. In the case of ranging packets, this will typically amount to milliseconds. In actual practice, the tracking loop will hold the sampling point on the envelope much tighter than 2 ns; therefore, by the end of the transmission, the tracking loop has the information to hold its sample point steady (with respect to its local crystal) on the received signal (which is sourced by the other devices crystal). In other words, by the end of the packet, the tracking loop has exactly measured the crystal difference. This crystal difference is the very thing necessary to correct the values in the ranging computation.

E.1.5.1 Characterizing crystal offsets with digital tracking loops

For a digital tracking loop, the most convenient way to express the crystal difference is with two numbers. A tracking interval number is the total number of units during which the signal was tracked, and a tracking offset number is a count of the number of times the tracking loop had to add or drop a unit to hold the sample point steady on the incoming signal. If the other oscillator frequency was lower than the tracking loop's local oscillator, then the tracking loop would be adding units to hold the sample point steady. The tracking offset is simply a count and a sign bit. All numbers are expressed from the local device's point of view; therefore, positive counts are characterizing crystals in the other device that are running slower (so the receiver was adding time units to match it) and negative numbers characterize faster crystals in the other device (so the receiver was dropping local units to keep up). The offset is thus a signed magnitude integer (not the twos-complement number that might be expected). The actual units (generally called "parts") that are called out in the count are whatever units happen to be convenient for a given PHY implementation. Since the numbers are used only as a ratio, the type of unit need not be specified as long as the numbers express the same unit.

E.1.5.2 Characterizing crystal offsets with analog tracking loops

PHYs that use analog phase-locked loops (PLLs) to track the received signal do not lend themselves as directly to the expression of the tracking offsets as counts. However, the PLL steady-state error signal is still a direct measure of the crystal offset. The analog PLL-based PHY can convert the PLL error signal to a number [for example, with an analog-to-digital converter (ADC)], put that result in the offset count field (taking care to get the sign bit correct), and put a convenient scaling number (like a million) into the total tracking interval field so that the ratio again expresses the difference of the crystals from the local oscillator's point of view.

E.1.5.3 Characterizing crystal offsets with different tracking loops

The use of the receiver's tracking loop to characterize the crystal offset is convenient for some PHY implementations, but it is not required for compliance. In fact, RDEVs are not required to support crystal characterization at all. If two RDEVs are involved in a ranging exchange and only one of them is supporting crystal characterization, all the information needed for a good ranging computation is available. If both RDEVs support crystal characterization, they will get the same ratio with opposite sign; therefore, there is little new information.

If neither RDEV supports crystal characterization, the application puts more traffic on the air to support ranging. For this situation, the application does the measurement twice. The first time is simply the normal exchange. On the second measurement, the roles are reversed. The device that was the originator on the first measurement is the responder for the second measurement, and likewise the responder on the first

measurement becomes the originator for the second measurement. Then the application does the range computation twice. Because neither measurement provided for correcting for crystal offsets, the answers for both measurements are likely to be totally wrong. But, the computations did involve the same crystals so the error in the measurements is the same. Because the application reversed the measurement sequence between the two measurements, the answers have errors with opposite signs. The bottom line is that while the two individual answers are both hopelessly inaccurate by themselves, the average of the two individual answers will be exactly correct. A further refinement of this technique is called symmetric double-sided two-way ranging (SDS-TWR) and is discussed in E.4.2. The refinement shown in Annex E seeks greater efficiency by combining the two independent measurements into a single stream with the originator sending an acknowledgment for the responder's acknowledgment. While the "acknowledge for an acknowledge" approach is absolutely sound mathematically for ranging and the additional efficiency is tempting, the "acknowledge for an acknowledge" message sequence construct is beyond the scope of this standard.

E.1.5.4 Size of units

The units of measurement in the crystal characterization ratio are not rigidly defined in this standard to allow vendors the freedom to choose a value that works well with their PHY implementation. Design freedom is good, but to ensure at least a minimum level of ranging accuracy, this standard insists on a value that allows the ratio to express individual parts per million of oscillator difference. When the ranging computation is done for typical packet sizes and turnaround times, the desired answer will typically only be tens of parts per million compared to the numbers being subtracted. It is in this context that this standard calls for using the nominal 500 MHz chip time (nominally 2 ns) as the largest acceptable unit for the crystal characterization numbers. While an implementation using this value will be compliant, it would typically yield ranging errors on the order of a meter due to poor crystal characterization. To achieve reduced ranging errors, it is recommended that smaller units for crystal characterization be used since this translates directly to reduced errors in the ranging computation. When both RDEVs of a two-way ranging exchange are supporting crystal characterization, the application has a choice of which set of numbers will be used to make the correction. The ranging application would be wise to manage the reality that different devices might present different quality results for the crystal characterization.

The LSB of the ranging counter caps the highest ranging accuracy that can be achieved by compliant devices (which use the straightforward two-way ranging techniques described here). The LSB represents a nominal 16 ps, which corresponds to about half a centimeter of flight for energy in free space.

E.1.6 Accounting for internal propagation paths

The ranging counter is supposed to be capable of measuring events at the device antenna very precisely. It is understood that an actual implementation will not be trivial. Typically the device's PHY will have a counter somewhere in the digital section, multiple correction values stored in registers, and some arithmetic hardware to apply the correction values. The end result of all this is that it appears (from the numbers reported) that the PHY has a counter that is somehow magically positioned right at the antenna of the device and is taking snapshots of the counter values for events as they happen right at the antenna. That is important because the computation is supposed to be for the time of flight through the air, not through some impedance matching network feeding an antenna. Subtracting the correcting values for internal propagation times is not hard. What is hard is actually knowing the values of the internal propagation times. This standard provides a CALIBRATE mechanism (involving MLME primitives) that allows an application to cause a PHY (at a time that the application chooses) to invoke whatever capability that the PHY might have to characterize the internal propagation times of the PHY. Inclusion of a device capability for antenna loopback with an associated self-calibration algorithm is encouraged, but beyond the scope of this standard. A defensively written ranging application can maintain tables of correction factors at the computation nodes where the table entries are individually associated with the unique devices it is using for ranging. In this way, the application can compensate (after the fact) for devices in the ranging environment that have done a poor job of self-calibration.

E.1.6.1 PIB attributes for internal propagation paths

This standard provides a defined place to go to for the correction factors characterizing the delays of the internal propagation paths. There are two separate PHY PIB attributes that separately cover the transmit and receive paths. The LSB of these values represents a time interval of about one-half of a centimeter of travel at the speed of light and is the same size as used throughout the ranging computations. The intended use of these PIB attributes is for them to be written by the application at a time of the application's choosing and for the values to stay with the device until rewritten. One possible way for the application to learn what values to write to the PIB attributes is to invoke the CALIBRATE primitives. This standard does not mandate that the CALIBRATE primitives are used. The standard simply makes them available for use, if desired.

E.1.6.2 Support for self-calibration and one-way ranging

This standard does not preclude position awareness through one-way ranging. Successful one-way ranging requires that the internal propagation paths to the transmit antenna and from the receive antenna be accounted for separately. The PHY PIB attribute *phyTXRMARKEROffset* represents the time from the internal ranging counter reference to the transmit antenna. Likewise, the PHY PIB attribute *phyRXRMARKEROffset* represents the time from the receive antenna to the internal ranging counter reference.

E.1.6.3 Use of the calibrate primitives

The CALIBRATE.confirm primitives return either the values that are correct for the RMARKER offsets, if the PHY takes care of all computations itself, or the status COMPUTATION_NEEDED. The actual implementation of the self-calibration could be as simple as returning hardwired values for the RMARKER offsets. In this situation, the hardwired values would be selected by the vendor at the time of device manufacture to represent a best guess of what the offsets might ultimately be. Alternatively, the self-calibrate implementation might involve a full channel sounding of a loopback path and a sophisticated pattern-matching algorithm to determine from the sounding waveform when an internally generated pulse reflected back from an antenna. In either of the two scenarios, the status COMPUTATION_NEEDED would not be used because either (in the first case) the calibrate implementation was so crude that there was nothing to do, or (in the second case) the calibrate implementation was so sophisticated that the PHY took care of all of the computations without assistance.

It is a property of the algorithms that a node which only does ranging transmissions within a one-way infrastructure-based ranging application need not support calibration in any form.

E.1.6.4 Use of the COMPUTATION_NEEDED status

Two extreme implementations of PHY self-calibration were described in E.1.6.2. This standard supports a reasonable middle ground implementation where the PHY does a channel sounding of a loopback path using the same hardware resources as normally used for leading edge scanning, but then the PHY defers the actual computations associated with the channel sounding to a higher layer. As discussed in E.1.4.5, the computations associated with processing a channel sounding can be numerically intense and can be beyond the resources of a particular PHY implementation.

When the higher layer receives a status of COMPUTATION_NEEDED in response to a CALIBRATE.request primitive, the higher layer will then use the sounding primitives, as described in E.1.4.6, to get a list of SoundingPoints from the PHY. The higher layer will then process the SoundingPoints to determine the values of the RMARKER offsets.

E.1.7 Timestamp reports

The two ranging counter values (start and stop), the ranging FoM value, and the two values that (as a ratio) characterize the crystal offsets all characterize a single ranging measurement. These five individual numbers that characterize a measurement are referred to in a group as a timestamp report. It then takes (at least) two timestamp reports to do a time-of-flight computation. The numbers in a single timestamp report have meaning in the context of each other. As such, they are generated by the PHY as a set and not split apart during subsequent data handling.

E.1.7.1 Presentation of timestamp reports

It should be noted that timestamp reports will occur at seemingly nonintuitive times in the actual primitives and the message sequence charts. For example, a timestamp report is included in the MCPS-DATA.confirm primitive. When the MCPS-DATA.confirm primitive is used following an initial transmission, the elements of the timestamp report are not all known. However, when the MCPS-DATA.confirm primitive is used following an acknowledgment in a ranging message sequence, all of the elements are known. Likewise, the MCPS-DATA.indication primitive includes a timestamp report, but when the MCPS-DATA.indication primitive is used in response to the initial reception of the first message of a ranging message sequence, not all of the elements of the timestamp report are known. However, when the MCPS-DATA.indication primitive is used following reception of the acknowledgment message, all of the elements are known.

E.1.7.2 Start and stop times in the timestamp report

The timestamp report as both a start time and a stop time. This can appear to be counterintuitive since either start or stop number by itself is useless and that the only real utility for the numbers is in their difference. A different strategy would be to have the PHY do arithmetic on the pair of numbers and present only the difference in the timestamp report. In this standard, the numbers are handled separately by the PHY to allow ranging by PHY implementations having few arithmetic or logic resources. Another reason is to allow an infrastructure node in a one-way ranging environment to issue a new timestamp report for each arriving RFRAME without being concerned about when the “start time” was.

E.1.8 Private ranging

It is important to note that for some applications of this standard, the range information will be the critical deliverable information for the entire system. As such, it is reasonable to protect this information as well as safeguard the integrity of the ranging traffic itself.

E.1.8.1 Simple encryption of the timestamp reports

At the end of the two-way exchange, half of the information necessary for the range computation is in each of two devices. Either half, by itself, is useless to the desired ranging application as well as to any undesired hostile device. When the two halves of the information are brought together, the range is computed by simple arithmetic. The single most critical and effective thing that an application can do to keep hostile devices from learning range information is encrypting the time reports whenever they are being transmitted. There is no problem doing this, as the reports are moved after the time-critical ranging exchange is complete and there is nothing time critical about movement of the timestamp reports.

E.1.8.2 Dynamic preamble selection (DPS)

It is anticipated that typical ranging traffic will take place using the normal channel codes and preambles in regular network use. Therefore, even if the time reports are encrypted and a hostile device is denied knowledge of the ranges, a hostile device can monitor traffic and listen for long preambles. It can then turn on its transmitter, spoof the acknowledgment transmission, and generally disrupt the ranging traffic. This

hostile behavior creates a race between the hostile device and the legitimate responder, but the hostile device can expect to win the race because the legitimate responder will be parsing the MAC header to discover whether an acknowledgment is appropriate before it starts transmitting. To defeat this spoof attack, this standard offers the DPS option, where RDEVs are allowed to move their long preamble RFRAMEs to codes that are altogether different from the codes in normal use. Furthermore, the different preambles are coordinated using encrypted messages so that the hostile device is denied knowledge of the preambles that will be used. And finally, there are no retries allowed with these preambles so that a "jam and spoof the retry" attack will also be defeated.

When the DPS option is invoked by the devices that support it, the hazard is created where if the two-way ranging packet is not received as expected, the devices waiting and listening for special unique length 127 preambles will have become lost. To render this hazard safe, this standard provides for an additional timeout, DPSIndexDuration, which is used whenever DPS is used to ensure that devices at risk of becoming lost are always returned to an interoperable state, as described in 5.1.8.3.

DPS provides no additional ranging capability beyond resistance to attacks by hostile nodes. PHYs that do not implement DPS do not be give up any one-way or two-way ranging capabilities.

E.2 Time-of-arrival estimation from channel sounding

The range between a pair of transmitter and receiver devices can be estimated from the measured multipath profile characterizing the wireless channel between them. The peaks of the profile correspond to the arrivals, the first denoted as the time of arrival. Given τ and knowing that the signal travels at the speed of light c, the range between the two devices can be estimated as $c \times \tau$. The multipath profile appears in the form of a cross-correlation function of the received signal and the transmitted pseudo-random template sequence. In the proposed circuit in Figure E.2 to estimate the delay in an AWGN channel, the receiver first samples the received signal through an ADC and then digitally correlates it with the template to generate a cross-correlation function.

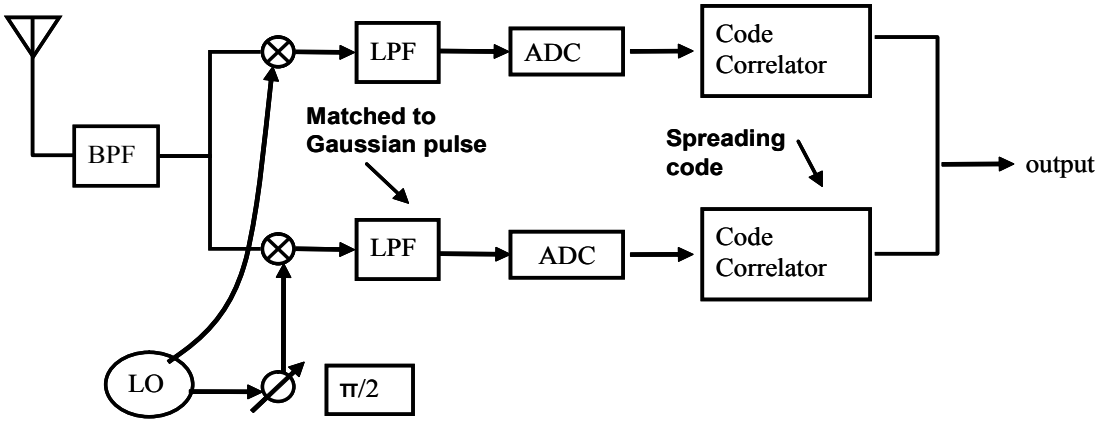


Figure E.2—Circuit to compute multipath profile at receiver

Narrowband communication systems can afford a sampling rate equal to and up to five times the Nyquist rate[12] in the conventional approach to furnish good estimation accuracy. However, such a high sampling rate

[12]This rate is twice the bandwidth of the transmitted signal.

is difficult to implement with UWB devices demanding low cost and low power consumption. Motivated by low sampling rate, Qi and Kohno [B17] propose an approach tailored to such UWB devices that resorts to linear interpolation of the cross-correlation function through a second-order approximation of the maximum likelihood estimate. This estimate exploits both the given autocorrelation function of the template sequence and the given statistical characteristics of the noise, as shown in Figure E.3.

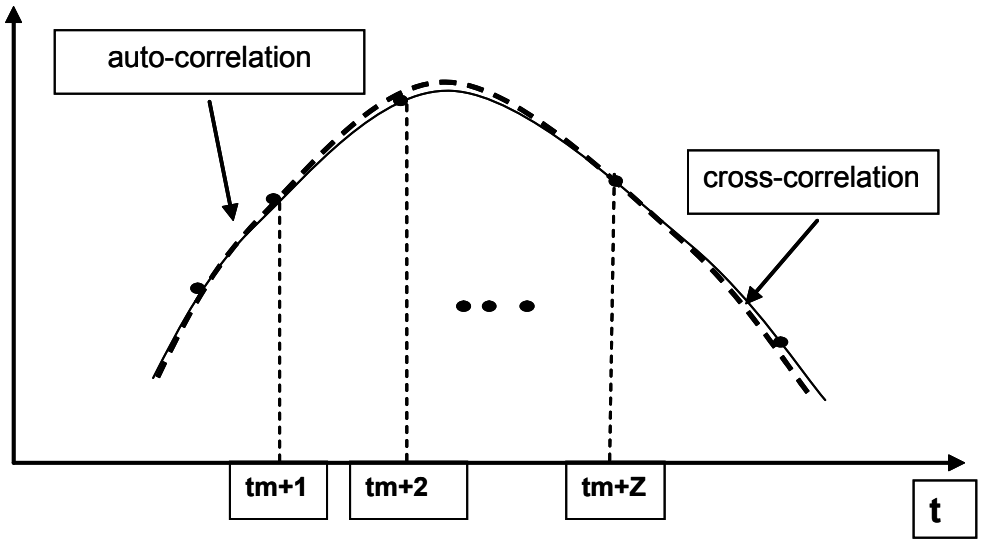


Figure E.3—Autocorrelation and cross-correlation functions for simplified maximum likelihood estimator

Let the three largest adjacent correlation samples be denoted as $h(t_3) = [h(t_1)\ h(t_2)\ h(t_3)]^T$, the corresponding time instants as $t_3 = [t_1\ t_2\ t_3]^T$, and the inverse of the correlation matrix as:

$$W_3 = \begin{bmatrix} g(0) & g(T_s) & g(2T_s) \\ g(T_s) & g(0) & g(T_s) \\ g(2T_s) & g(T_s) & g(0)) \end{bmatrix}$$

where T_s is the sampling period. Let:

$$g(a) = \int s(t) \times s(t - w)dt$$

with $s(t)$ being a ternary pseudo-random sequence of length 31. The delay estimate can be expressed as a simple algebraic solution:

$$\hat{\tau} = \frac{t_3^T \times W_3 \times h(t_3)}{1_3^T \times W_3 \times h(t_3)}$$

where $1_3 = [1\ 1\ 1]^T$. This solution can be shown to be optimal in the sense that the estimate is approaching the theoretical lower limit as the sampling rate grows sufficiently large.

Figure E.4 shows simulation results for a Gaussian UWB pulse of width 500 MHz, a PRF of 30.875 MHz, and an ADC sampling rate of 494 MHz (or 16×PRF). The conventional approach denotes choosing the sample time with largest peak, the interpolation approach denotes simple interpolation without the

autocorrelation function, and the simplified maximum likelihood denotes the proposed approach. Figure E.4 shows a decreasing RMS estimation error with increasing SNR, and Figure E.5 shows the same decreasing error with increasing ADC sampling rate. In both plots, the simplified maximum likelihood outperforms the other two approaches.

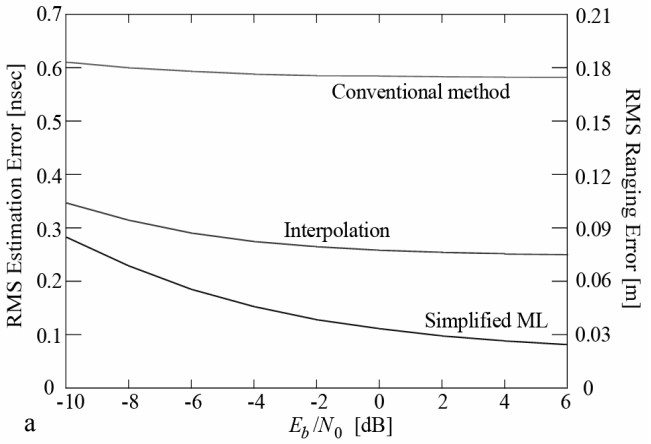


Figure E.4—Performance comparison between the simplified maximum likelihood estimator and other approaches as a function of E_b/N_0

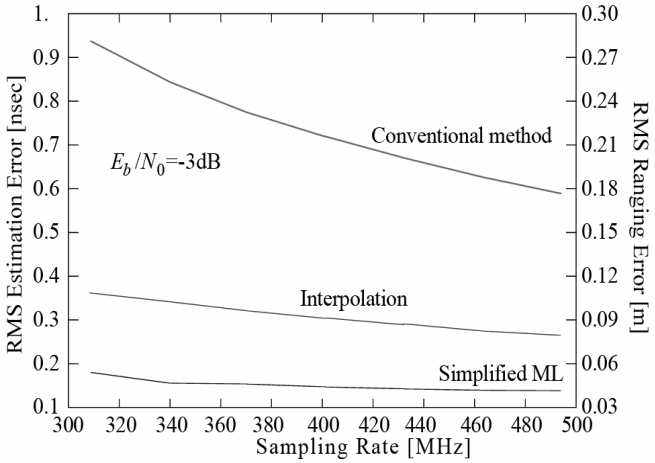


Figure E.5—Performance comparison between the simplified maximum likelihood estimator and other approaches as a function of ADC sampling rate

E.3 Time-of-arrival estimation in non-line-of-sight (NLOS) conditions

In line-of-sight (LOS) conditions, the dominant peak in the cross-correlation function generated at the receiver corresponds to the first arrival. Since its strength is generally much greater than the subsequent peaks in the profile, it proves easy to isolate. However, in NLOS conditions, the first peak corresponding to the direct path is seldom the strongest, attenuated by transmission through walls and other objects; the strongest peak often corresponds to a reflected path whose travel time is greater than the direct path.

In the latter case, a sequential linear cancellation scheme (Qi et al. [B16]) is devised for leading edge detection based on the aforementioned simplified maximum likelihood scheme. It can cope with the accuracy degradation when the first arriving signal component is weak compared to a dominant multipath

component. This scheme reduces to an iterative algorithm. In each step, the amplitude $\hat{A}$ of the present strongest component in the cross-correlation function is estimated based on a sliding delay τ:

$$\hat{A} = \frac{g(\hat{\tau}) \times W_3 \times h(t_3)}{g(\hat{\tau}) \times W_3 \times g(\hat{\tau})}$$

The autocorrelation samples, scaled by the amplitude $\hat{A}$ and the time delay of the strongest component, are subtracted from the cross-correlation samples, effectively eliminating this component as in Figure E.6. Since only the delay of the first arrival is of interest, components with delays greater than τ are subsequently removed in the following iterations, as shown in Figure E.7, until no such components greater than a certain threshold exist. Guvenc and Sahinoglu [B9], [B10] and Lee and Scholtz [B13] cover threshold estimation techniques for UWB systems based on correlation.

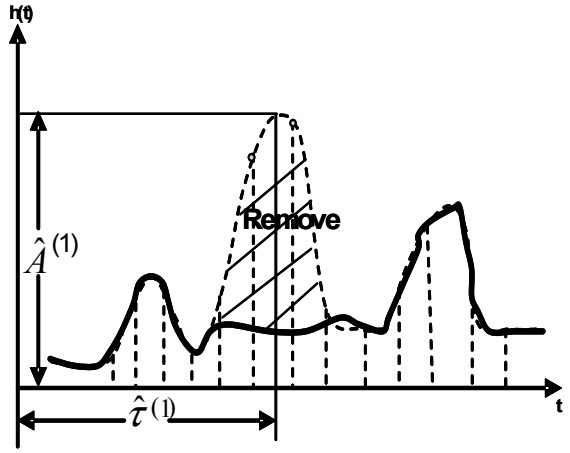


Figure E.6—Leading-edge detection in NLOS conditions

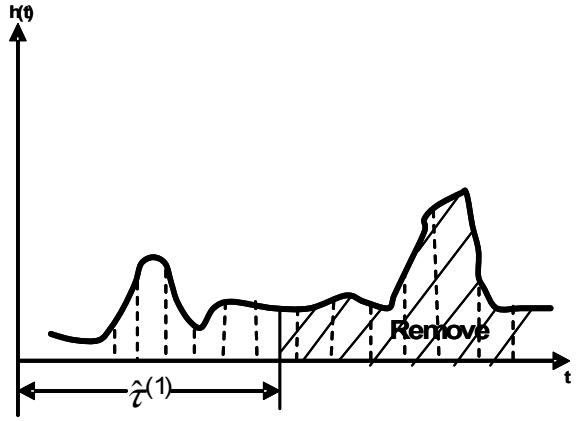


Figure E.7—Leading-edge detection in NLOS conditions after removing delays

E.4 Asynchronous ranging

As described in the previous subclause, the time of arrival is extracted from the cross-correlation function generated at the receiver. It is used for the computation of the time of flight, t_p, defined as the time for

propagation of the signal between the transmitter and receiver. The latter is found through an exchange of messages between the two devices in order to estimate range. The number of messages depends on the backbone structure of the network, which is described in detail in E.5. With clock synchronization between the devices in the network, a single message suffices in one-way ranging to estimate t_p; in the absence of such synchronization, more messages are required. The finite crystal tolerance of the clocks is susceptible to drift and, therefore, has an effect on the number of messages required. This subclause considers two schemes for asynchronous ranging.

E.4.1 Two-way ranging (TWR)

In the absence of clock synchronization between two ranging devices, request device A uses its own clock as a time reference, as depicted in Figure E.8.

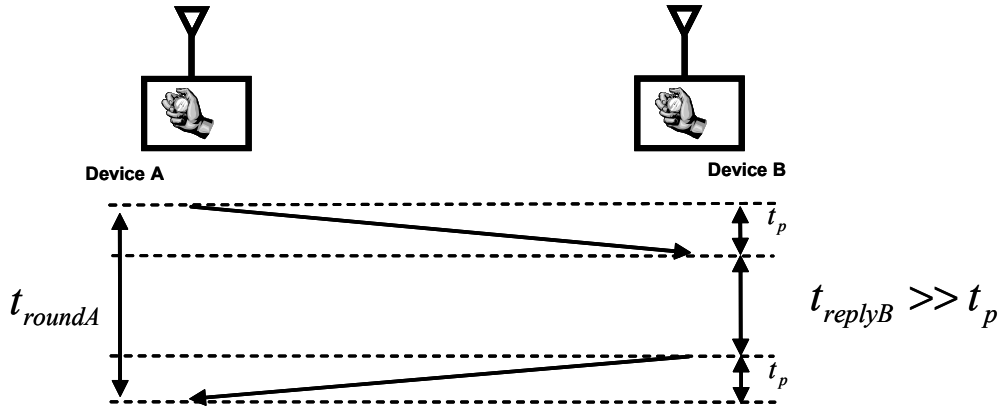


Figure E.8—Exchange of message in two-way ranging

Device A begins the session by sending a range request message to device B. While device B can measure the absolute time of arrival of the message, lacking synchronization with device A, it does not know the time of departure of the message and, therefore, cannot extract t_p. Rather device B waits a time t_{replyB}, known to both devices, to send a request back to device A. Now device A can measure the round-trip time $t_{roundA} = 2t_p + t_{replyB}$ and extract t_p with respect to its own reference time.

Concern should be given to the finite tolerance of the device crystal reference frequency since frequency error introduces error in the measurement of t_p. In reference to Figure E.8, the true value of t_p is computed in terms of the transmitted and received times (denoted by subscripts T and R, respectively) at devices A and B:

$$2t_p = (\underbrace{\tau_{BR} - \tau_{AT}}_{t_p}) + (\underbrace{\tau_{AR} - \tau_{BT}}_{t_p}) = (\tau_{AR} - \tau_{AT}) + (\tau_{BR} - \tau_{BT})$$

Therefore, the estimated value $\hat{t}_p$ follows as:

$$2\hat{t}_p = (\underbrace{\tau_{AR} - \tau_{AT}}_{t_p}) \times (1 + e_A) + (\underbrace{\tau_{BR} - \tau_{BT}}_{t_p}) \times (1 + e_B)$$

where e_A and e_B represent the crystal tolerances of the respective devices expressed in parts per million. Substituting for $\tau_{AR} - \tau_{AT} = 2t_p + t_{replyB}$ and $\tau_{BR} - \tau_{BT} = -t_{replyB}$ and simplifying gives:

$$\hat{t}_p - t_p = \frac{1}{2}(t_{replyB} \times e_A - t_{replyB} \times e_B + 2t_p \times e_A)$$

Note the t_{replyB} is not the turnaround time between the received message from device A and the sent message from device B, but rather includes both the packet duration and this turnaround time. Since the packet duration is on the order of several milliseconds, this duration implies $t_{replyB} >> t_p$ and, therefore:

$$\hat{t}_p - t_p \approx \frac{1}{2} \times t_{replyB} \times (e_A - e_B)$$

Table E.1 presents some typical values for $\hat{t}_p - t_p$ according to the other system parameters.

Table E.1—Typical errors in time-of-flight estimation using TWR

$t_{replyB}/(e_A - e_B)$	2 ppm	20 ppm	40 ppm	80 ppm
100 μs	0.1 ns	1 ns	2 ns	4 ns
5 ms	5 ns	50 ns	100 ns	200 ns

The scope specifies a ranging precision of 1 m; therefore, the estimated t_p needs to lie within 3 ns of the true time of flight given the speed of light. Obviously for the normative packet duration, even with high-quality crystals with tolerance of 2 ppm, the measurement error is greater than the required resolution of the ranging system.

E.4.2 Symmetric double-sided two-way ranging (SDS-TWR)

In order to compensate for the shortcomings of simple two-way ranging, Hach [B11] proposes an additional message exchange in the ranging session to reduce the effect of the finite crystal tolerances of the devices. Figure E.9 shows the message exchange.

The diagram shows that the round-trip times t_{roundA} and t_{roundB} can be expressed in terms of t_p nd the respective t_{replyA} and t_{replyB} as follows:

$$t_{roundA} = 2t_p + t_{replyB}$$

$$t_{roundB} = 2t_p + t_{replyA}$$

Combining the two equations allows for isolating the true value of t_p as:

$$4t_p = t_{roundA} - t_{replyA} + t_{roundB} - t_{replyB}$$

and the estimated $\hat{t}_p$ follows by introducing the finite crystal tolerance e_A and e_B:

$$4\hat{t}_p = (t_{roundA} - t_{replyA}) \times (1 + e_A) + (t_{roundB} - t_{replyB}) \times (1 + e_B)$$

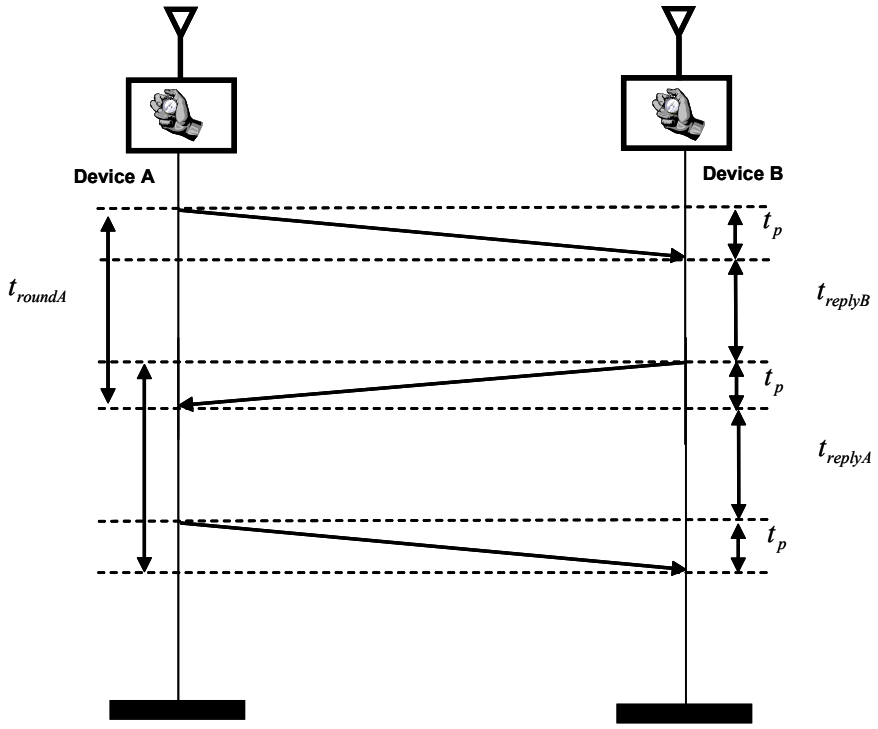


Figure E.9—Exchange of message in SDS-TWR

Without loss of generality, replacing t_{replyA} and t_{replyB} with:

$$t_{replyA} = t_{reply}$$

$$t_{replyB} = t_{reply} + \Delta_{reply}$$

reduces it to:

$$\hat{t}_p - t_p = \frac{1}{2} \times t_p \times (e_A + e_B) + \frac{1}{4} \times \Delta_{reply} \times (e_A - e_B)$$

Assuming that $t_p << \Delta_{reply}$, the equation simplifies further to:

$$\hat{t}_p - t_p \approx \frac{1}{4} \times \Delta_{reply} \times (e_A - e_B)$$

Table E.2 shows the typical errors in the SDS-TWR time-of-flight estimation versus frequency tolerance.

The extra message in the SDS-TWR accommodates a much smaller error margin even with low-quality crystals of 80 ppm.

E.5 Location estimation from range data

The ranging capabilities of the devices can be used to estimate their locations through network collaboration. A device wishing to determine its location gathers at least three or four ranges to neighboring

Table E.2—Typical errors in time-of-flight estimation using SDS-TWR

$\Delta_{reply}/(e_A-e_B)$ (ms)	2 ppm (ns)	20 ppm (ns)	40 ppm (ns)	80 ppm (ns)
1	0.0005	0.005	0.01	0.02
10	0.005	0.05	0.1	0.2
100	0.05	0.5	1	2

devices with known location in a two- or three-dimensional network. The technique used to triangulate the estimated ranges to an estimated location of the device depends largely on the network topology and communication protocols. The subclause considers the two main approaches to simple triangulation.

E.5.1 Time of arrival

Consider device A initiating a ranging session by sending a request message to device B. Device B can estimate the time of arrival τ_A from the message through one-way ranging. If the two clocks are synchronized to the same time reference, device B can also extract the time of departure τ_D included in the message by device A and hence compute the time of flight $t_p = \tau_D - \tau_A$. In practice, it proves difficult or inefficient to establish and/or maintain synchronization between two mobile devices; therefore, a network lacking a wired backbone will need to resort to asynchronous ranging described in E.4.

The time-of-arrival technique for triangulation of ranges applies to the general network lacking synchronization between devices and/or a priori organization. The technique assumes that three (or more) ranges $c \times t_1$, $c \times t_2$, and $c \times t_3$ are gathered from anchor devices $i = 1, 2, 3$, respectively, with known locations (x_i, y_i), as in Figure E.10.

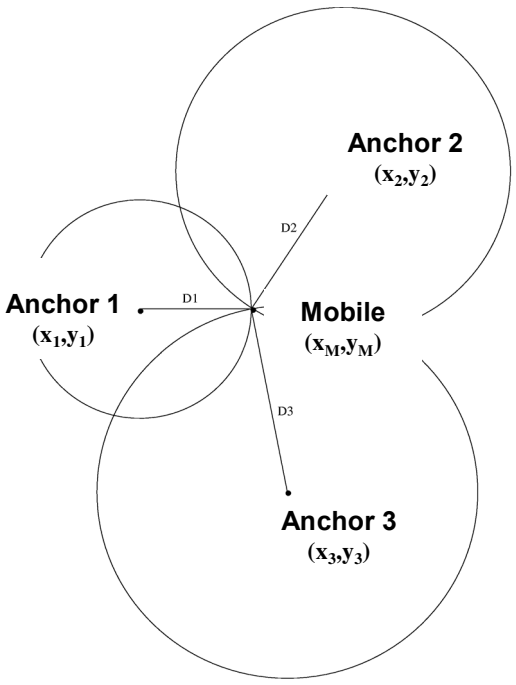


Figure E.10—Time-of-arrival triangulation of ranges to determine location

$$c \times t_1 = \sqrt{(x_1 - x_M)^2 + (y_1 - y_M)^2}$$
$$c \times t_2 = \sqrt{(x_2 - x_M)^2 + (y_2 - y_M)^2}$$
$$c \times t_3 = \sqrt{(x_3 - x_M)^2 + (y_3 - y_M)^2}$$

Solving the set of equations translates to finding the intersection of the three circles (or spheres in three dimensions), yielding the unknown coordinates of the mobile device (x_M, y_M).

Time of arrival is appealing due to its application to the general network architecture; however, the associate asynchronous ranging requires two or more messages per ranging session. This requirement may potentially increase the network traffic considerably.

E.5.2 Time difference of arrival

A pre-installed network can be configured so that the mobile devices within a deployment area can maintain connectivity with at least three anchor devices positioned at known locations and connected through wire to maintain clock synchronization. Clock synchronization enables one-way ranging in conjunction with the time-difference-of-arrival technique, as opposed to time of arrival. In order to carry out a range request, the devices in the network classified as stationary anchor devices and mobile devices operate in one of the two following modes: Mode 1 or Mode 2.

E.5.2.1 Mode 1

The synchronized anchor nodes jointly send range requests to the mobile node at the same time instant; therefore, the number of messages equals the number of anchors. Although the mobile node lacks synchronization with the three anchors indexed through $i = 1, 2, 3$, it can still measure the time difference of arrivals $t_{32} = \tau_3 - \tau_2$ and $t_{31} = \tau_3 - \tau_1$ within its own time reference, where τ_i denotes the arrival time of the message from anchor i at the mobile M. Figure E.11 depicts the locations of anchors i and respective locations (x_i, y_i). Solving the following equations translates to finding the intersection of two hyperbolae and yields the unknown location (x_M, y_M) of the mobile device. The burden of the calculation lies on the mobile device, which may have limited resources with respect to the anchor devices due to its smaller dimensions to preserve battery life.

E.5.2.2 Mode 2

This mode operates in a similar fashion to Mode 1; however, the mobile device initiates the range request by sending a single message received by all the neighboring anchor nodes. The request message arrives at the anchor nodes at different times of arrival measured individually. These values are either distributed throughout the wired backbone or sent to a central controller to compute the time difference of arrivals t_{32} and t_{31} and in turn the location of the mobile device. This mode offers the advantage of reducing the wireless network traffic to just one message per request and also does not place the burden of computation on the mobile node with potentially limited resources.

E.6 Network location algorithms

In the basic approach to location estimation described in E.5, a mobile device gathers range measurements from a number of anchor devices with known locations and triangulates these ranges to a single point (or area) through simple geometrical relationships. This approach assumes that the mobile device receives at least three or four estimates in a two- or three-dimensional coordinate system.

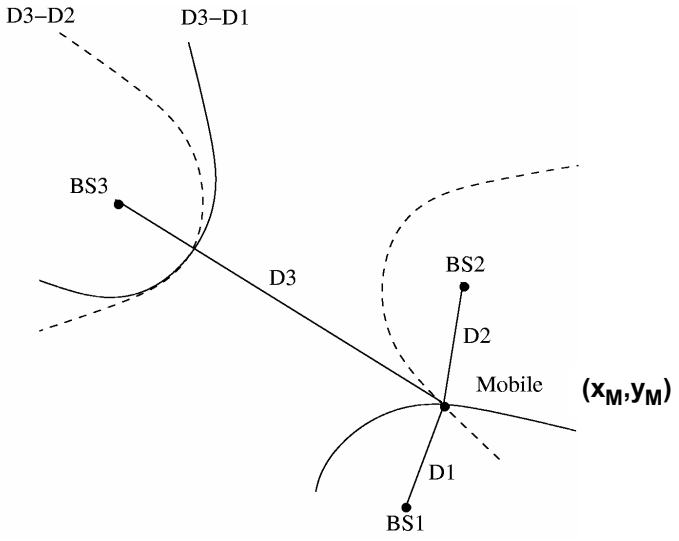


Figure E.11—Time-difference-of-arrival triangulation of ranges to determine location

Interest in dense sensor networks due to falling price and reduced size has motivated research in network location algorithms in recent years, where the number of mobile devices vastly outnumbers the number of fixed stations. Consider a rapidly deployable network whose nodes are scattered about an area of interest and self-organize in an ad hoc fashion to determine their locations through simple messaging and ranging. Most of the sensors in such networks lack connectivity to fixed stations. Rather, the high sensor-to-sensor connectivity allows them to infer their locations from the locations of other sensors that do have connectivity to the fixed stations. This class of algorithms to process large amounts of range data is commonly known as *network location algorithms*. These algorithms can render good location accuracy despite significant errors in range estimates between sensors.

This subclause provides a survey of the benchmark network location algorithms developed in recent years and divides them into three main classes: ad hoc algorithms, centralized algorithms, and convex optimization algorithms.

E.6.1 Ad hoc algorithms

Network location algorithms arose from the need to locate nodes in ad hoc networks characterized by high mobility and dynamic architecture, with nodes joining and leaving the network at random times. Here greater importance is placed on continuously updating location in a distributed manner rather than furnishing precision. Niculescu and Nath [B14] contributed to this pioneering effort. Their work describes several algorithms to this end based on the range measurements available to the nodes. The strength of these algorithms lies in their simplicity and in turn their applications to ad hoc networks. In general they foster reduced complexity for larger networks.

The first algorithm known as *DV-Hop* generates location based on mere connectivity quantified as the minimum number of hops between two nodes, or the minimum-hop distance. Hence the nodes in the network lack any ranging capabilities. The network is categorized into two types of nodes: *anchor nodes* whose locations are known and *sensor nodes* whose locations are unknown. Rather than estimate range between each other, the nodes simply determine the number of hops between each other. Given the ground-truth ranges between the anchors through simple messaging of their known locations through the ad hoc network, each anchor node i computes a factor f_i, which is simply the sum of the ground-truth ranges between all the anchors in the network divided by the sum of the respective minimum hops.

Consider the network in Figure E.12. Anchor L1 computes factor f_1 by summing the ground-truth ranges from L2 (40 m) and L3 (100 m), divided by the sum of the minimum hops 6 and 2, or $f_1 = (40 + 100)/(2 + 6)$ m. Anchor L2 and L3 compute their factors f_2 and f_3 in the same manner as $f_2 = (40 + 75)/(2 + 5)$ m and $f_2 = (75 + 100)/(5 + 6)$ m, respectively, and then distribute them to all the sensors in the network. Ultimately the sensor node A, pictured in Figure E.13, finds its location from its three closest neighboring anchor nodes; however, rather than through the triangulation of three measured ranges R_1, R_2, and R_3 to them, respectively, it finds its location through DV-ranges $R_1 = 3 \times f_1$, $R_2 = 2 \times f_2$, and $R_3 = 3 \times f_3$ given through the minimum hops and the factors.

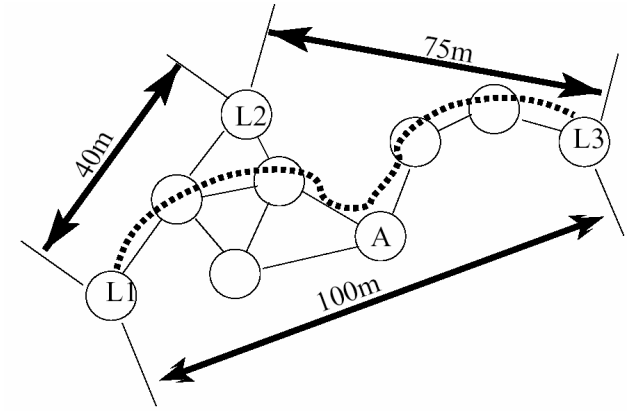


Figure E.12—Network for comparison between the simplified maximum likelihood estimator and other approaches

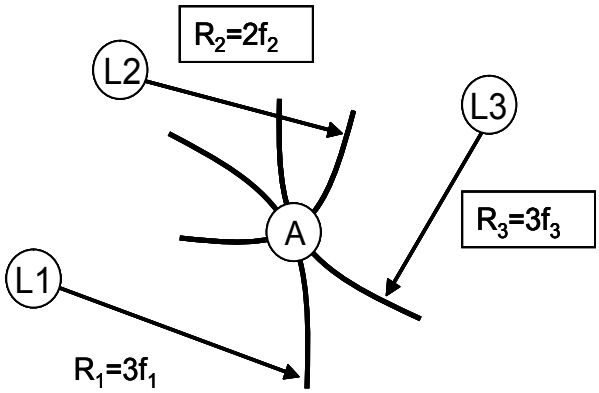


Figure E.13—Performance comparison between the simplified maximum likelihood estimator and other approaches

In the same paper, the authors present an alternative algorithm known as *DV-Distance* as an adaptation of the ad hoc positioning system to accommodate nodes with ranging technology for enhanced precision without compromising simplicity. Rather than scaling the ground-truth distance by the minimum-hop distance between two anchors in computing the factors of each anchor node, it is scaled by the sum of the measured distances on the multihop path between two anchors. It directly follows in the triangulation step that the distances between the unknown sensor and the three closest anchor nodes are scaled by the measured distances on the multihop path between the two.

Saverese et al. [B19] present another algorithm designed for ad hoc networks, which also incorporates range estimates. Knowing its coordinates (x_1, y_1), anchor L1 orients a local coordinate system pictured in Figure E.14 in the direction of an arbitrary neighbor, e.g., L2. Given the measured range r_{12}, the coordinates $(x_2, y_2) = (x_1 + r_{12}, y_1)$ of sensor 2 are easily computed in the same coordinate system. Given further the

range estimates r_{13} between anchor L1 and sensor L3 and r_{23} between sensors L2 and L3, the coordinates of $(x_3, y_3) = (x_2 + dx, \text{y2} + dy)$ are computed through:

$$dx = \frac{{r_{12}}^2 + {r_{13}}^2 + {r_{23}}^2}{2 \times r_{12}}$$

$$dy = \sqrt{{r_{12}}^2 - dx^2}$$

In this manner, anchor L1 propagates the coordinates of its successive neighbors throughout the network as an initialization step. The other anchor nodes independently do likewise.

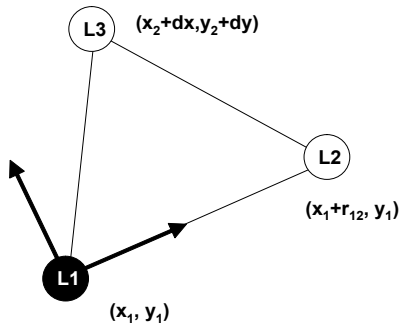


Figure E.14—Ad hoc network location algorithm proposed by Savarese et al.

In propagation through the individual neighbors of the n_A anchor nodes in the network, node 2 receives n_A estimates of its coordinates. It can then refine its estimate $(\hat{x}_2, \hat{y}_2)$ as the equally weighted average of all the accumulated estimates. In sequence, sensor 3 refines its estimate through the propagating of the coordinates from anchor L1 and sensor 2 as in the initialization step; however, it replaces (x_2, y_2) with $(\hat{x}_2, \hat{y}_2)$. Propagation then continues and eventually converges to an acceptable solution after a number of iterations, although the authors provide no proof. In the network simulations, the algorithm furnishes a location error of about 5% even with range errors up to 50%.

E.6.2 Centralized algorithms

Centralized algorithms gather all the data available from the network to process it collectively. As expected, these algorithms in general render better results than the ad hoc algorithms, which consider only local data and process it independently of the data available to other parts of the network. The drawback of the former involves designating a central controller, a condition which may be unsuitable for some applications. Even so, the dynamic links of nodes in motion may require rapid updating; alternatively, relaying information across a large network sanctions the centralized processing of obsolete data at the controller, and this condition limits scalability. It is worth noting that most of these algorithms have distributed versions, which, however, compromise the quality of the results. This subclause considers the architecture of two centralized algorithms commonly referred to in literature.

Many centralized algorithms estimate the locations of the unknown nodes by minimizing an objective function such as in Savvides et al. [B20]. A popular function, as used here, is the least-squared sum of the residuals; the residual is defined as the difference between a measured range and the range estimated through the algorithm. The minimization is performed through Kalman filtering, which ultimately finds only a local minimum. In fact, an initialization step is required to estimate the positions of the nodes from the measured ranges and the anchor nodes. Consider sensor node C in Figure E.15 as an example. The simplistic initialization step finds a bounding rectangle for it as the intersection of the individual bounding squares of

each anchor node (here A and B) to the sensor: the length of half the side of the square is the sum of the measured distances on the multihop between the anchor and the sensor:

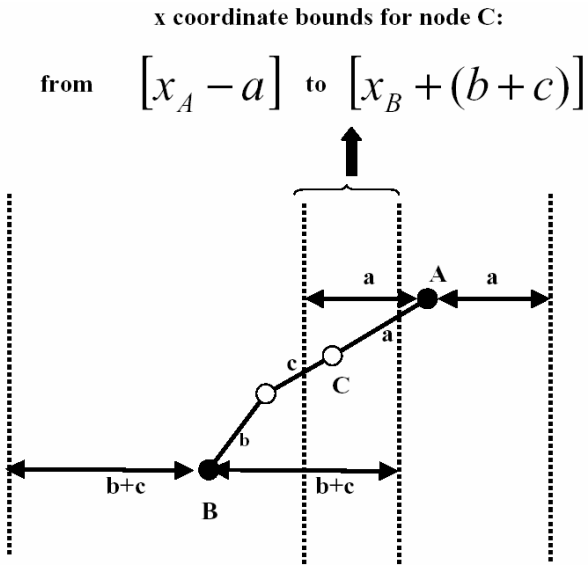


Figure E.15—Autocorrelation and cross-correlation functions for the maximum likelihood estimator

Shang et al. [B21] have designed another centralized algorithm to minimize the least-squared sum of the residuals, but use a more powerful technique called *multidimensional scaling* to find the global maximum. As in Savvides, the initial step consists of computing the minimum-hop distances between all nodes in the network, but here they are stored in a range matrix R having the following structure:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots \\ r_{21} & r_{22} & r_{23} & \cdots \\ r_{31} & r_{32} & r_{33} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

The minimum-hop distance between two nodes is just the sum of the estimated ranges between them.[13] Performing the singular value decomposition of R^2 yields a matrix of eigenvectors in V and respective squared eigenvalues on the diagonal matrix A. The relative coordinates of the nodes are subsequently reconstructed as $X = V\sqrt{A}$. These coordinates are then scaled, translated, and rotated to fit the anchor nodes.

E.6.3 Convex optimization algorithms

The most powerful class of network location algorithms places the geometrical constraints of the physical world on the nodes while minimizing an objective function similar to the one in E.6.2. The advantage of the technique over the ones in E.6.2 lies in its ability to achieve the global maximum independent of any initial estimates. Doherty et al. [B5] first proposed the technique in defining the following optimization problem with quadratic constraints to minimize a linear objective function:

[13]The estimated ranges are actually normalized to fit the framework of the multidimensional scaling.

$$\min \sum |\alpha_{ij}|$$

such that:

$$\left\| \hat{\bar{x}}_i - \hat{\bar{x}}_j \right\|_2 = d_{ij}, \ \forall (i,j) \in N$$
$$\left\| \hat{\bar{x}}_i - \hat{\bar{x}}_j \right\|_2 \geq R, \quad \forall (i,j) \notin \bar{N}$$

where

$d_{ij} = \hat{d}_{ij} + \alpha_{ij}$	is the estimated range resultant of the algorithm
$\hat{d}_{ij}$	is the measured range
α_{ij}	is the residual between the two
N	is the set containing the index of neighboring nodes
$\bar{N}$	is the set of non-neighboring nodes
R	is the radio range, i.e., the range at which a node loses connectivity with another node in the network

Since many of the constraints are nonconvex, the paper relaxes the problem simply by removing these constraints. The corresponding solution not only offers less precision, but also forces the sensors to lie within the convex hull of the anchors. An alternative approach proposed by Biswas et al. [B3] relaxes the problem to a semi-definite program that yields an average and standard variation for the positions of the unknown nodes.

Rather than relaxing the problem from the original, Gentile [B8] directly applies linear constraints given through the triangle inequality:

$$\min \sum |\alpha_{ij}|$$

such that:

$$\left.\begin{array}{l} d_{ij} + d_{jk} \geq d_{ik} \\ d_{ij} + d_{ik} \geq d_{jk} \\ d_{jk} + d_{ik} \geq d_{ij} \end{array}\right\}, \left\{\begin{array}{l} \forall (i,j) \in N \\ \forall (j,k) \in N \\ \forall (i,k) \in N \end{array}\right.$$

The original convex constraints necessitate no relaxation and hence render a much tighter solution than the other two approaches mentioned. A distributed version of the algorithm yields the same results as the centralized version with no compromise in achieving an optimal result (Gentile [B7]).

In order to substantiate the effectiveness of the network location algorithms, Table E.3 presents the results from Gentile [B8]. The simulation platform consists of a network with 50 sensor nodes uniformly distributed in a unit area. The number of anchor nodes varies as a parameter {3, 5, 7} in the table and R as {0.20, 0.25, 0.30}. The noise parameter controls the percentage of Gaussian-distributed noise perturbing the

measured radio range from the ground-truth range and varies as {0.0, 0.1, 0.2, 0.3}. The table shows that despite the measured range errors up to 30%, the location errors yielded by the algorithm can lie on the order of only 5%.

Table E.3—Location results according to varying network parameters

noise	*R* = 0.20			*R* = 0.25			*R* = 0.30		
	3	5	7	3	5	7	3	5	7
0.0	0.043	0.042	0.041	0.007	0.006	0.006	< 1e–6	< 1e–6	< 1e–6
0.1	0.075	0.064	0.064	0.053	0.044	0.027	0.045	0.036	0.025
0.2	0.085	0.080	0.068	0.077	0.065	0.049	0.057	0.057	0.046
0.3	0.107	0.088	0.087	0.095	0.083	0.077	0.077	0.074	0.046

E.6.4 Location estimation using multipath delays

For location estimation in a multipath environment, the conventional approach is based on leading-edge detection, where the time-of-arrival estimate of the first arriving signal is taken as the distance between the transmitter and the receiver of interest up to a constant and the delays of other multipath signals are completely ignored. This approach works well when the first arrival signal is sufficiently strong and via a LOS propagation path. However, in a typical UWB channel, the first arrival signal is usually weak, e.g., 6 dB lower than a dominant multipath component, and can be subject to NLOS propagation. Hence the conventional approach can cause severe degradation of the positioning accuracy. To address this problem, one method is to utilize time-of-arrival estimates of multipath components in addition to the first arriving signals for location estimation. Although subject to NLOS propagation, the second and later arriving signals should also carry information regarding the position of interest. Hence the method incorporating the multipath delays can improve the positioning accuracy under certain conditions.

For simplicity, consider that a mobile node is synchronized with B anchor nodes, whose locations $\{(x_b, y_b), b = 1, 2, \ldots, B)$ are known. Each anchor node receives radio signals transmitted from the mobile node via multipath propagation. A received signal at the b^{th} anchor node is expressed as:

$$r_b(t) = \sum_{i=1}^{N_b} A_{bi} \times s(t - \tau_{bi}) + n_b(t)$$

where τ_{bi} is the delay of the i-th multipath component, given by:

$$\tau_{bi} = \frac{1}{c}\sqrt{(x - x_b)^2 + (y - y_b)^2} + l_{bi}$$

which consists of the LOS delay corresponding to the distance between the mobile node and the anchor node, and the NLOS induced path length error l_{bi}. The quantity $l_b = (l_{b1}, l_{b2}, \ldots, l_{bN})^{\mathrm{T}}$ is usually modeled as a multivariate random variable, which can be determined by field experiments or theoretical models. Noise $n_b(t)$'s are independent white Gaussian processes, and A_{bi} is the signal amplitude. Estimation of the multipath delays yields:

$$\hat{\tau}_{bi} = \tau_{bi} + \xi_{bi}$$

where $\xi_b = (\xi_{b1}, \xi_{b2}, \ldots, \xi_{bN})^T$ is a multivariate Gaussian random variable with zero mean and an explicit covariance matrix F_ξ. Based on the equation for $r_b(t)$ and the probability density functions of l_b and ξ_b, location estimation of (x, y) can be formulated as the maximum a priori estimation. It is shown that the positioning accuracy enhancement depends on two principal factors, the strength of multipath components and the variance of the NLOS induced errors. In certain situations, significant accuracy improvement, e.g., greater than 50%, can be obtained. The limitation of this approach is that its computation complexity is higher than the conventional approach. The exact formula of accuracy improvement and detailed discussion on this approach can be found in Qi et al. [B16].

Annex F

(informative)

Example UWB PHY transmit data frame encoding

F.1 Channel used in the example

This annex provides one example of how the PHY would encode a short sample PSDU received from the MAC sublayer. In the example, the PHY transmits at 850 kb/s on channel 3 using preamble code index 6. The annex shows how the data are changed by the PHY encoding steps that eventually lead to bursts of pulses for transmission.

F.2 Encoding progression

F.2.1 Transmit PSDU

In this example, the MAC sublayer presents a sample PSDU to the PHY via the PHY service access point (SAP). The sample that will be encoded is the following 17-octet PSDU:

UWB welcomes IEEE

converted from ASCII to decimal as follows:

85 87 66 32 119 101 108 99 111 109 101 115 32 73 69 69 69

which, in hexadecimal, is as follows:

55 57 42 20 77 65 6C 63 6F 6D 65 73 20 49 45 45 45

Note that the MAC sublayer would not usually present such a PSDU to the PHY because it does not contain a valid cyclic redundancy check (CRC). The purpose of this annex is to provide an example to help the reader to understand the PHY encoding process, not the MAC sublayer.

F.2.2 PSDU bits

For the rest of this annex, the ternary +, –, 0 notation will be used where + represents a one, 0 represents a zero, and – represents a –1.

The sample PSDU is converted to binary, LSB first and starting with the first bit in time first as follows:

+0+0+0+0+++0+0+00+0000+000000+00+++0+++0+0+00++000++0++0++000++0++++0++0+0++0++0+
0+00++0++00+++000000+00+00+00+0+0+000+0+0+000+0+0+000+0

F.2.3 Reed-Solomon encoded bits

Reed-Solomon encoding is then applied to these bits as described in 14.3.3.1 to give the following bits:

+0+0+0+0+++0+0+00+0000+000000+00+++0+++0+0+00++000++0++0++000++0++++0++0+0++0++0+
0+00++0++00+++000000+00+00+00+0+0+000+0+0+000+0+0+000+000++0++000+++00+0++++0+00+
++++++00+++0+++00+0+00

F.2.4 Convolutional encoder input bits

The next step is to prepend the PHR to this Reed-Solomon encoded data for input to the convolutional coder. Because of the data rate and length of PSDU, the PHR is as follows in Table F.1.

Table F.1—Example PHR

Rt1	Rt0	L6	L5	L4	L3	L2	L1	L0	Rng	Ext	Pr1	Pr0	C5	C4	C3	C2	C1	C0
0	1	0	0	1	0	0	0	1	0	0	0	1	1	1	0	0	1	1

Prepending this to the PSDU gives the following bit stream:

0+00+000+000+++00+++0+0+0+0+++0+0+00+0000+000000+00+++0+++0+0+00++000++0++0++000+
+0++++0++0+0++0++0+0+00++0++00+++000000+00+00+00+0+0+000+0+0+000+0+0+000+000++0++
000+++00+0++++0+00+++++++00+++0+++00+0+00

F.2.5 Convolutional encoder output bits

Two tail bits are appended to the input bits, and they are input to the convolutional encoder. The convolutional encoder produces two types of output bit: position encoding bits, denoted g_0, and sign or parity bits, denoted g_1.

For this example, the position bits g_0 will be simply a delayed version of the previous bit stream with a zero tail bit or as follows:

00+00+000+000+++00+++0+0+0+0+++0+0+00+0000+000000+00+++0+++0+0+00++000++0++0++000
++0++++0++0+0++0++0+0+00++0++00+++000000+00+00+00+0+0+000+0+0+000+0+0+000+000++0+
+000+++00+0++++0+00+++++++00+++0+++00+0+000

The sign bits g_1 will be as follows:

0+0++0+0+0+0++0++++0+0000000+0+0000++0+00+0+0000+0+++0+0+0+0000+++++0+++0++0+++0+
++0+00+0++000++0++0000++++0+++++0++0000+0++0++0++00000+0+00000+0+00000+0+0+0+++0+
++0++0+++00+00+00+++00000++++0+0+0+++000+00

F.2.6 Scrambler output bits

The scrambler is now enlisted to help with the encoding process. For this complex channel, the scrambler initializer, starting at s_{-15} and going to s_{-1}, is as follows:

1 1 1 0 0 0 1 0 1 1 0 1 1 0 1

Since there are 205 of each type of convolutional bits, there will be 205 symbols. Each symbol produces a burst of length 16 so the scrambler provides 3280 bits, which are as follows:

00+00+++0++0+++0++0+00++0++00++0+++0+0++0+0+0++00++++0++++++0+0+000++00000+++++0
0+0+0000+0000+0++++000++000+++000+00+0+00+00+00++0++++0++0++0+0++000++0++0++++0+0
0+0++0++000+++0+++0++0+00+00++00++0+++0++0+0+0+0++00++0+++++++++0+0+0++0000000+++
+++0+000000+00000+++00000++0000+00+0000+0+000++0++000++++00+0++0+00+000+0+++0+++0
++00+++00++00++0+0+00+0+0+0+0+++++0+++++++++0000++00000000+000+0+0000000++00++++0
00000+0+0+000+00000++++++00++0000+00000+0+0+000++0000++++++00+0+000+00000+0++++00+
+0000+++000+0+0+000+00+00++++++00++0++0+00000+0+0++0+++0000++++0++00+000+0000++0
+0++00++000+0++++0+0+0+00+++000++++++++0+00+00+000000+++0++0++00000+00++0++0+0000+
+0+0++0+++000+0++++0++00+00+++000++0+0++0+00+00+0++++0+++0++0+++000++00++0++00+0
0+0+0+0++0+0++0+++++++0++++0++000000++000++0+00000+0+00+0+++0000++++0+++00+000+00
0++00+0++00++00+0+0+++0+0+0+0+++++00++++++++0000+0+0000000+000++++000000++00+000+
00000+0+0++00++0000+++++0+0+0+000+0000++++++++00++000+000000+0+0+00++00000++++++0+
0+0000+00000+++++000++0000+0000+00+0+000++000++0++++00+0+00+0+++0+++0+00+
++000++00+++0+00+00+0+0+00+++0++0++++++0+00++0++00000+++0+0++0+0000+00++++0+++00
0++0+000++00+00+0+++00+0+0++0+++00+0+++++0++00+0+++0000++0+0+++00+000+0++++00+0+
+00+++000+0+++0+0+00+00+++00+++++0++0+00+0+0000++0+++0++++000+0++00++000+00+++0+
0+0+00++0+00+++++++0+0+++0+000000++++00+++00000+000+0+00+0000++00++++0++000+0+0+0
00++0+00++++++00+0+++0+00000+0+++00+++0000+++00+0+00+000+00+0++++0++00++0+++000++
0+0+0++00+00+0++++++0+0++0+++00000++++0++00+0000+000++0+0++000++00+0++++0+00+0+0+
++000+++0+++++00+00+00++0000+0++0++0+0+000+++0++0+++++00+00++0++0000+0++0+0++0+0
00+++0++++0+++00+00++000++00+0++0+0+00+0+0+++0+++++0+++++00++0000++0000+0+0+000+
0+000++++++00++++00+00000+0+000+0++0000++++00+++0+000+000+0+00+++00++00++++0+00+0
+0+0+000+++0++++++++00+00++0000000+0++0+0+000000+++0+++++00000+00++0000+0000++0+0
+000++000+0+++++00+0+00+++0000+0++++0+00+000+++000+++0++00+00+00+00++0+0++0++0++0
+0++++0++0++0++++000++0++0++000+00+0++0++0+00++0+++0++0+++0+0++00++0++00++++0+0+
0++0+0+000++++++0+++++00+00000++0000+0++0000+0+000+++0+000++++00+00+++00+000+0++0
+00+0++00+++0+++0+++0+0+00++00++00+++++0+0+0+0+0+0000++++++++++000+0000000000+00
++000000000++0+0+00000000+0+++++0000000+++0000+000000+00+000++00000++0++00+0+0000+
0++0+0++++000+++0++++000+00+00++000+00++0++0+0+00++0+0++0+++++0+0++++0++0000++++
000++0+000+000+00+0+++00++00++0+++00+0+0+0+0++00+0+++++++++0+0+++00000000++++00+0
000000+000+0++000000++00+++0+00000+0+0+00+++0000++++++0+00+000+00000+++0++00++0000
+00++0+0+0+000++0+0+++++++00+0++++000000+0+++000+00000+++00+00++0000+00+0++0+0+00
0++0+++0+++++00+0++00++0000+0+++0+0+0+000+++00+++++++00+00+0+000000+0++0++++00000
+++0++000+0000+00++0+00++000++0+0+++0+0+00+0++++00++++0+++000+0+0000++00+00++++0
00+0+0++0+000+00++++++0+++00++0+0000++00+0+0+++000+0+0++++++00+00++++++0000+0++0+000
0+000+++0+++000++00+00++00+00+0+0++0+0+0++0+++++0++++++0++0000++00000++0+000+0+00
00+0+++00++++000+++00+0+000+00+00+0++++00++0++0+++000+0+0++0++00+00++++++0++0+0++
0+0000++0++++0+++000+0++000++00+00+++0+00+0+0++0+00+++0+++++0+++0+00++0000++00++
+0+0+000+0+0+00++++++00++++++0+0000+0+00000+++000++++0000+00+00+000+000++0++0++00+
+00+0++0++0+0+0+0+++0++0++++++++00++0++0000000+0+0++0+000000+++++0+++00000+0000++
00+0000++000+0+0++000+0+00+++++0+00++++0+0000+++0

F.2.7 Ternary output symbols

These convolutional encoder and scrambler outputs are used to generate the ternary pulses. In this example, each symbol consists of 512 chips. During each symbol, the 512 chips are silent except for a train of 16 pulses. Table F.2 and Table F.3 give the chip position of the first pulse and show the signs of the burst of pulses, starting with the first in time. The chip positions in this example can range from 0, the first chip position, to 511, the final chip position. Each symbol is also numbered from 0 to 204.

Table F.2—Example ternary output symbols 1

Symbol number	Burst Chip Position	Burst	Symbol number	Burst Chip Position	Burst
0	64	++−++−−−+−−+−−−+	51	16	+−−++−−+−+−+++−+
1	48	++−+−−++−++−−++−	52	288	−+−+−+++++−−++++
2	368	−−−+−+−−+−+−+−−+	53	368	−−−−++++−+−+++++
3	96	−++++−++++++−+−+	54	320	−−+−−−+++++−−−−−
4	0	−−−++−−−−−+++++−	55	48	−−++−+++−+++++−+
5	288	+−+−++++−++++−+−	56	336	+−++−−++−−−++++
6	112	+++−−−++−−−+++−−	57	336	−+−+−+−+++−++++−
7	32	+−++−+−++−++−++−	58	368	++++++−−++−−−+−−
8	80	+−++++−++−++−+−+	59	0	++++−+−+−++−−+++
9	272	−+++−−+−−+−−−+−	60	320	++−−−−−+−+−++++
10	64	−−+−++−++−−−+++−	61	16	−+++++−−−−−++−−
11	112	−−−+−−+−++−++−−+	62	256	++++−++++−++−+−+
12	96	−++−+++−++−+−+−+	63	64	−−++−−−++−++++−−
13	352	−++−−++−++++++++	64	80	+−+−−+−++−−−+−++
14	288	+−+−+−−++++++−−	65	304	++−+++−+−−+++−−−
15	368	++++−+−−−−−+−−−	66	304	++−−+++−+−−+−−+−
16	64	−−+++−−−−−++−−−−	67	80	+−+−−+++−++−++++
17	16	+−−+−−−−+−+−−−++	68	48	−−+−++−−+−−+++++
18	352	−++−−−++++−−+−++	69	112	+++−+−++−+−−−−+−
19	288	+−++−+++−+−−−+−−	70	352	−++++−+++−−−++−+
20	336	+−++−−+++−−++−−+	71	256	−−−++−−+−−+−+++−
21	80	−+−+−++−+−+−+−+−	72	32	+−+−+−−+−−−++−+−
22	368	−−−−+−−−−−−−−++	73	368	++++−++−−+−+++−−
23	64	++−−++++++++−+++	74	320	−−++−+−+++−−+−−−
24	336	−+−+++++++−−++−−	75	80	−+−−−−++−+−−++−−
25	48	−−++++++−+−+−+++	76	272	+−−−+−+++−+−+−−+
26	272	−+++++−−−−−−++−−	77	320	−−+++−−+++++−++−
27	0	++++−+++++−+−+−+	78	16	+−−+−+−−−−++−+++
28	320	−−++−−−−−++++++−−	79	96	+−−−−+++−+−−++−−
29	336	−+−+++−+++++−+−−	80	0	−−−+−−+++−+−+−+−
30	304	++−−++−−−−+++−−−	81	352	−++−+−−++++++++
31	80	−+−+−+++−++−++−−	82	352	−+++−+−−−−−−++++
32	368	−−−−++−−+−−+−+++	83	64	++−−−+++++−++++
33	64	++−+−+−−+−−−++++	84	272	+−−+−−−−++−−++++
34	368	−−−−−+−−++−+++−+	85	352	+−−+++−+−+−+++−−
35	0	−−−++−+−++−−++−−	86	288	+−++−−−−−−++−+−−
36	32	−+−++++−+−+−+−−+	87	336	+−+−−−−+−+++−−+
37	304	−−+++−−−−−−+−++	88	48	−−++++−−−++−+−++
38	16	+−−+−−−−−−+++−++	89	272	+−−−+−−+−++++−++
39	96	+−−+++++−++−−+−−	90	320	−−++−+++−−−++−+−
40	32	+−+++++−+−+−−+−−	91	80	−+−−++−++−+−−−−−
41	16	+−−−+−++++−++−−+	92	336	−+−+−−+−−−+++++−
42	320	++−−−+++−−+−+−−+	93	112	−−−+−−++−++++−++
43	16	+−−+−−+−++++−+++	94	352	−++−+−++−−−++−−+
44	96	+−−+−−−+++−−++−−	95	352	−++++−+−−+−+−+++
45	96	+−−++−++−+−+−+−−	96	0	+++−−−+−−−−−++−+
46	32	+−+−−+−−−−−−+−−−	97	288	−+−−++−−−−+−++−+
47	48	−−+−−++++++−−+++	98	336	+−+−+−−−+++−++−+
48	48	++−+−−−−+−+−−+−	99	112	−−−−++−++−−+−−++
49	368	−−−+++−−−−+−−−+	100	320	++−+−−+−+−−+−+++
50	32	−+−−−+−−−++−−+−+	101	112	−−−+−−−−+−−−++−+

Table F.3—Example ternary output symbols 2

Symbol number	Burst Chip Position	Burst	Symbol number	Burst Chip Position	Burst
102	352	+−−+++−−++−+−−+−	153	80	+−+−−−−+−+−+−−+
103	32	−+−−+−+−+++−++++	154	304	−−++++−−−−−+−++
104	80	+−+++++−−++−−−−+	155	16	+−−−+−−−−+++−++
105	272	+−−−−+−+−+−−−+−+	156	64	++−−++++−++−−+−+
106	256	−−−++++++−−++++−	157	80	+−+−−−++−+−+++++
107	32	+−+++++−+−+++−+−	158	304	++−−+−++++−−−−−
108	272	+−−−−++++−−+++−+	159	336	+−+++−−−+−−−−++
109	256	−−−+−−−+−+−−+++−	160	16	−++−++−−++++−++−
110	96	−++−−++++−+−−+−+	161	352	−++−+−+−−−++−+++
111	32	−+−+−−−+++−+++++	162	352	−+++++−−+−++−−++
112	368	+++−−+−−++−−−−−	163	0	−−−−+−+++−+−+−+−
113	288	+−+−−+−+−++++++−	164	64	++−−−++−−−−−−++
114	304	++−+++++−−−−+−−	165	16	+−−+−+−−−−−+−++
115	48	++−−−−+−−−−++−+−	166	352	−++++−−−−−+++−++
116	16	−+++−−+++−+−−−−−	167	256	+++−++++−++−−+−+
117	64	++−+−++−−−++++−+	168	352	−++−−−++−+−+++−+
118	112	−−−−+−++−+++−−−+	169	32	−+−−+−++++−−++++
119	64	++−−−+−−++−++−++	170	80	+−+++−−−+−+−−−−+
120	16	+−−++−+−++−++−++	171	272	−++−++−−−−+++−+−
121	288	+−+−−−−+−−+−−+−−	172	96	+−−+−+++−++−−−−−
122	48	++−−−++−++−++−−−	173	352	−+++−−++−+−−−−++
123	16	+−−+−++−++−+−−++	174	320	++−+−+−−−+++−+−+
124	352	+−−−+−−+−−−+−+−−	175	368	−−−−−++−++−−−−−+
125	64	−−++−++−−++++−+−	176	256	−−−+−++−+−−−−+−−
126	80	+−++−+−+−−−++++	177	96	+−−−+−−−+++−−++−
127	336	−+−−−−−++−++++−	178	320	++−−++−++−+−+−−+
128	16	+−−−−+−++−−−−+−+	179	80	+−+−++−++++−+++
129	0	−−−+++−+−−−++++−	180	112	+++−++−−−−++−−−−
130	288	+−++−−−++−+++−+−	181	352	−++−+−−−+−+−−−−+
131	80	−+−++−+−−++−−−+−	182	352	+−−−++−−−−+++−−−
132	304	−−+−−−+−+−++−−++	183	320	++−+−+++−++−++−+
133	48	−−++−−−−+−+−+−+	184	368	−−−−++−−+−−+−−−+
134	336	−+−++++−−−−−−−−−	185	320	++−+−+−−+−−++−++
135	48	++−−−+−−−−−−−−−	186	368	−−−−−+−−+−+−−+−+
136	16	−++−−+++++++++−−	187	256	−−−++−++++−+++−−
137	32	−+−+−−−−−−−+−++	188	32	−+−++−−−++−+−−+
138	368	−−−+++++++−−−+++	189	48	++−+−−+−+−++−+−−
139	32	+−++++++−++−+++−	190	368	+++−+++++−+++−+−
140	272	−++++++−+−−++−+−	191	352	+−−++++−−++−−−+−
141	0	++++−+−−+−+−−−−+	192	288	−+−−−+−+−+−−++++
142	320	++−−−+−−−−+++−++	193	16	−++−−−−−+−++++−
143	16	+−−++−−−+−−++−++	194	288	−+−−−−−+++−−−+++
144	32	+−+−+++−+−+−−+−−	195	272	−++++−++−++−+++−
145	112	+++−+−++++−++−−−	196	256	−−−++−++−++−−++−
146	352	+−−−−+++−−+−+++−	197	32	−+−++−++−+−+−+−+
147	0	+++−++−+−−−++−−+	198	48	++−++−++++++++−−
148	352	+−−+−−−++−+−+−+−	199	304	−−+−−+++++++−+−+
149	96	+−−++−+−−−−−−−−−	200	48	−−+−++++++−−−−−+
150	288	+−+−−−++++++++−−	201	368	−−−+++++−++++−−+
151	48	++−−+−−−−−−+−−−	202	32	−+−−−−++−−−+−+−+
152	80	−+−−++++++−−++−−	203	16	−+++−+−++−−−−−+−
			204	64	++−−−−+−++++−−−+

Annex G

(informative)

MPSK PHY

G.1 General

The following is an informative translation of the MPSK PHY, which is one of the co-alternative PHYs in the CWPAN specification. Another co-alternative PHY in the CWPAN specification is the O-QPSK PHY specified in Clause 10. The CWPAN specification is NITS/CWPAN Part 15.4, Chinese standard for the Wireless Medium Access Control (MAC) and the Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (GB/T 15629.15-2010). The CWPAN specification also defines operation in the 314–316 MHz and the 430–434 MHz bands, but operation in those bands is not described in this annex.

G.2 780 MHz band data rates

The data rate of the MPSK PHY is 250 kb/s when operating in the 780 MHz band.

G.3 Modulation and spreading

The MPSK PHY employs a 16-ary orthogonal modulation technique. During each data symbol period, four information bits are used to select 1 of 16 orthogonal PN sequences to be transmitted. The PN sequences for successive data symbols are concatenated, and the aggregate chip phase is modulated onto the carrier using PSK.

G.3.1 Reference modulator diagram

The functional block diagram in Figure G.1 is provided as a reference for specifying the 780 MHz band MPSK PHY modulation and spreading functions. The number in each block refers to the subclause that describes that function. Each bit in the PPDU is processed through the bit-to-symbol mapping, symbol-to-chip mapping, pre-processing, and modulation functions in octet-wise order, beginning with the Preamble field and ending with the last octet of the PSDU. Within each octet, the LSB, b_0, is processed first and the MSB, b_7, is processed last.

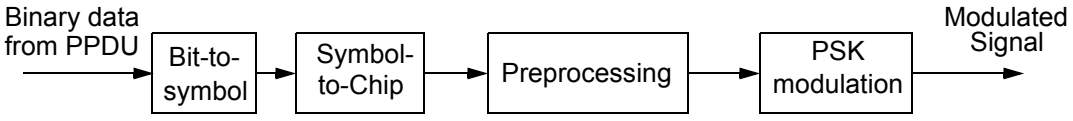


Figure G.1—Modulation and spreading functions

G.3.2 Bit-to-symbol mapping

All binary data contained in the PPDU is encoded using the modulation and spreading functions shown in Figure G.1. This subclause describes how binary information is mapped into data symbols.

The 4 LSBs (b_0, b_1, b_2, b_3) of each octet is mapped into one data symbol, and the 4 MSBs (b_4, b_5, b_6, b_7) of each octet is mapped into the next data symbol. Each octet of the PPDU is processed through the modulation and spreading functions, as illustrated in Figure G.1, sequentially, beginning with the Preamble field and ending with the last octet of the PSDU. Within each octet, the least significant symbol (b_0, b_1, b_2, b_3) is processed first, and the most significant symbol (b_4, b_5, b_6, b_7) is processed second.

G.3.3 Symbol-to-chip mapping

Each data symbol is mapped into a 16-chip PN sequence as specified in Table G.1.

Table G.1—Symbol-to-chip mapping for MPSK

Data symbol	Chip phases (c0 c1 … c14 c15)
0	$0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}$
1	$\frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}$
2	$\frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}$
3	$\frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi$
4	$\pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}$
5	$-\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}$
6	$\frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}$
7	$-\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0$
8	$0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}$
9	$-\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}$
10	$\frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}$
11	$-\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi$

Table G.1—Symbol-to-chip mapping for MPSK *(continued)*

Data symbol	Chip phases (c0 c1 … c14 c15)
12	$\pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}$
13	$\frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}\ \frac{\pi}{16}$
14	$\frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0\ \frac{\pi}{4}$
15	$\frac{\pi}{4}\ \frac{\pi}{16}\ \frac{9\pi}{16}\ \pi\ -\frac{7\pi}{16}\ \frac{\pi}{4}\ -\frac{15\pi}{16}\ 0\ -\frac{15\pi}{16}\ \frac{\pi}{4}\ -\frac{7\pi}{16}\ \pi\ \frac{9\pi}{16}\ \frac{\pi}{4}\ \frac{\pi}{16}\ 0$

G.3.4 Pre-processing

The chip sequence that the chip phase is mapped to consists of some DC value. To mitigate the DC effect, this pre-processing block subtracts each chip by the following value:

$$A_{\mathrm{DC}} = \left(\frac{1}{4}\right)\exp\left(j\frac{\pi}{4}\right)$$

G.3.5 PSK modulation

The chip phases representing each data symbol are modulated onto the carrier using PSK with raised cosine pulse shaping. Because each data symbol is represented by a 16-chip sequence, the chip rate is 16 times the symbol rate.

G.3.6 Pulse shape

The raised cosine pulse shape with roll-off factor of $r = 0.5$ is used to represent each baseband chip and is given by:

$$p(t) = \begin{cases} \dfrac{\sin(\pi t/T_c)}{\pi t/T_c} \times \dfrac{\cos(r\pi t/T_c)}{1-4r^2t^2/T_c^2}, & t \neq 0 \\ 1, & t = 0 \end{cases}$$

Given the discrete-time sequence of consecutive complex-valued chip samples, given by:

$$\{\exp(jc_k)\}_{k=-\infty}^{\infty}$$

the continuous-time pulse shaped complex baseband signal is given by:

$$y(t) = \sum_{k=-\infty}^{\infty} \exp(jc_k)p(t - kT_c)$$

where

T_c is the inverse of the chip rate.

G.3.7 Chip transmission order

During each symbol period, the least significant chip phase, c_0, is transmitted first, and the most significant chip phase, c_{15}, is transmitted last.

G.4 MPSK PHY RF requirements

This subclause describes the MPSK PHY radio performance requirements.

G.4.1 Transmit power

The transmit power (EIRP) is limited to 10 mW.

G.4.2 Operating frequency range

The 780 MHz MPSK PHY operates in the 779–787 MHz frequency band.

G.4.3 Transmit PSD mask

When operating in the 780 MHz band, the transmitted spectral products are less than the limits specified in Table G.2. For both relative and absolute limits, average spectral power is measured using a 100 kHz resolution bandwidth. For the relative limit, the reference level is the highest average spectral power measured within ±600 kHz of the carrier frequency f_c.

Table G.2—780 MHz band MPSK PHY transmit PSD limits

Frequency	Relative limit	Absolute limit
$\|f - f_c\| > 1.2$ MHz	–20 dB	–20 dBm

G.4.4 Symbol rate

The MPSK PHY symbol rate is 62.5 ksymbol/s when operating in the 780 MHz band with an accuracy of ±40 ppm.

G.4.5 Receiver sensitivity

A compliant device is capable of achieving a sensitivity of –85 dBm or better under the conditions defined in 8.1.7.

G.4.6 Receiver interference rejection

The minimum receiver interference rejection levels are given in Table G.3. The adjacent channel is one on either side of the desired channel that is closest in frequency to the desired channel, and the alternate channel is one more removed from the adjacent channel. For example, when channel 2 is the desired channel, channel 1 and channel 3 are the adjacent channels, and channel 0 is an alternate channel.

Table G.3—Minimum receiver interference rejection requirements for 780 MHz MPSK PHY

Adjacent channel rejection	Alternate channel rejection
0 dB	30 dB

The adjacent channel rejection is measured as follows: the desired signal is a compliant MPSK PHY signal, as defined by G.3, of pseudo-random data. The desired signal is input to the receiver at a level 3 dB greater than the maximum allowed receiver sensitivity given in G.4.5.

In either the adjacent or the alternate channel, a compliant signal, as defined by G.3, is input at the relative level specified in Table G.3. The test is performed for only one interfering signal at a time. The receiver will meet the error rate criteria defined in G.4.5 under these conditions.

Annex H

(informative)

Considerations for the 950 MHz band

H.1 General

This annex describes the way in which the IEEE 802.15.4 MAC may be configured and used to meet the requirements of the Japanese regulation for 950 MHz, as described in ARIB STD-T96 [B2] and English translation of ARIB STD-T96 [B6].

H.2 Listen before talk (LBT) considerations

The regulation requires that a device uses listen before talk (LBT) prior to transmission if the duty cycle of transmission exceeds 0.1%. There is also a requirement that a device does not continuously transmit. The maximum continuous transmission time and the duty cycle of transmission are dependent on the LBT duration. The detailed operation is described in ARIB STD-T96 [B2].

The regulation applies to a complete product. MAC PIB attributes are provided to permit a higher layer to control both the duty cycle and the listen before talk functionality.

The PIB attribute *macTxTotalDuration* is provided to allow a higher layer to control the transmission duty cycle of operation. The value represents the total number of symbols transmitted since last set to zero. The higher layer may read the value at any time, and it may set the value (typically to zero). This provides a mechanism for the higher layer to calculate the actual transmission time and hence the percentage transmission time over any arbitrary period.

The PIB attributes *macTxControlActiveDuration* and *macTxControlPauseDuration* permit a higher layer to control both the duration for which a device may transmit and the duration of the pause period, i.e., the time during which the MAC will pause to allow other devices access to the channel. These values are dependent on the transmission power and channel.