INTERNATIONAL STANDARD

ISO/IEC 18000-63

First edition
2013-01-15

Information technology — Radio frequency identification for item management —

Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C

Technologies de l'information — Identification par radiofréquence (RFID) pour la gestion d'objets —

Partie 63: Paramètres de communications d'une interface radio entre 860 MHz et 960 MHz, Type C

Reference number
ISO/IEC 18000-63:2013(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 18000-63 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

ISO/IEC 18000 consists of the following parts, under the general title *Information technology — Radio frequency identification for item management*:

— *Part 1: Reference architecture and definition of parameters to be standardized*

— *Part 2: Parameters for air interface communications below 135 kHz*

— *Part 3: Parameters for air interface communications at 13,56 MHz*

— *Part 4: Parameters for air interface communications at 2,45 GHz*

— *Part 6: Parameters for air interface communications at 860 MHz to 960 MHz General*

— *Part 61: Parameters for air interface communications at 860 MHz to 960 MHz Type A*

— *Part 62: Parameters for air interface communications at 860 MHz to 960 MHz Type B*

— *Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C*

— *Part 64: Parameters for air interface communications at 860 MHz to 960 MHz Type D*

— *Part 7: Parameters for active air interface communications at 433 MHz*

Introduction

This part of ISO/IEC 18000 describes a passive backscatter radio frequency identification (RFID) system that supports the following system capabilities:

- identification and communication with multiple tags in the field;
- selection of a subgroup of tags for identification or with which to communicate;
- reading from and writing to or rewriting data many times to individual tags;
- user-controlled permanently lockable memory;
- data integrity protection;
- Interrogator-to-tag communications link with error detection;
- tag-to-Interrogator communications link with error detection;
- support for both passive back-scatter tags with or without batteries.

This part of ISO/IEC 18000 specifies the physical and logical requirements for a passive-backscatter, RFID system operating in the 860 MHz to 960 MHz frequency range. The system comprises Interrogators, also known as readers, and tags, also known as labels.

An Interrogator transmits information to a tag by modulating an RF signal in the 860 MHz to 960 MHz frequency range. The tag receives both information and operating energy from this RF signal. Passive tags are those which receive all of their operating energy from the Interrogator's RF waveform. If tags maintain a battery then they may operate using some passive principles; however, they do not necessarily get all their operating energy from the Interrogator's RF waveform.

An Interrogator receives information from a tag by transmitting a continuous-wave (CW) RF signal to the tag; the tag responds by modulating the reflection coefficient of its antenna, thereby backscattering an information signal to the Interrogator. The system is Interrogator-Talks-First (ITF), meaning that a tag modulates its antenna reflection coefficient with an information signal only after being directed to do so by an Interrogator.

Interrogators and tags are not required to talk simultaneously; rather, communications are half-duplex, meaning that Interrogators talk and tags listen, or vice versa.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning radio frequency identification technology.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC.

Information on the declared patents may be obtained from:

Contact details	
Patent Holder:	
Legal Name	Atmel Automotive GmbH
Contact for license application:	
Name & Department	Leo Merken, Legal Department, ATMEL Corporation
Address	2325 Orchard Parkway
Address	San Jose, CA 95131 USA
Tel.	+1 408 436 4251
Fax	+1 408 487 2615
E-mail	leo.merken@atmel.com
URL (optional)	
Patent Holder:	
Legal Name	CISC Semiconductor Design+Consulting GmbH
Contact for license application:	
Name & Department	Markus Pistauer, CEO
Address	Lakeside B07
Address	9020 Klagenfurt, Austria
Tel.	+43(463) 508 808
Fax	+43(463) 508 808-18
E-mail	m.pistauer@cisc.at
URL (optional)	www.cisc.at

Patent holder:	
ETRI (Electronics Telecommunication Reseach Institute)	
Contact for license application:	
Name & Department:	Min-Sheo Choi, Intellectual Property Management Team
Address:	138 Gajeongno, Yuseong-gu
Address:	Daejeon, 305-700, Korea
Tel.	+82-42-860-0756
Fax	+82-42-860-3831
E-mail	choims@etri.re.kr
URL (optional)	www.etri.re.kr
Patent Holder:	
Legal Name	Impinj, Inc.
Contact for license application:	
Name & Department	Chris Diorio, CTO
Address	701 N. 34th Street, Suite 300
Address	Seattle, WA 98103, USA
Tel.	+1.206 834 1115
Fax	+1.206 517.5262
E-mail	diorio@impinj.com
URL (optional)	www.impinj.com

Patent Holder:	
Legal Name:	Magellan Technology Pty. Limited
Contact for license application:	
Name & Department:	Ms Jean Angus
Address:	65 Johnston St
Address:	Annandale, NSW 2038, Australia
Tel.	+61 2 9562 9800
Fax	+61 2 9518 7620
E-mail:	license@magellan-technology.com
URL (optional):	www.magellan-technology.com
Patent Holder:	
Legal Name	NXP B.V.
Contact for license application:	
Name & Department	Aaron Waxler – IP Licensing & Claims
Address	411 East Plumeria,
Address	San Jose, CA 95134-1924, USA
Tel.	+1 914 860-4296
Fax	
E-mail	Aaron.Waxler@nxp.com
URL (optional)	

Patent Holder:	
Legal Name	TAGSYS SAS
Contact for license application:	
Name & Department	Mr. Alain Fanet President
Address	785 Voie Antiope, TI Athélia 3
Address	F-13600 La Ciotat
Tel.	+33 332188900
Fax	+33 332188900
E-mail	alain.fanet@tagsysrfid.com
URL (optional)	www.tagsysrfid.com

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

The latest information on IP that may be applicable to this part of ISO/IEC 18000 can be found at www.iso.org/patents

Information technology — Radio frequency identification for item management —

Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C

1 Scope

This part of ISO/IEC 18000 defines the air interface for radio frequency identification (RFID) devices operating in the 860 MHz to 960 MHz Industrial, Scientific, and Medical (ISM) band used in item management applications. It provides a common technical specification for RFID devices that can be used by ISO committees developing RFID application standards. This part of ISO/IEC 18000 is intended to allow for compatibility and to encourage inter-operability of products for the growing RFID market in the international marketplace. It defines the forward and return link parameters for technical attributes including, but not limited to, operating frequency, operating channel accuracy, occupied channel bandwidth, maximum effective isotropic radiated power (EIRP), spurious emissions, modulation, duty cycle, data coding, bit rate, bit rate accuracy, bit transmission order, and, where appropriate, operating channels, frequency hop rate, hop sequence, spreading sequence, and chip rate. It further defines the communications protocol used in the air interface.

This part of ISO/IEC 18000 specifies the physical and logical requirements for a passive-backscatter, Interrogator-Talks-First (ITF) systems. The system comprises Interrogators, also known as readers, and tags, also known as labels. An Interrogator receives information from a tag by transmitting a continuous-wave (CW) RF signal to the tag; the tag responds by modulating the reflection coefficient of its antenna, thereby backscattering an information signal to the Interrogator. The system is ITF, meaning that a tag modulates its antenna reflection coefficient with an information signal only after being directed to do so by an Interrogator.

In detail, this part of ISO/IEC 18000 contains Type C.

Type C uses PIE in the forward link and a random slotted collision-arbitration algorithm.

This part of ISO/IEC 18000 specifies

— physical interactions (the signalling layer of the communication link) between Interrogators and tags,

— Interrogator and tag operating procedures and commands,

— the collision arbitration scheme used to identify a specific tag in a multiple-tag environment.

2 Conformance

To claim conformance with this part of ISO/IEC 18000, an Interrogator or tag shall comply with all relevant clauses of this part of ISO/IEC 18000, except those marked as “optional”. The Interrogator or tag shall also operate within local radio regulations, which can further restrict operation.

Relevant conformance test methods are provided in ISO/IEC TR 18047-6.

Conformance can also require a license from the owner of any intellectual property utilized by said device.

2.1 Interrogator conformance and obligations

To conform to this part of ISO/IEC 18000, an Interrogator shall

— support Type C

— implement the mandatory commands defined in this part of ISO/IEC 18000;

— modulate/transmit and receive/demodulate a sufficient set of the electrical signals defined in the signalling layer of this part of ISO/IEC 18000 to communicate with conformant tags; and

— operate within the applicable local regulations.

To conform to this part of ISO/IEC 18000, an Interrogator may

— implement any subset of the optional commands defined in this part of ISO/IEC 18000, and

— implement any proprietary and/or custom commands in conformance with this part of ISO/IEC 18000.

To conform to this part of ISO/IEC 18000, the Interrogator shall not

— implement any command that conflicts with this part of ISO/IEC 18000 or any of the parts 61, 62 and 64, or

— require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 18000.

2.2 Tag conformance and obligations

To conform to this part of ISO/IEC 18000, a tag shall:

— support Type C;

— operate over the frequency range from 860 MHz to 960 MHz, inclusive;

— implement the mandatory commands defined in this part of ISO/IEC 18000 for the supported types;

— modulate a backscatter signal only after receiving the requisite command from an Interrogator; and

— conform to local radio regulations.

To conform to this part of ISO/IEC 18000, a tag may

— implement any subset of the optional commands defined in this part of ISO/IEC 18000; and

— implement any proprietary and/or custom commands as defined in Clauses 7.

To conform to this part of ISO/IEC 18000, a tag shall not:

— implement any command that conflicts with this part of ISO/IEC 18000 or any of the parts 61, 62 and 64;

— require the use of an optional, proprietary, or custom command to meet the requirements of this part of ISO/IEC 18000; or

— modulate a backscatter signal unless commanded to do so by an Interrogator using the signalling layer defined in this part of ISO/IEC 18000.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7816-6, *Identification cards — Integrated circuit cards — Part 6: Interindustry data elements for interchange*

ISO/IEC 15961, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: application interface*

ISO/IEC 15962, *Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions*

ISO/IEC 15963, *Information technology — Radio frequency identification for item management — Unique identification for RF tags*

ISO/IEC 19762 (all parts), *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

EPCglobal Tag Data Standards version 1.3 and above, EPCglobal Inc.

4 Terms and definitions, symbols and abbreviated terms

4.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762 (all parts) and the following apply.

4.1.1
battery saver mode
battery saving functionality based on low power threshold detection with optional duty cycling

4.1.2
collision arbitration loop
algorithm used to prepare for and handle a dialogue between an Interrogator and a tag

NOTE This is also known as collision arbitration.

4.1.3
cover-coding
method by which an Interrogator obscures information that it is transmitting to a tag by requesting a random number from the tag, then performing a bit-wise EXOR of the data or password with the received random number, and, finally, transmitting the cover-coded (also called ciphertext) string to the tag, which uncovers the data or password by performing a bit-wise EXOR of the received cover-coded string with the original random number

4.1.4
Dense-Interrogator environment
operating environment within which most or all of the available channels are occupied by active Interrogators

EXAMPLE 25 active Interrogators operating in 25 available channels.

4.1.5
EPCglobal Application
application whose MB01 data structure includes a “0” bit in bit 17h and denotes an acceptance of EPCglobal standards and policies

4.1.6
ISO Application
application whose MB01 data structure includes a “1” bit in bit 17h and where bits 18h through 1Fh encode an application family identifier as defined in ISO/IEC 15961-3

4.1.7
physical layer
data coding and modulation waveforms used in Interrogator-to-tag and tag-to-Interrogator signalling

4.1.8
slotted random anticollision
collision-arbitration algorithm where tags load a random (or pseudo-random) number into a slot counter, decrement this slot counter based on Interrogator commands, and reply to the Interrogator when their slot counter reaches zero

4.1.9
Activation
waking up a tag from the **hibernate** (4.1.19) state

4.1.10
battery assistance
battery support for radio frequency communication

4.1.11
battery assisted mode
working mode of battery assisted tags with non-empty battery

4.1.12
Full Function Sensors
sensors that can be user configurable and reset, can produce a detailed set of sensor measurement observations, and can be connected to the RFID tag in a variety of ways

4.1.13
passive mode
working mode of passive tags or battery assisted tags with battery drained below a manufacturer-specific threshold

4.1.14
PacketCRC
16-bit cyclic-redundancy check (CRC) code that a tag may dynamically calculate over its PC, optional XPC, and UII and backscatter during inventory

4.1.15
Simple Sensors
sensors that are factory programmed and not user configurable, producing a single output observation such as a fail/pass condition or simple measurement of a particular sensor activity

4.1.16
Simple Sensor functionality
functionality whereby sensors provide a valid Simple Sensor data address and transmit Simple Sensor data subsequent to the UII as part of the reply to the *ACK* command (Type C) or as part of the TagMsg (Type D)

4.1.17
StoredCRC
16-bit cyclic-redundancy check (CRC) code that a tag calculates over its StoredPC and UII and stores in UII memory at powerup, and can backscatter during inventory

4.1.18
recommissioning
significant altering of a tag's functionality and/or memory contents, as commanded by an Interrogator, typically in response to a change in the tag's usage model or purpose

4.1.19
hibernate
state of energy saving when the device is not required to be used

4.1.20
PacketPC
protocol-control information that a tag with nonzero-valued XI dynamically calculates and backscatters during inventory

4.1.21
StoredPC
protocol-control information stored in UII memory that a tag with zero-valued XI backscatters during inventory

4.1.22
tag energized
tag in the **ready**, **arbitrate, reply, acknowledged, open,** or **secured** state

4.1.23
tag not energized
tag not in the **ready**, **arbitrate, reply, acknowledged, open,** or **secured** state

5 Symbols, abbreviated terms and notations

5.1 Symbols

BAP	battery assisted passive
BLF	backscatter-link frequency (BLF = $1/T_{pri}$ = DR/TRcal)
DR	divide ratio
FT	frequency tolerance
INACT_T	inactivity threshold
M	number of subcarrier cycles per symbol
M_h	RF signal envelope ripple (overshoot)
M_{hh}	FHSS signal envelope ripple (overshoot)
M_{hl}	FHSS signal envelope ripple (undershoot)
M_{hs}	FHSS signal level during a hop
M_l	RF signal envelope ripple (undershoot)
M_s	RF signal level when OFF
Q	slot-count parameter. *Q* is an integer in the range (0,15)

R=>T	Interrogator-to-tag (reader-to-tag)
RTcal	Interrogator-to-tag (reader-to-tag) calibration symbol
T=>R	tag-to-Interrogator (tag-to-reader)
T_1	time from Interrogator transmission to tag response
T_2	time from tag response to Interrogator transmission
T_3	time an Interrogator waits, after T_1, before it issues another command
T_4	minimum time between Interrogator commands
T_f or $T_{f,10\text{-}90\%}$	RF signal envelope fall time
T_{hf}	FHSS signal envelope fall time
T_{hr}	FHSS signal envelope rise time
T_{hs}	time for an FHSS signal to settle to within a specified percentage of its final value
T_{pri}	backscatter-link pulse-repetition interval (T_{pri} = 1/BLF = TRcal/DR)
T_r or $T_{r,10\text{-}90\%}$	RF signal envelope rise time
T_s	time for an RF signal to settle to within a specified percentage of its final value
Tari	reference time interval for a data-0 in Interrogator-to-tag signalling
Tf	fall time
Tr	rise time
TRcal	tag-to-Interrogator (tag-to-reader) calibration symbol
UII	unique item identifier
x_{fp}	floating-point value
$xxxx_2$	binary notation
$xxxx_h$	hexadecimal notation

5.2 Abbreviated terms

AC	activation code
AFI	application family identifier
AMSK	Activation Mask
ASF	application sub family
ASIC	application specific integrated circuit
BAM	battery assisted mode
BAT	battery assisted tag
ciphertext	information that is cover-coded
CRC	cyclic redundancy check
CRC-16	sixteen bit CRC

CRC-5	five bit CRC
CW	continuous wave
dBch	decibels referenced to the integrated power in the reference channel
DBR	dead battery response
DSB	double sideband
DSB-ASK	double-sideband amplitude-shift keying
DSFID	data storage format identifier
DSSS	direct sequence spread spectrum
EOF	end of frame
EPC™	electronic product code
FCC	Federal Communications Commission
FHSS	frequency hopping spread spectrum
Handle	16-bit tag-authentication number
ITF	Interrogator-talks-first

NOTE The common usage is RTF (Reader-talks-first) but the more precise term is ITF, which is used throughout this part of ISO/IEC 18000.

LSB	least significant bit
MSB	most significant bit
NoS	number of sensors
NRZ	non return to zero
NSI	numbering system identifier
OTP	one-time programmable
PC	protocol control
PIE	pulse interval encoding
Pivot	decision threshold differentiating an R=>T data-0 symbol from a data-1 symbol
Plaintext	information that is not cover-coded
PM	passive mode
ppm	parts per million
PR-ASK	phase-reversal amplitude shift keying
R/W	read/write
RFU	reserved for future use
RN16	16-bit random or pseudo-random number
RNG	random or pseudo-random number generator
RO	read only

RTC	real time clock
SDS	Sensor Directory System
SOF	start of frame
SS	Simple Sensor
SSB	single sideband
SSB-ASK	single-sideband amplitude-shift keying
SSD	simple sensor data
TCRS	Tag Capabilities Reporting and Setting
TEDS	Transducer Electronic Data Sheet
TID	tag-identification or tag identifier, depending on context
UMI	user-memory indicator
UTC	universal coordinated time
WC	wildcard
Word	16 bits
XI	XPC_W1 indicator
XPC	extended protocol control
XPC_W1	XPC word 1
XPC_W2	XPC word 2
XEB	XPC extension bit110

5.3 Notation

This part of ISO/IEC 18000 uses the following notational conventions for Type C.

— States and flags are denoted in bold. Example: **ready**.

— Commands are denoted in italics. Variables are also denoted in italics. Where there might be confusion between commands and variables, this part of ISO/IEC 18000 will make an explicit statement. Example: *Query*.

— Command parameters are underlined. Example: Pointer.

— For logical negation, labels are preceded by '~'. Example: If **flag** is true, then **~flag** is false.

— The symbol, R=>T, refers to commands or signalling from an Interrogator to a tag (reader-to-tag).

— The symbol, T=>R, refers to commands or signalling from a tag to an Interrogator (tag-to-reader).

— The term "address" is used as a synonym for "bit address". All addresses specified in the Clauses denoted to Type C are bit-addresses, unless specified differently (e.g. by adding an according prefix such as "word" in "word address"). The same applies for figures, where all shown addresses are to be interpreted as bit-addresses, unless specified differently.

— The term "word" always refers to a 16-bit word.

6 Type C

6.1 Parameter tables

Table 2, Table 2, Table 4 and Table 4 contain the parameters for Type C in accordance with ISO/IEC 18000-1. Detailed description of the operating modes and parameters are specified in the subsequent clauses.

Table 1 — Interrogator to tag link parameters

Ref.	Parameter Name	Description	Reference
Int:1	Operating Frequency Range	860 MHz – 960 MHz, as required by the local regulations	See clause **6.4.1.1**
Int:1a	Default Operating Frequency	Determined by local radio regulations and by the radio-frequency environment at the time of the communication	See clause **6.4.1.1**
Int:1b	Operating Channels (spread-spectrum systems)	In accordance with the local regulations; if the channelisation is unregulated, then as specified	See clause **6.4.1.2.10**, **Annex G**
Int:1c	Operating Frequency Accuracy	As specified	See clause **6.4.1.2.1**
Int:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	In accordance with the local regulations	See clause **6.4.1.2.9**
Int:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	In accordance with the local regulations	See clause **6.4.1.2.9**
Int:2	Occupied Channel Bandwidth	In accordance with the local regulations	
Int:2a	Minimum Receiver Bandwidth	In accordance with the local regulations	
Int:3	Interrogator Transmit Maximum EIRP	In accordance with the local regulations	
Int:4	Interrogator Transmit Spurious Emissions	As specified; tighter emission limits may be imposed by the local regulations	See clause **6.4.1.2.11**
Int:4a	Interrogator Transmit Spurious Emissions, In-Band (spread-spectrum systems)	As specified; tighter emission limits may be imposed by the local regulations	See clause **6.4.1.2.11**
Int:4b	Interrogator Transmit Spurious Emissions, Out-of-Band	As specified; tighter emission limits may be imposed by the local regulations	See clause **6.4.1.2.11**
Int:5	Interrogator Transmitter Spectrum Mask	As specified; tighter emission limits may be imposed by the local regulations	**Figure 6**, **Figure 7**
Int:6	Timing	As specified.	See clause **6.4.1.6**, **Figure 16**, **Table 13**
Int:6a	Transmit-to-Receive Turn-Around Time	10 – 310 µs	See clause **6.4.1.6**, **Figure 16**, **Table 13**
Int:6b	Receive-to-Transmit Turn-Around Time	3,5 – 610 µs when tag is in **reply** & **acknowledged** states; no limit otherwise	See clause **6.4.1.6**, **Figure 16**, **Table 13**

Ref.	Parameter Name	Description	Reference
Int:6c	Dwell Time or Interrogator Transmit Power-On Ramp	1500 µs, maximum settling time	See clause **6.4.1.2.6**, **Figure 3**, **Table 6**
Int:6d	Decay Time or Interrogator Transmit Power-Down Ramp	500 µs, maximum	**Figure 3**, **Table 7**
Int:7	Modulation	DSB-ASK, SSB-ASK, or PR-ASK	See clause **6.4.1.2.2**
Int:7a	Spreading Sequence (direct-sequence [DSSS] systems)	Not applicable.	
Int:7b	Chip Rate (spread-spectrum systems)	Not applicable.	
Int:7c	Chip Rate Accuracy (spread-spectrum systems)	Not applicable.	
Int:7d	Modulation Depth	90% nominal	See clause **6.4.1.2.5**, **Figure 2**, **Table 5**
Int:7e	Duty Cycle	48% – 82.3% (time the waveform is high)	See clause **6.4.1.2.3**, **Figure 1**, **Table 5**
Int:7f	FM Deviation	Not applicable.	
Int:8	Data Coding	PIE	See clause **6.4.1.2.3**, **Figure 1**
Int:9	Bit Rate	26.7 kbit/s to 128 kbit/s (assuming equiprobable data)	See clause **6.4.1.2.4**
Int:9a	Bit Rate Accuracy	+/– 1%, minimum	See clause **6.4.1.2.4**
Int:10	Interrogator Transmit Modulation Accuracy	As specified	See clause **6.4.1.2.4**
Int:11	Preamble	Required	See clause **6.4.1.2.8**
Int:11a	Preamble Length	As specified	See clause **6.4.1.2.8**
Int:11b	Preamble Waveform(s)	As specified	See **Figure 4**
Int:11c	Bit Sync Sequence	None	
Int:11d	Frame Sync Sequence	Required	See clause **6.4.1.2.8**, **Figure 4**
Int:12	Scrambling (spread-spectrum systems)	Not Applicable.	
Int:13	Bit Transmission Order	MSB is transmitted first	See clause **6.4.1.4**
Int:14	Wake-up process	As specified	See clause **6.4.1.2**
Int:15	Polarization	Not specified	

Table 2 — Tag to Interrogator link parameters

Ref.	Parameter Name	Description	Reference
Tag:1	Operating Frequency Range	860 MHz – 960 MHz, inclusive	See clause **6.4.1.1**
Tag:1a	Default Operating Frequency	Tags respond to Interrogator signals that satisfy Int:1a	See clause **6.4.1.1**
Tag:1b	Operating Channels (spread-spectrum systems)	Tags respond to Interrogator signals that satisfy Int:1b	See clause **6.4.1.2.10**
Tag:1c	Operating Frequency Accuracy	As specified	See clause **6.4.1.3.3** **Table 9**
Tag:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	Tags respond to Interrogator signals that satisfy Int:1d	See clause **6.4.1.2.9**
Tag:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	Tags respond to Interrogator signals that satisfy Int:1e	See clause **6.4.1.2.9**
Tag:2	Occupied Channel Bandwidth	In accordance with the local regulations	
Tag:3	Transmit Maximum EIRP	In accordance with the local regulations	
Tag:4	Transmit Spurious Emissions	In accordance with the local regulations	
Tag:4a	Transmit Spurious Emissions, In-Band (spread spectrum systems)	In accordance with the local regulations	
Tag:4b	Transmit Spurious Emissions, Out-of-Band	In accordance with the local regulations	
Tag:5	Transmit Spectrum Mask	In accordance with the local regulations	
Tag:6a	Transmit-to-Receive Turn-Around Time	3,5 – 610 µs in **reply** & **acknowledged** states; no limit otherwise	See clause **6.4.1.6**, **Figure 16**, **Table 13**
Tag:6b	Receive-to-Transmit Turn-Around Time	10 – 310 µs	See clause **6.4.1.6**, **Figure 16**, **Table 13**
Tag:6c	Dwell Time or Transmit Power-On Ramp	Receive commands 1.5 ms after power-up	See clause **6.4.1.2**
Tag:6d	Decay Time or Transmit Power-Down Ramp	Not applicable.	
Tag:7	Modulation	ASK and/or PSK modulation (selected by tag)	See clause **6.4.1.3.1**
Tag:7a	Spreading Sequence (direct sequence [DSSS] systems)	Not applicable.	
Tag:7b	Chip Rate (spread spectrum systems)	Not applicable.	
Tag:7c	Chip Rate Accuracy (spread spectrum systems)	Not applicable.	
Tag:7d	On-Off Ratio	Not specified	

Ref.	Parameter Name	Description	Reference
Tag:7e	Subcarrier Frequency	40 kHz to 640 kHz	See clause **6.4.1.3.3** **Table 9**
Tag:7f	Subcarrier Frequency Accuracy	As specified	**Table 9**
Tag:7g	Subcarrier Modulation	Miller, at the data rate	See clause **6.4.1.3.2.3** , **Figure 13**
Tag:7h	Duty Cycle	FM0: 50%, nominal Subcarrier: 50%, nominal	See clause **6.4.1.3.2.1** , **6.4.1.3.2.3**
Tag:7i	FM Deviation	Not applicable.	
Tag:8	Data Coding	Baseband FM0 or Miller-modulated subcarrier (selected by the Interrogator)	See clause **6.4.1.3.2**
Tag:9	Bit Rate	FM0: 40 kbps to 640 kbps Subcarrier modulated: 5 kbps to 320 kbps	See clause **6.4.1.3.3** **Table 9**
Tag:9a	Bit Rate Accuracy	Same as Subcarrier Frequency Accuracy; see Tag:7f	See clause **6.4.1.3.3**, **Table 9**, **Table 10**
Tag:10	Tag Transmit Modulation Accuracy (frequency-hopping [FHSS] systems	Not applicable.	
Tag:11	Preamble	Required	See clause **6.4.1.3.2.2** , See clause **6.4.1.3.2.4**
Tag:11a	Preamble Length	As specified	See **Table 9**, See **Figure 15**
Tag:11b	Preamble Waveform	As specified	See **Table 9**, See **Figure 15**
Tag:11c	Bit-Sync Sequence	None	
Tag:11d	Frame-Sync Sequence	None	
Tag:12	Scrambling (spread-spectrum systems)	Not applicable.	
Tag:13	Bit Transmission Order	MSB is transmitted first	See clause **6.4.1.4**
Tag:14	Reserved	Deliberately left blank.	
Tag:15	Polarization	Tag dependent; not specified by this document	
Tag:16	Minimum Tag Receiver Bandwidth	Tag dependent; not specified by this document	

Table 3 — Protocol parameters

Ref.	Parameter Name	Description	Reference
P:1	Who talks first	Interrogator	See clause **6.4**
P:2	Tag addressing capability	As specified	See clause **6.4.2.1**
P:3	Tag ID	Contained in tag memory	See clause **6.4.2.1.3**
P:3a	Tag ID Length	Variable, as specified in clause **6.4.2.1.3**	See clause **6.4.2.1.3**
P:3b	Tag ID Format	see clause **6.4.2.1.3**	See clause **6.4.2.1.3**
P:4	Read size	Multiples of 16 bits.	See clause **6.4.2.11.3.2** **Table 34**
P:5	Write Size	Multiples of 16 bits	See clause **6.4.2.11.3.3**, **Table 36**, See clause **6.4.2.11.3.7**, **Table 47**
P:6	Read Transaction Time	Varied with R=>T and T=>R link rate and number of bits being read	See clause **6.4.2.11.3.2**
P:7	Write Transaction Time	20 ms (maximum) after end of *Write* command	See clause **6.4.2.11.3.3**, See clause **6.4.2.11.3.7**, **Figure 22**
P:8	Error detection	Interrogator-to-tag: *Select* command: CRC-16 *Query* command: CRC-5 Other Inventory commands: Command length Access commands: CRC-16 Tag-to-Interrogator: PC, UII: CRC-16 RN16: None or CRC-16 (varies with command) handle: CRC-16 All other: CRC-16	See clause **6.4.2.10** and its subsections
P:9	Error correction	None	
P:10	Memory size	Tag dependent, extensible (size is neither limited nor specified by this document)	
P:11	Command structure and extensibility	As specified.	**Table 19**

Table 4 — Anti-collision parameters

Ref.	Parameter Name	Description	Reference
A:1	Type (Probabilistic or Deterministic)	Probabilistic	See clause **6.4.2.6**
A:2	Linearity	Linear up to 2^{15} tags in the Interrogator's RF field; above that number, NlogN for tags with unique UIIs	See clause **6.4.2.8**
A:3	Tag inventory capacity	Unlimited for tags with unique UIIs.	See clause **6.4.2.8**

6.2 Protocol overview

6.2.1 Physical layer

An Interrogator sends information to one or more tags by modulating an RF carrier using double-sideband amplitude shift keying (DSB-ASK), single-sideband amplitude shift keying (SSB-ASK), or phase-reversal amplitude shift keying (PR-ASK) using a pulse-interval encoding (PIE) format. Tags receive their operating energy from this same modulated RF carrier.

An Interrogator receives information from a tag by transmitting an unmodulated RF carrier and listening for a backscattered reply. Tags communicate information by backscatter modulating the amplitude and/or phase of the RF carrier. The encoding format, selected in response to Interrogator commands, is either FM0 or Miller-modulated subcarrier. The communications link between Interrogators and tags is half-duplex, meaning that tags shall not be required to demodulate Interrogator commands while backscattering. A tag shall not respond to a mandatory or optional command using full-duplex communications.

6.2.2 Tag-identification layer

An Interrogator manages tag populations using three basic operations:

Select. The operation of choosing a tag population for inventory and access. A *Select* command may be applied successively to select a particular tag population based on user-specified criteria. This operation is analogous to selecting records from a database.

Inventory. The operation of identifying tags. An Interrogator begins an inventory round by transmitting a *Query* command in one of four sessions. One or more tags may reply. The Interrogator detects a single tag reply and requests the PC/XPC word(s), UII, and CRC from the tag. Inventory comprises multiple commands. An inventory round operates in one and only one session at a time.

Access. The operation of communicating with (reading from and/or writing to) a tag. An individual tag must be uniquely identified prior to access. Access comprises multiple commands, some of which employ one-time-pad based cover-coding of the R=>T link.

6.3 Command types and command structure

6.3.1 General

This clause defines 4 command types: Mandatory, optional, custom, and proprietary.

All commands defined in this International Standard are either mandatory or optional, as specified in the definition of the command itself. Custom or proprietary commands are vendor-defined.

6.3.2 Mandatory

Conformant tags shall support all mandatory commands.

Conformant Interrogators shall support all mandatory commands.

6.3.3 Optional

Tags may or may not support optional commands.

Interrogators may or may not support optional commands.

If an optional command is used, it shall be implemented in the manner specified in this document.

6.3.4 Custom

Custom commands are not specified in this document.

An Interrogator shall issue a custom command only after (i) singulating a tag, and (ii) reading (or having prior knowledge of) the tag manufacturer's identification in the tag's TID memory. An Interrogator shall use a custom command only in accordance with the specifications of the tag manufacturer identified in the tag ID.

A custom command shall not solely duplicate the functionality of any mandatory or optional command defined in this International Standard by a different method.

6.3.5 Proprietary

Proprietary commands are not specified in this document.

All proprietary commands shall be capable of being permanently disabled. Proprietary commands are intended for manufacturing purposes and shall not be used in field-deployed RFID systems.

6.4 Description of operating procedure

The operating procedure defines the physical and logical requirements for an ITF, random-slotted collision arbitration, RFID system operating in the 860 MHz – 960 MHz frequency range.

6.4.1 Signalling

The signalling interface between an Interrogator and a tag may be viewed as the physical layer in a layered network communication system. The signalling interface defines frequencies, modulation, data coding, RF envelope, data rates, and other parameters required for RF communications.

6.4.1.1 Operational frequencies

Tags shall receive power from and communicate with Interrogators within the frequency range from 860 MHz to 960 MHz, inclusive. An Interrogator's choice of operational frequency will be determined by local radio regulations and by the local radio-frequency environment. Interrogators that are claimed to operate in Dense-Interrogator environments shall support, but are not required to always use, the optional Dense-Interrogator mode described in Annex G.

6.4.1.2 Interrogator-to-Tag (R=>T) communications

An Interrogator communicates with one or more tags by modulating an RF carrier using DSB-ASK, SSB-ASK, or PR-ASK with PIE encoding. Interrogators shall use a fixed modulation format and data rate for the duration of an inventory round, where "inventory round" is defined in ISO/IEC 19762. The Interrogator sets the data rate by means of the preamble that initiates the round.

The high values in Figure 1, Figure 2, Figure 3, Figure 4 and Figure 5, correspond to emitted CW (i.e. an Interrogator delivering power to the tag or tags) whereas the low values correspond to attenuated CW.

6.4.1.2.1 Interrogator frequency accuracy

Interrogators that claim to operate in single- or Multiple-Interrogator environments shall have a frequency accuracy that meets the local regulations.

Interrogators that are claimed to operate in Dense-Interrogator environments shall have a frequency accuracy of +/– 10 ppm over the nominal temperature range (–25 °C to +40 °C), and +/– 20 ppm over the extended temperature range (–40°C to +65 °C) while transmitting, unless the local regulations specify tighter accuracy, in which case the Interrogator frequency accuracy shall meet the local regulations.

Interrogators rated by the manufacturer to have a temperature range wider than nominal but different from extended shall have a frequency accuracy of +/– 10 ppm over the nominal temperature range and +/– 20 ppm to the extent of their rated range. If the local regulations specify tighter frequency accuracy then the Interrogator shall meet the local regulations.

6.4.1.2.2 Modulation

Interrogators shall communicate using DSB-ASK, SSB-ASK, or PR-ASK modulation, detailed in Annex H. Tags shall demodulate all three modulation types.

6.4.1.2.3 Data encoding

The R=>T link shall use PIE, shown in Figure 1. Tari is the reference time interval for Interrogator-to-tag signalling, and is the duration of a data-0. High values represent transmitted CW; low values represent attenuated CW. Pulse modulation depth, rise time, fall time, and PW shall be as specified in Table 5, and shall be the same for a data-0 and a data-1. Interrogators shall use a fixed modulation depth, rise time, fall time, PW, Tari, data-0 length, and data-1 length for the duration of an inventory round. The RF envelope shall be as specified in Figure 2.

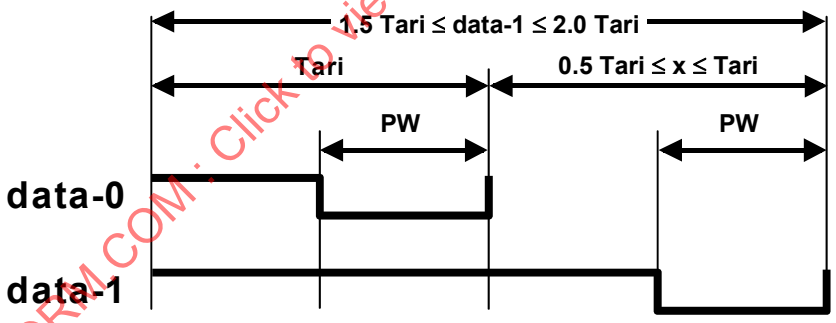


Figure 1 — PIE Symbols

6.4.1.2.4 Tari values

Interrogators shall communicate using Tari values in the range of 6.25µs to 25µs. Interrogator compliance shall be evaluated using at least one Tari value between 6.25µs and 25µs with at least one value of the parameter *x*. The tolerance on all parameters specified in units of Tari shall be +/–1%. The choice of Tari value and *x* shall be in accordance with local radio regulations.

6.4.1.2.5 R=>T RF envelope

The R=>T RF envelope shall comply with Figure 2 and Table 5. The electric or magnetic field strength A (as appropriate) is the maximum amplitude of the RF envelope, measured in units of V/m or A/m, respectively. Tari is defined in Figure 1. The pulsewidth is measured at the 50% point on the pulse. An Interrogator shall not change the R=>T modulation type (i.e. shall not switch between DSB-ASK, SSB-ASK, or PR-ASK) without first powering down its RF waveform (see 6.4.1.2.7).

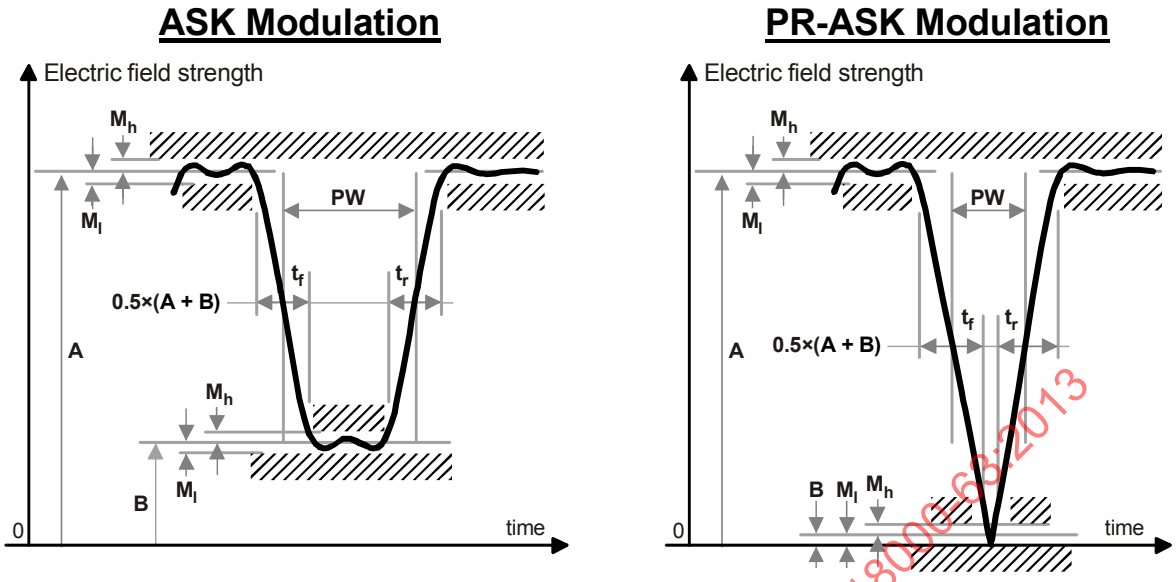


Figure 2 — Interrogator-to-Tag RF envelope

Table 5 — RF envelope parameters

Tari	Parameter	Symbol	Minimum	Nominal	Maximum	Units
6.25 µs to 25 µs	Modulation Depth	(A–B)/A	80	90	100	%
	RF Envelope Ripple	$M_h = M_l$	0		0.05(A–B)	V/m
	RF Envelope Rise Time	$t_{r,10–90\%}$	0		0.33Tari	µs
	RF Envelope Fall Time	$t_{f,10–90\%}$	0		0.33Tari	µs
	RF Pulsewidth	PW	MAX(0.265Tari, 2)		0.525Tari	µs

6.4.1.2.6 Interrogator power-up waveform

The Interrogator power-up RF envelope shall comply with Figure 3 and Table 6. Once the carrier level has risen above the 10% level, the power-up envelope shall rise monotonically until at least the ripple limit M_l. The RF envelope shall not fall below the 90% point in Figure 3 during interval T_s. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table 6 (i.e. before the end of T_s). Interrogators shall meet the frequency-accuracy requirement specified in 6.4.1.2.1 by the end of interval T_s in Figure 3.

6.4.1.2.7 Interrogator power-down waveform

The Interrogator power-down RF envelope shall comply with Figure 3 and Table 7. Once the carrier level has fallen below the 90% level, the power-down envelope shall fall monotonically until the power-off limit M_s. Once powered off, an Interrogator shall remain powered off for a least 1ms before powering up again.

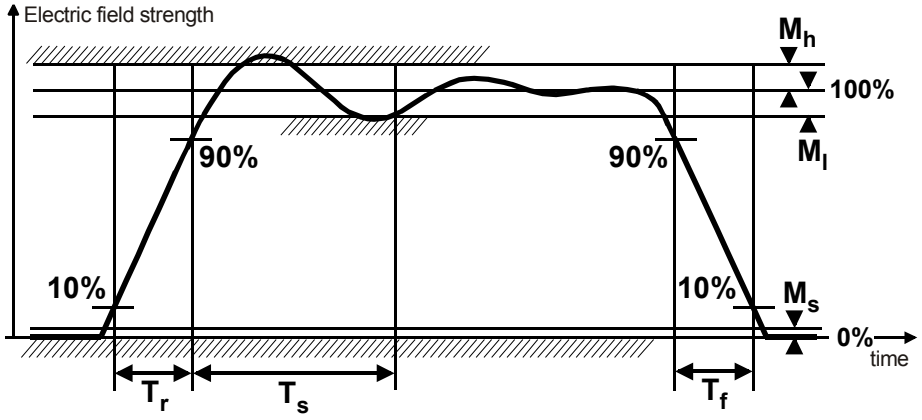


Figure 3 — Interrogator power-up and power-down RF envelope

Table 6 — Interrogator power-up waveform parameters

Parameter	Definition	Minimum	Nominal	Maximum	Units
T_r	Rise time	1		500	µs
T_s	Settling time			1500	µs
M_s	Signal level when OFF			1	% full scale
M_l	Undershoot			5	% full scale
M_h	Overshoot			5	% full scale

Table 7 — Interrogator power-down waveform parameters

Parameter	Definition	Minimum	Nominal	Maximum	Units
T_f	Fall time	1		500	µs
M_s	Signal level when OFF			1	% full scale
M_l	Undershoot			5	% full scale
M_h	Overshoot			5	% full scale

6.4.1.2.8 R=>T preamble and frame-sync

An Interrogator shall begin all R=>T signalling with either a preamble or a frame-sync, both of which are shown in Figure 4. A preamble shall precede a *Query* command (see 6.4.2.11.2.1) and denotes the start of an inventory round. All other signalling shall begin with a frame-sync. The tolerance on all parameters specified in units of Tari shall be +/–1%. PW shall be as specified in Table 5. The RF envelope shall be as specified in Figure 2.

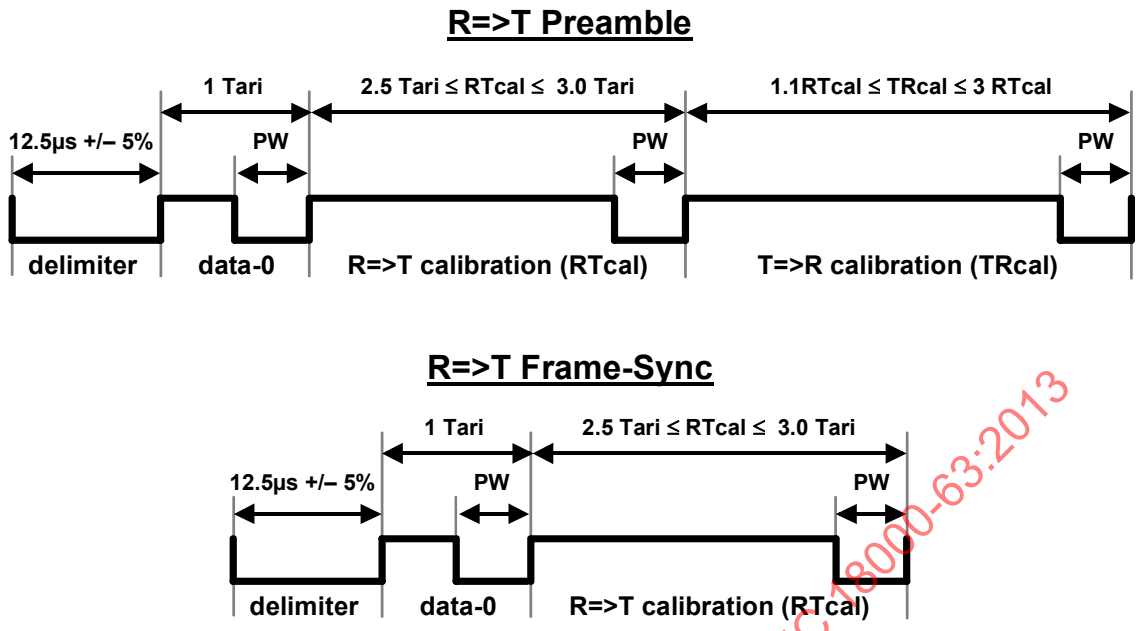


Figure 4 — R=>T preamble and frame-sync

A preamble shall comprise a fixed-length start delimiter, a data-0 symbol, an R=>T calibration (RTcal) symbol, and a T=>R calibration (TRcal) symbol.

RTcal: An Interrogator shall set RTcal equal to the length of a data-0 symbol plus the length of a data-1 symbol (RTcal = 0_{length} + 1_{length}). A tag shall measure the length of RTcal and compute *pivot* = RTcal / 2. A tag shall interpret subsequent Interrogator symbols shorter than *pivot* to be data-0s, and subsequent Interrogator symbols longer than *pivot* to be data-1s. A tag shall interpret symbols longer than 4 RTcal to be invalid. Prior to changing RTcal, an Interrogator shall transmit CW for a minimum of 8 RTcal.

TRcal: An Interrogator shall specify a tag's backscatter link frequency (its FM0 datarate or the frequency of its Miller subcarrier) using the TRcal and divide ratio (DR) in the preamble and payload, respectively, of a *Query* command that initiates an inventory round. Equation (1) specifies the relationship between the backscatter link frequency (BLF), TRcal, and DR. A tag shall measure the length of TRcal, compute BLF, and adjust its T=>R link rate to be equal to BLF (Table 9 shows BLF values and tolerances). The TRcal and RTcal that an Interrogator uses in any inventory round shall meet the constraints in Equation (2):

$$BLF = \frac{DR}{TRcal} \quad (1)$$

$$1.1 \times RTcal \leq TRcal \leq 3 \times RTcal \quad (2)$$

A frame-sync is identical to a preamble, minus the TRcal symbol. An Interrogator, for the duration of an inventory round, shall use the same length RTcal in a frame-sync as it used in the preamble that initiated the round.

6.4.1.2.9 Frequency-hopping spread-spectrum waveform

When an Interrogator uses frequency-hopping spread spectrum (FHSS) signalling, the Interrogator's RF envelope shall comply with Figure 5 and Table 8. The RF envelope shall not fall below the 90% point in Figure 5 during interval T_{hs}. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table 8 (i.e. before T_{hs}). The maximum time between frequency hops and the minimum RF-off time during a hop shall meet local regulatory requirements. Interrogators shall meet the frequency-accuracy requirement specified in 6.4.1.2.1 by the end of interval T_{hs} in Figure 5.

6.4.1.2.10 Frequency-hopping spread-spectrum channelisation

Interrogators that are claimed to operate in single-Interrogator environments shall meet the local regulations for spread-spectrum channelisation. Interrogators that are claimed to operate in multiple- or Dense-Interrogator environments shall meet the local regulations for spread-spectrum channelisation, unless the channelisation is unregulated, in which case Interrogators shall adopt the channelisation described by Annex G (Annex G describes multiple- and Dense-Interrogator channelised signalling).

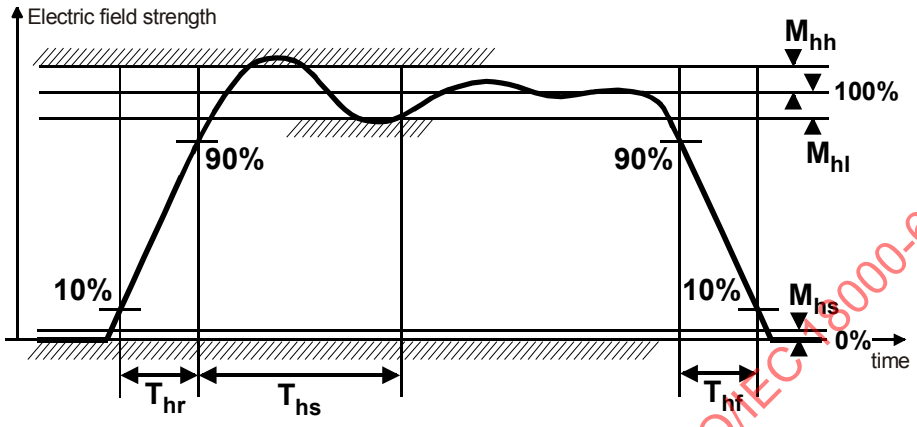


Figure 5 — FHSS Interrogator RF envelope

Table 8 — FHSS waveform parameters

Parameter	Definition	Minimum	Nominal	Maximum	Units
T_{hr}	Rise time			500	µs
T_{hs}	Settling time			1500	µs
T_{hf}	Fall time			500	µs
M_{hs}	Signal level during hop			1	% full scale
M_{hl}	Undershoot			5	% full scale
M_{hh}	Overshoot			5	% full scale

6.4.1.2.11 Transmit mask

Interrogators that are claimed to operate according to this International Standard shall meet the local regulations for out-of-channel and out-of-band spurious radio-frequency emissions.

Interrogators that are claimed to operate in Multiple-Interrogator environments, in addition to meeting the local regulations, shall also meet the Multiple-Interrogator Transmit Mask specified in this International Standard:

Multiple-Interrogator Transmit Mask: For an Interrogator transmitting random data in channel *R*, and any other channel $S \neq R$, the ratio of the integrated power *P*() in channel *S* to that in channel *R* shall not exceed the specified values:

$|R - S| = 1$: $10\log_{10}(P(S) / P(R)) < -20$ dB
$|R - S| = 2$: $10\log_{10}(P(S) / P(R)) < -50$ dB
$|R - S| = 3$: $10\log_{10}(P(S) / P(R)) < -60$ dB

$|R - S| > 3$: $10\log_{10}(P(S) / P(R)) < -65$ dB

Where $P()$ denotes the total integrated power in the specified channel. This mask is shown graphically in Figure 6, with dBch defined as dB referenced to the integrated power in the reference channel. The channel width shall be as specified by the local regulations, unless the width is unregulated, in which case Interrogators shall adopt a channel width as described in Annex I. The channel spacing shall be set equal to the channel width (measured channel centre to channel centre). For any transmit channel R, two exceptions to the mask are permitted, provided that

neither exception exceeds –50 dBch, and

neither exception exceeds local regulatory requirements.

An exception occurs when the integrated power in a channel S exceeds the mask. Each channel that exceeds the mask shall be counted as an exception.

Interrogators that are claimed to operate in Dense-Interrogator environments shall meet both the local regulations and the Transmit Mask shown in Figure 6 of this International Standard, except when operating in the optional Dense-Interrogator mode described in Annex G, in which case they shall instead meet the Dense-Interrogator Transmit Mask described below and shown in Figure 7. Interrogators may meet the Dense-Interrogator Transmit Mask during non-Dense-Interrogator operation. Regardless of the mask used, Interrogators certified for operation in Dense-Interrogator environments shall not be permitted the two exceptions to the transmit mask that are allowed for Interrogators certified for operation in Multiple-Interrogator environments.

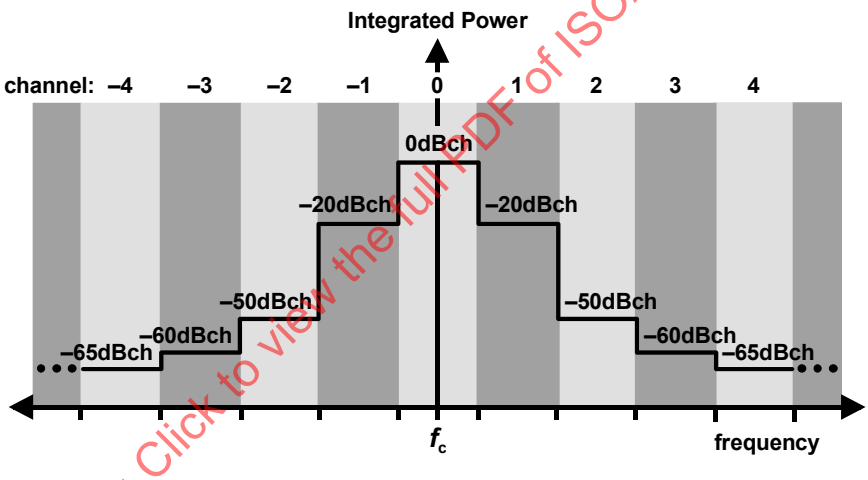


Figure 6 — Transmit mask for Multiple-Interrogator environments

Dense-Interrogator Transmit Mask: For Interrogator transmissions centred at a frequency f_c, a 2.5/Tari bandwidth R_{BW} also centred at f_c, an offset frequency f_o = 2.5/Tari, and a 2.5/Tari bandwidth S_{BW} centred at ($n \times f_o$) + f_c (integer n), the ratio of the integrated power $P()$ in S_{BW} to that in R_{BW} with the Interrogator transmitting random data shall not exceed the specified values:

$|n| = 1$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -30$ dB

$|n| = 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -60$ dB

$|n| > 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -65$ dB

Where $P()$ denotes the total integrated power in the 2.5/Tari reference bandwidth. This mask is shown graphically in Figure 7, with dBch defined as dB referenced to the integrated power in the reference channel.

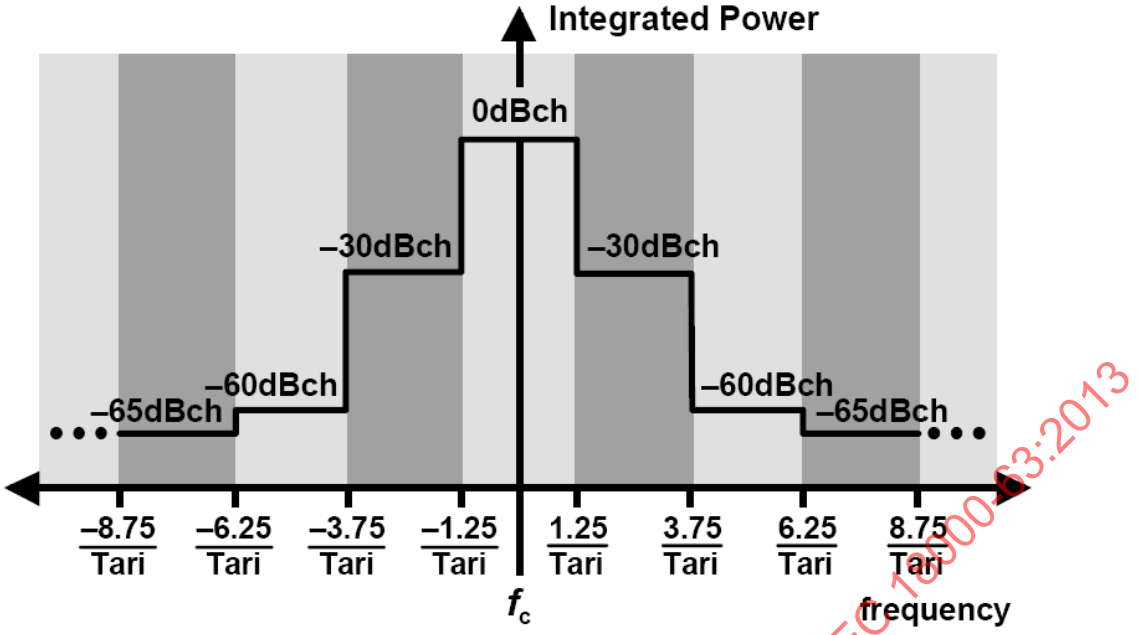


Figure 7 — Transmit mask for Dense-Interrogator environments

NOTE It is recommended that future Interrogator designs fulfill the Dense Interrogator transmit mask. Additional information is provided in Annex Q.

6.4.1.3 Tag-to-Interrogator (T=>R) communications

A tag communicates with an Interrogator using backscatter modulation, in which the tag switches the reflection coefficient of its antenna between two states in accordance with the data being sent.

A tag shall backscatter using a fixed modulation format, data encoding, and data rate for the duration of an inventory round, where "inventory round" is defined in 6.4.2.8. The tag selects the modulation format; the Interrogator selects the encoding and data rate by means of the *Query* command that initiates the round. The low values in Figure 9, Figure 10, Figure 11, Figure 13, Figure 14 and Figure 15 correspond to the antenna-reflectivity state the tag exhibits during the CW period prior to a T=>R preamble (e.g. an ASK tag absorbing power), whereas the high values correspond to the antenna-reflectivity state the tag exhibits during the first high pulse of a T=>R preamble (e.g. an ASK tag reflecting power).

6.4.1.3.1 Modulation

Tag backscatter shall use ASK and/or PSK modulation. The tag vendor selects the modulation format. Interrogators shall demodulate both modulation types.

6.4.1.3.2 Data encoding

Tags shall encode the backscattered data as either FM0 baseband or Miller modulation of a subcarrier at the data rate. The Interrogator commands the encoding choice.

6.4.1.3.2.1 FM0 baseband

Figure 8 shows basis functions and a state diagram for generating FM0 (bi-phase space) encoding. FM0 inverts the baseband phase at every symbol boundary; a data-0 has an additional mid-symbol phase inversion. The state diagram in Figure 8 maps a logical data sequence to the FM0 basis functions that are transmitted. The state labels, S_1–S_4, indicate four possible FM0-encoded symbols, represented by the two phases of each of the FM0 basis functions. The state labels also represent the FM0 waveform that is transmitted upon entering the state. The labels on the state transitions indicate the logical values of the data

sequence to be encoded. For example, a transition from state S_2 to S_3 is disallowed because the resulting transmission would not have a phase inversion on a symbol boundary.

Figure 9 shows generated baseband FM0 symbols and sequences. The duty cycle of a 00 or 11 sequence, measured at the modulator output, shall be a minimum of 45% and a maximum of 55%, with a nominal value of 50%. FM0 encoding has memory; consequently, the choice of FM0 sequences in Figure 9 depends on prior transmissions. FM0 signalling shall always end with a "dummy" data-1 bit at the end of a transmission, as shown in Figure 10.

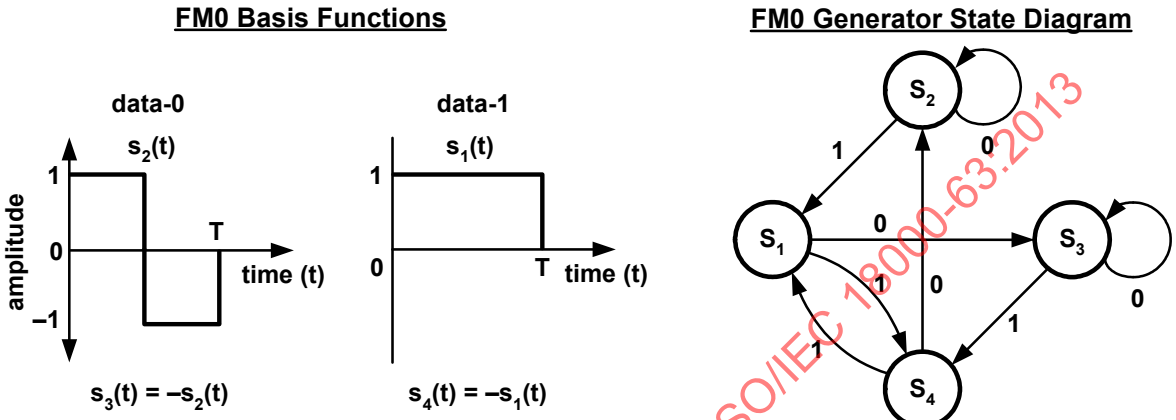


Figure 8 — FM0 basis functions and generator state diagram

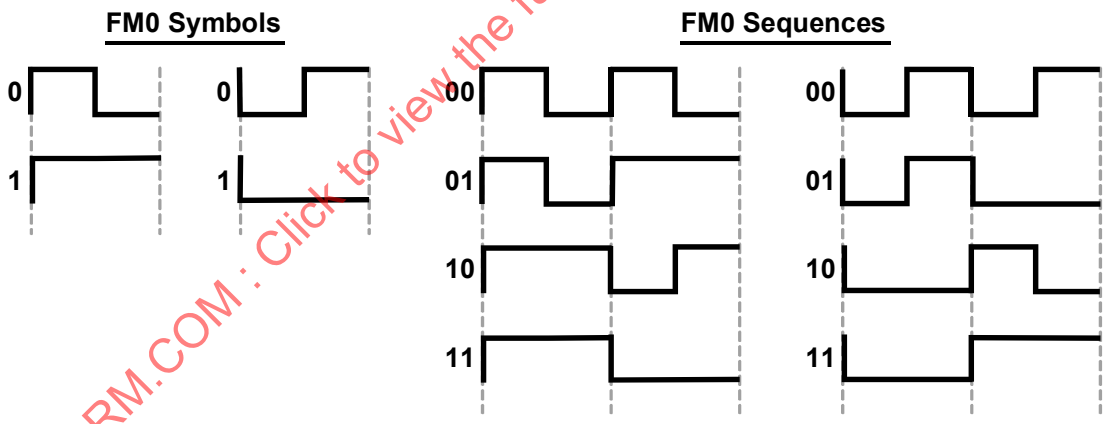


Figure 9 — FM0 symbols and sequences

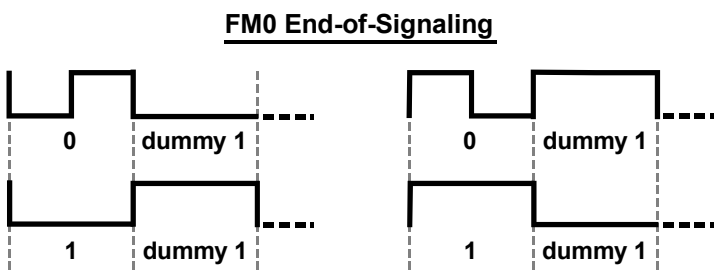


Figure 10 — Terminating FM0 transmissions

6.4.1.3.2.2 FM0 preamble

T=>R FM0 signalling shall begin with one of the two preambles shown in Figure 11. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round, unless a tag is replying to a command that writes to memory, in which case a tag shall use the extended preamble regardless of TRext (i.e. the tag replies as if TRext=1 regardless of the TRext value specified in the *Query*—see 6.4.2.11.3). The "v" shown in Figure 11 indicates an FM0 violation (i.e. a phase inversion should have occurred but did not).

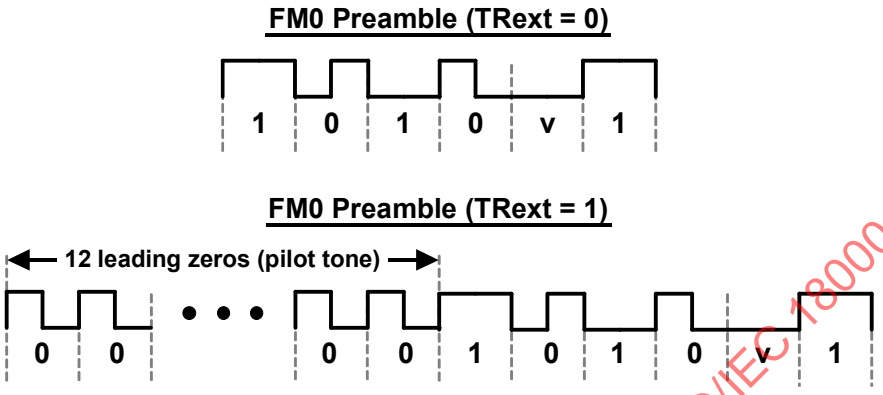


Figure 11 — FM0 T=>R preamble

6.4.1.3.2.3 Miller-modulated subcarrier

Figure 12 shows basis functions and a state diagram for generating Miller encoding. Baseband Miller inverts its phase between two data-0s in sequence. Baseband Miller also places a phase inversion in the middle of a data-1 symbol. The state diagram in Figure 12 maps a logical data sequence to baseband Miller basis functions. The state labels, S_1–S_4, indicate four possible Miller-encoded symbols, represented by the two phases of each of the Miller basis functions. The state labels also represent the baseband Miller waveform that is generated upon entering the state. The transmitted waveform is the baseband waveform multiplied by a square-wave at M times the symbol rate. The labels on the state transitions indicate the logical values of the data sequence to be encoded. For example, a transition from state S_1 to S_3 is disallowed because the resulting transmission would have a phase inversion on a symbol boundary between a data-0 and a data-1.

Figure 13 shows Miller-modulated subcarrier sequences; the Miller sequence shall contain exactly two, four, or eight subcarrier cycles per bit, depending on the M value specified in the *Query* command that initiated the inventory round (see Figure 12). The duty cycle of a 0 or 1 symbol, measured at the modulator output, shall be a minimum of 45% and a maximum of 55%, with a nominal value of 50%. Miller encoding has memory; consequently, the choice of Miller sequences in Figure 13 depends on prior transmissions. Miller signalling shall always end with a "dummy" data-1 bit at the end of a transmission, as shown in Figure 14.

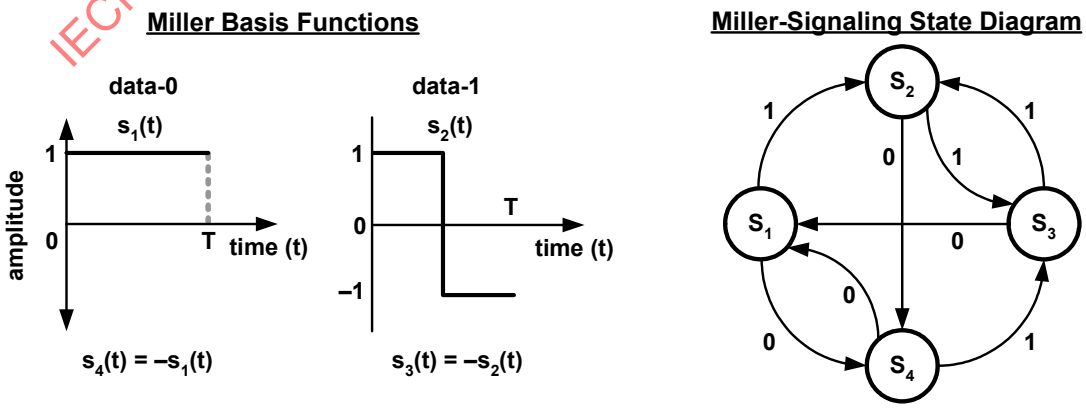


Figure 12 — Miller basis functions and generator state diagram

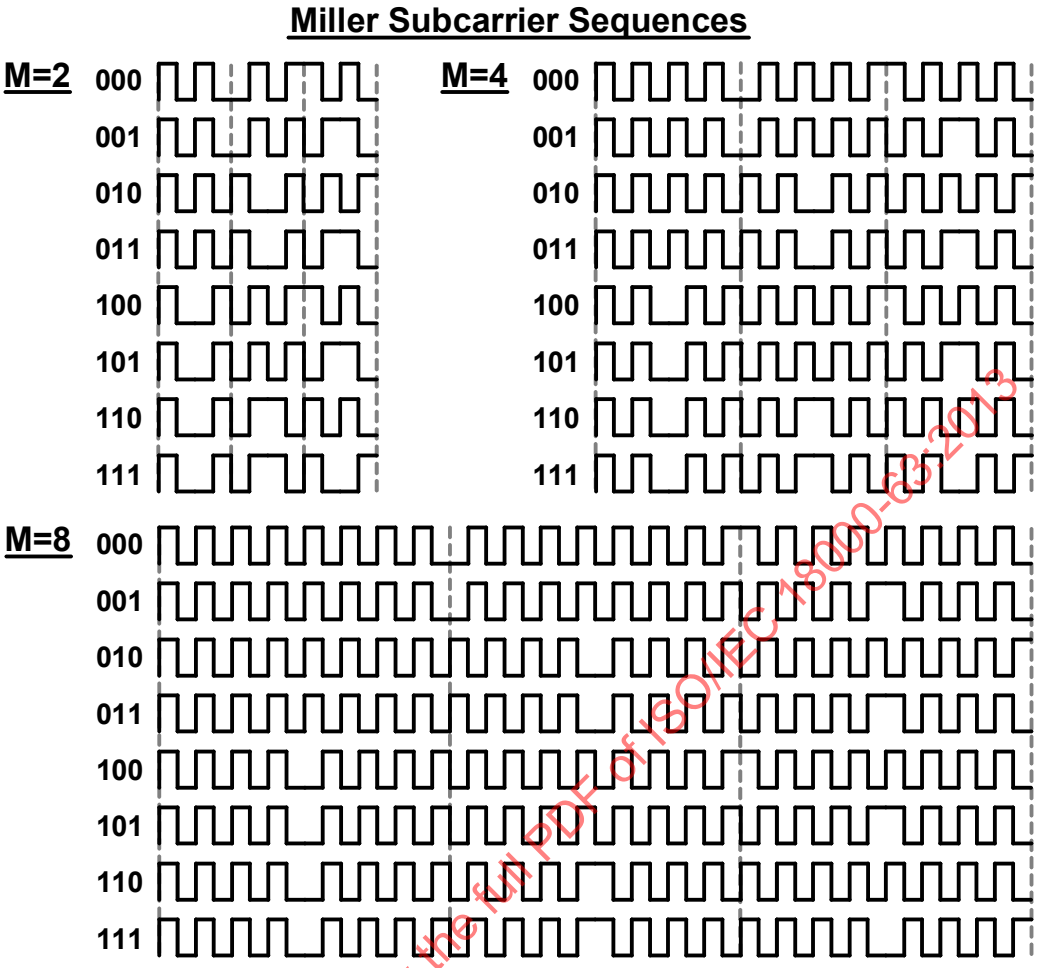


Figure 13 — Subcarrier sequences

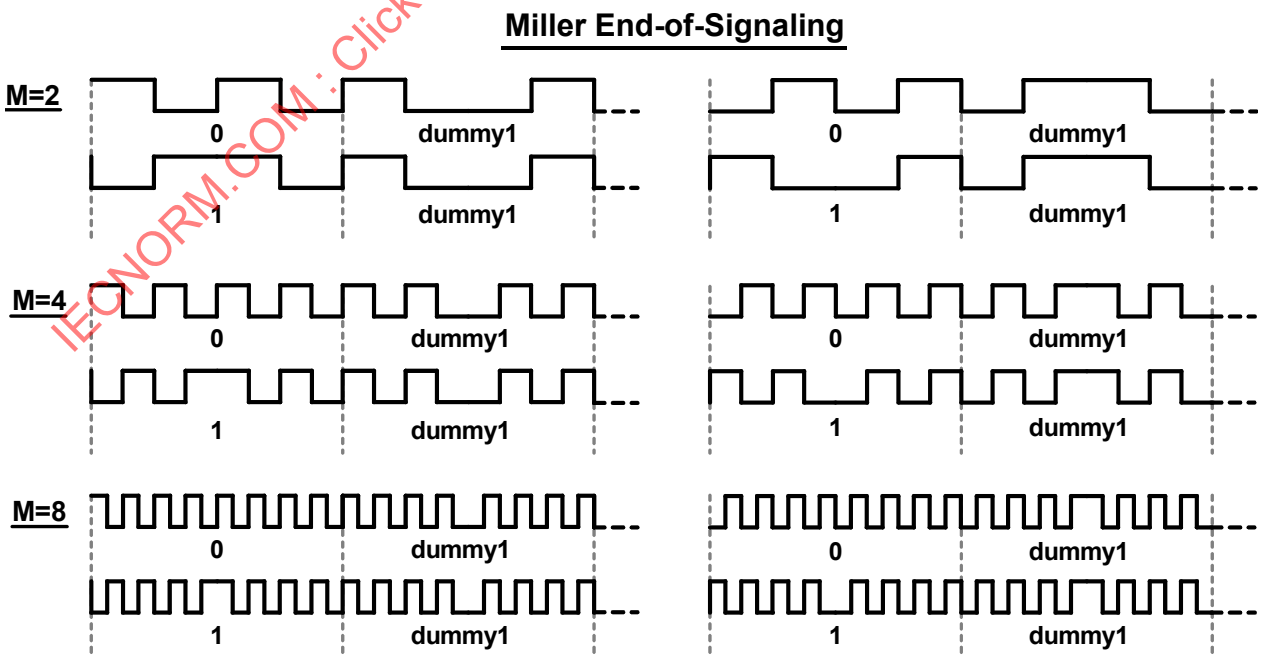


Figure 14 — Terminating subcarrier transmissions

6.4.1.3.2.4 Miller subcarrier preamble

T=>R subcarrier signalling shall begin with one of the two preambles shown in Figure 15. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round, unless a tag is replying to a command that writes to memory, in which case a tag shall use the extended preamble regardless of TRext (i.e. the tag replies as if TRext=1 regardless of the TRext value specified in the *Query*—see 6.4.2.11.3).

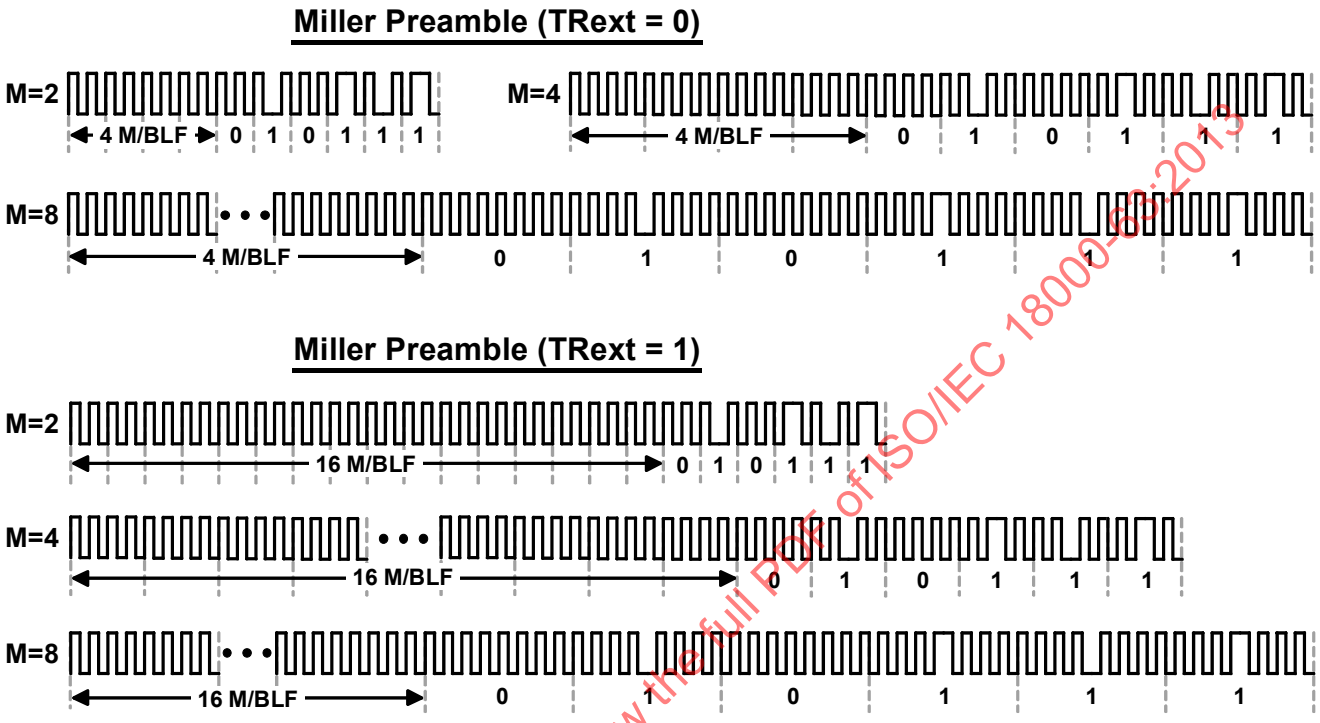


Figure 15 — Subcarrier T=>R preamble

6.4.1.3.3 Tag supported Tari values and backscatter link rates

Tags shall support all R=>T Tari values in the range of 6.25μs to 25μs, over all parameters allowed by 6.4.1.2.3.

Tags shall support the T=>R link frequencies and tolerances specified in Table 9, and the T=>R data rates specified in Table 10. The frequency-variation requirement in Table 9 includes both frequency drift and short-term frequency variation during tag response to an Interrogator command. The *Query* command that initiates an inventory round specifies DR in Table 9 and M in Table 10; the preamble that precedes the *Query* specifies TRcal. BLF is computed using Eq. (1). These four parameters together define the backscatter frequency, modulation type (FM0 or Miller), and T=>R data rate for the round (see also 6.4.1.2.8).

Table 9 — Tag-to-Interrogator link frequencies

DR: Divide Ratio	TRcal [a] (µs +/– 1%)	BLF: Link Frequency (kHz)	Frequency Tolerance FT (-25 °C to 40 °C)	Frequency Tolerance FT (-40 °C to 65 °C)	Frequency variation during backscatter
64/3	33.3	640	+ / – 15%	+ / – 15%	+ / – 2.5%
	33.3 < TRcal < 66.7	320 < BLF < 640	+ / – 22%	+ / – 22%	+ / – 2.5%
	66.7	320	+ / – 10%	+ / – 15%	+ / – 2.5%
	66.7 < TRcal < 83.3	256 < BLF < 320	+ / – 12%	+ / – 15%	+ / – 2.5%
	83.3	256	+ / – 10%	+ / – 10%	+ / – 2.5%
	83.3 < TRcal ≤ 133.3	160 ≤ BLF < 256	+ / – 10%	+ / – 12%	+ / – 2.5%
	133.3 < TRcal ≤ 200	107 ≤ BLF < 160	+ / – 7%	+ / – 7%	+ / – 2.5%
	200 < TRcal ≤ 225	95 ≤ BLF < 107	+ / – 5%	+ / – 5%	+ / – 2.5%
8	17.2 ≤ TRcal < 25	320 < BLF ≤ 465	+ / – 19%	+ / – 19%	+ / – 2.5%
	25	320	+ / – 10%	+ / – 15%	+ / – 2.5%
	25 < TRcal < 31.25	256 < BLF < 320	+ / – 12%	+ / – 15%	+ / – 2.5%
	31.25	256	+ / – 10%	+ / – 10%	+ / – 2.5%
	31.25 < TRcal < 50	160 < BLF < 256	+ / – 10%	+ / – 10%	+ / – 2.5%
	50	160	+ / – 7%	+ / – 7%	+ / – 2.5%
	50 < TRcal ≤ 75	107 ≤ BLF < 160	+ / – 7%	+ / – 7%	+ / – 2.5%
	75 < TRcal ≤ 200	40 ≤ BLF < 107	+ / – 4%	+ / – 4%	+ / – 2.5%

A Allowing two different TRcal values (with two different DR values) to specify the same BLF offers flexibility in specifying Tari and RTcal.

Table 10 — Tag-to-Interrogator data rates

M: Number of subcarrier cycles per symbol	Modulation type	Data rate (kbit/s)
1	FM0 baseband	BLF
2	Miller subcarrier	BLF/2
4	Miller subcarrier	BLF/4
8	Miller subcarrier	BLF/8

NOTE The M values supported by the specified commands are specified with the commands.

6.4.1.3.4 Tag power-up timing

Tags energized by an Interrogator shall be capable of receiving and acting on Interrogator commands within a period not exceeding the maximum settling-time interval specified in Table 6 or Table 8, as appropriate (i.e. by the end of within T_s or T_{hs}, respectively).

6.4.1.3.5 Minimum operating field strength and backscatter strength

The tag manufacturer shall specify:

— the free-space sensitivity,

— the minimum relative backscattered modulated power (ASK modulation) or change in radar cross-section or equivalent (phase modulation), and

— the manufacturer's normal operating conditions,

for the tag mounted on one or more manufacturer-selected materials.

Information about the measurement of those parameters is provided in ISO/IEC 18046-3.

6.4.1.4 Transmission order

The transmission order for all R=>T and T=>R communications shall be most-significant bit (MSB) first.

Within each message, the most-significant word shall be transmitted first.

Within each word, the MSB shall be transmitted first.

6.4.1.5 Cyclic Redundancy Check (CRC)

A CRC is a cyclic-redundancy check that a tag uses to ensure the validity of certain R=>T commands, and an Interrogator uses to ensure the validity of certain backscattered T=>R replies. This International Standard uses two CRC types: (i) a CRC-16 and (ii) a CRC-5. Annex F describes both CRC types.

To generate a CRC-16 a tag or Interrogator shall first generate the CRC-16 precursor shown in Table 11, and then take the ones-complement of the generated precursor to form the CRC-16.

A tag or Interrogator shall verify the integrity of a received message that uses a CRC-16. The tag or Interrogator may use of the methods described in Annex F to verify the CRC-16.

At powerup, a Tag calculates and saves into memory a 16-bit StoredCRC - See 6.4.2.10.

Tags shall append a CRC-16 to those replies that use a CRC-16. – See 6.4.2.10 for command specific reply formats.

To generate a CRC-5 an Interrogator shall use the definition in Table 12.

A tag shall verify the integrity of a received message that uses a CRC-5. The tag may use the method as described in Annex F to verify a CRC-5.

Interrogator shall append the appropriate CRC to R=>T transmissions as specified in Table 19.

Table 11 — CRC-16 precursor

CRC-16 precursor				
CRC Type	**Length**	**Polynomial**	**Preset**	**Residue**
ISO/IEC 13239	16 bits	$x^{16} + x^{12} + x^5 + 1$	**$FFFF_h$**	**$1D0F_h$**

Table 12 — CRC-5 definition

CRC-5 Definition				
CRC Type	**Length**	**Polynomial**	**Preset**	**Residue**
—	5 bits	$x^5 + x^3 + 1$	01001_2	00000_2

6.4.1.6 Link timing

Figure 16 illustrates R=>T and T=>R link timing. The figure (not drawn to scale) defines Interrogator interactions with a tag population. Table 13 shows the timing requirements for Figure 16, while 6.4.2.10 describes the commands. Tags and Interrogators shall meet all timing requirements shown in Table 13. RTcal is defined in 6.4.1.2.8; T_{pri} is the T=>R link period (T_{pri} = 1 / BLF). As described in 6.4.1.2.8, an Interrogator shall use a fixed R=>T link rate for the duration of an inventory round; prior to changing the R=>T link rate, an Interrogator shall transmit CW for a minimum of 8 RTcal.

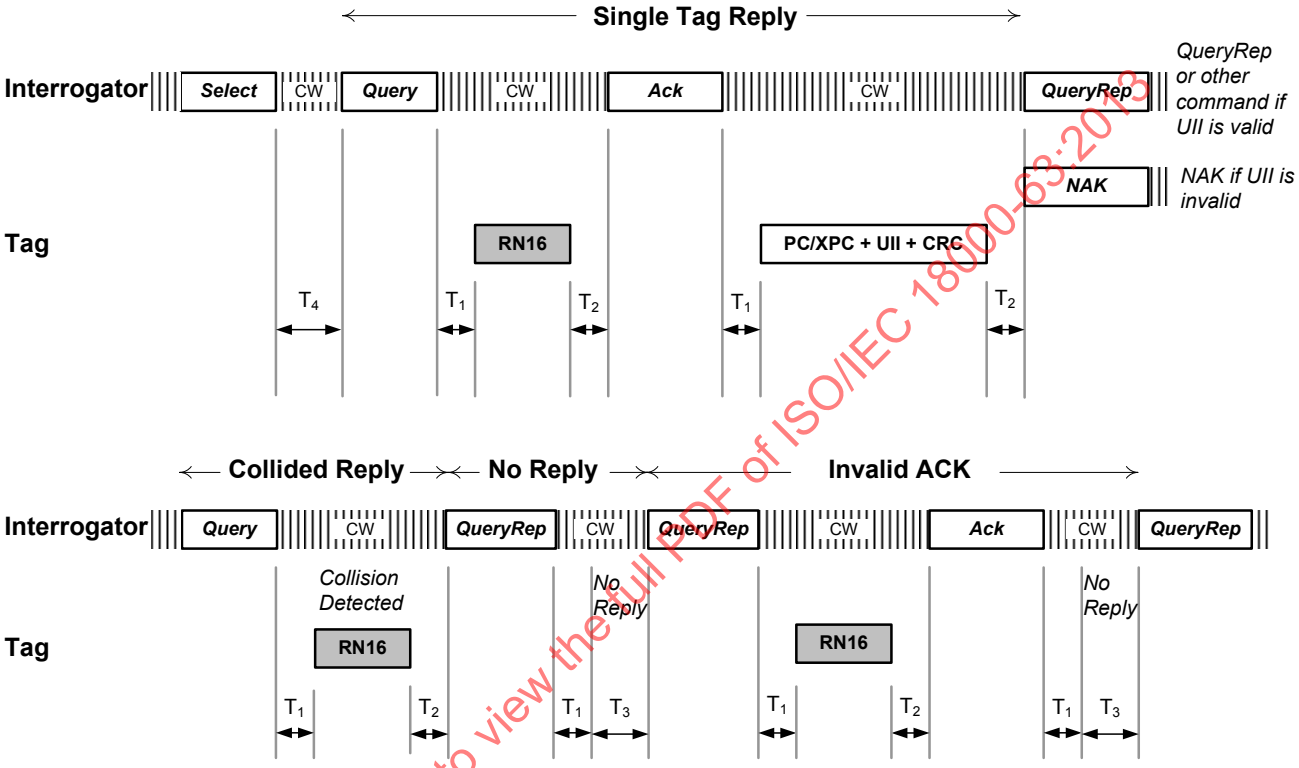


Figure 16 — Link timing

Table 13 — Link timing parameters

Parameter	Minimum	Nominal	Maximum	Description
T_1	MAX(RTcal,10T_{pri}) × (1 – |FT|) – 2µs	MAX(RTcal,10T_{pri})	MAX(RTcal,10T_{pri}) × (1 + |FT|) + 2µs	Time from Interrogator transmission to tag response (specifically, the time from the last rising edge of the last bit of the Interrogator transmission to the first rising edge of the tag response), measured at the tag's antenna terminals.
T_2	3.0T_{pri}		20.0T_{pri}	Interrogator response time required if a tag is to demodulate the Interrogator signal, measured from the end of the last (dummy) bit of the tag response to the first falling edge of the Interrogator transmission.
T_3	0.0T_{pri}			Time an Interrogator waits, after T_1, before it issues another command
T_4	2.0 RTcal			Minimum time between Interrogator commands

The following items apply to the requirements specified in Table 13:

1) T_{pri} denotes either the commanded period of an FM0 symbol or the commanded period of a single subcarrier cycle, as appropriate.

2) A tag may exceed the maximum value for T_1 when responding to commands that write to memory — see, for example, 6.4.2.11.3.3.

3) The maximum value for T_2 shall apply only to tags in the **reply** or **acknowledged** states (see 6.4.2.4.3 and 6.4.2.4.4). For a tag in the **reply** or **acknowledged** states, if T_2 expires (i.e. reaches its maximum value):

   — Without the tag receiving a valid command, the tag shall transition to the **arbitrate** state (see 6.4.2.4.2).

   — During the reception of a valid command, the tag shall execute the command.

   — During the reception of an invalid command, the tag shall transition to **arbitrate** upon determining that the command is invalid.

   — In all other states the maximum value for T_2 shall be unrestricted. A tag shall be allowed a tolerance of 20.0Tpri < T2(max) < 32Tpri in determining whether T2 has expired. “Invalid command” is defined in 6.4.2.11.

4) An Interrogator may transmit a new command prior to interval T_2 (i.e. during a tag response). In this case the responding tag is not required to demodulate or otherwise act on the new command, and may undergo a power-on reset.

5) FT is the frequency tolerance specified in Table 9.

6) T_1+T_3 shall not be less than T_4.

6.4.2 Tag selection, inventory, and access

Tag selection, inventory, and access may be viewed as the lowest level in the data link layer of a layered network communication system.

6.4.2.1 Tag memory

Tag memory shall be logically separated into four distinct banks, each of which may comprise zero or more memory words. A logical memory map is shown in Figure 17. The memory banks are:

— **Reserved memory** shall contain the kill and/or access passwords, if passwords are implemented on the tag. The kill password shall be stored at memory addresses 00_h to $1F_h$; the access password shall be stored at memory addresses 20_h to $3F_h$. See 6.4.2.1.1.

— **UII memory** shall contain a StoredCRC at memory addresses 00_h to $0F_h$, a StoredPC at addresses 10_h to $1F_h$, a code (such as an UII, and hereafter referred to as an UII) that identifies the object to which the tag is or will be attached beginning at address 20_h, and if the tag implements recommissioning (see 6.4.2.1.2) or supports sensors and batteries (see Clause 7) then an Extended Protocol Control word XPC_W1 and an optional Extended Protocol word XPC_W2 (reserved for future use) beginning at address 210_h.

— **TID memory** shall contain an 8-bit ISO/IEC 15963 allocation class identifier at memory locations 00_h to 07_h. TID memory shall contain sufficient identifying information above 07_h for an Interrogator to uniquely identify the custom commands and/or optional features that a tag supports. See 6.4.2.1.3.

— **User memory** is optional. See 6.4.2.1.4.

The logical addressing of all memory banks shall begin at zero (00_h). The physical memory map is vendor-specific. Commands that access memory have a MemBank parameter that selects the bank, and an address parameter, specified using the EBV format described in Annex A, to select a particular memory location within that bank. When tags backscatter memory contents, this backscatter shall fall on word boundaries (except in the case of a truncated reply – see 6.4.2.11.1.1).

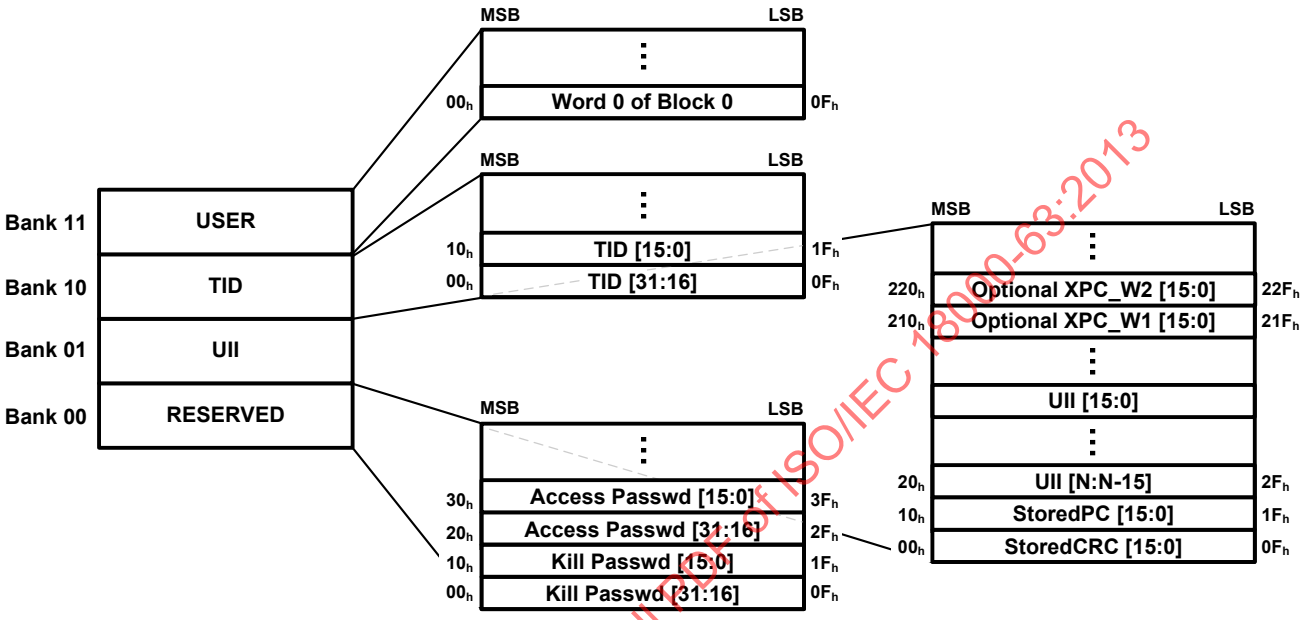


Figure 17 — Logical memory map

NOTE Although Figure 17 uses bit level addressing for illustration, read and write access can only done by addressing on 16-bit word level.

MemBank is defined as follows:

00_2 Reserved
01_2 UII
10_2 TID
11_2 User

Operations in one logical memory bank shall not access memory locations in another bank.

Memory writes, detailed in 6.4.2.9, involve the transfer of 16-bit words from Interrogator to tag. A *Write* command writes 16 bits (i.e. one word) at a time, using link cover-coding to obscure the data during R=>T transmission. The optional *BlockWrite* command writes one or more 16-bit words at a time, without link cover-coding. The optional *BlockErase* command erases one or more 16-bit words at a time. A *Write*, *BlockWrite*, or *BlockErase* shall not alter a tag's killed status regardless of the memory word address (whether valid or invalid) specified in the command.

Interrogators may lock, permanently lock, unlock, or permanently unlock the kill password, access password, UII memory, TID memory, and User memory, thereby preventing or allowing subsequent changes (as appropriate). A tag may optionally have its User memory partitioned into blocks; if it does, then an Interrogator may permanently lock these individual blocks. Recommissioning may alter the memory locking and/or permalocking. See 6.4.2.9 for a description of memory locking and unlocking, and 6.4.2.10 for a description of tag recommissioning. If the kill and/or access passwords are locked they are usable by only the *Kill* and *Access* commands, respectively, and are rendered both unwriteable and unreadable by any other command. Locking or permanently locking the UII, TID, or User memory banks, or permanently locking blocks within the User memory bank, renders the locked memory location unwriteable but leaves it readable.

6.4.2.1.1 Reserved memory bank

Reserved memory contains the kill (see 6.4.2.1.1.1) and/or access (see 6.4.2.1.1.2) passwords, if passwords are implemented on the tag. If a tag does not implement the kill and/or access password(s), the tag shall logically operate as though it has zero-valued password(s) that are permanently read/write locked (see 6.4.2.11.3.5), and the corresponding physical memory locations in Reserved memory need not exist.

6.4.2.1.1.1 Kill password

The kill password is a 32-bit value stored in Reserved memory 00_h to $1F_h$, MSB first. The default (unprogrammed) value shall be zero. An Interrogator may use the kill password to (1) recommission a tag, and/or (2) kill a tag and render it nonresponsive thereafter. A tag shall not execute a recommissioning or kill operation if its kill password is zero. A tag that does not implement a kill password operates as if it has a zero-valued kill password that is permanently read/write locked.

6.4.2.1.1.2 Access password

The access password is a 32-bit value stored in Reserved memory 20_h to $3F_h$, MSB first. The default (unprogrammed) value shall be zero. A tag with a nonzero access password shall require an Interrogator to issue this password before transitioning to the **secured** state. A tag that does not implement an access password operates as if it has a zero-valued access password that is permanently read/write locked.

6.4.2.1.2 UII memory bank

UII memory contains a StoredCRC-16 at memory addresses 00_h to $0F_h$, a StoredPC at memory locations 10_h to $1F_h$, and a UII beginning at address 20_h, and if a tag implements recommissioning then a first XPC word (XPC_W1) at 210_h to $21F_h$ and an optional second XPC word (XPC_W2) at 220_h to $22F_h$. If the tag supports sensors and batteries according to Clauses 7 and 8 then bit-based information is encoded in the first XPC word XPC_W1 at addresses 214_h to 219_h. Refer to 7.6.2 for an overview about PC and XPC_W1 bit definitions. The StoredCRC, StoredPC, UII, and XPC word or words shall be stored MSB first (i.e. the UII's MSB is stored in location 20_h).

The StoredCRC-16 is described in 6.4.2.1.2.1

The StoredPC, as described in 6.4.2.1.2.2, is subdivided into a UII length field in memory locations 10_h to 14_h, a User-memory indicator (UMI) in location 15_h, an XPC indicator (XI) in location 16_h, and a Numbering System Identifier (NSI) in memory locations 17_h to $1F_h$.

The UII is a code that identifies the object to which a tag is affixed. The UII for EPCglobal Applications is described in 6.4.2.1.2.3; the UII for ISO Applications is described in 6.4.2.1.2.4. Interrogators may issue a *Select* command that includes all or part of the UII in the mask. Interrogators may issue an *ACK* command to cause a Tag to backscatter its UII. Under certain circumstances the Tag may truncate its backscattered UII (see 6.4.2.11.1.1). An Interrogator may issue a *Read* command to read all or part of the UII.

The first XPC word XPC_W1, as described in 6.4.2.1.2.2, 6.4.2.1.2.5 and 7.6.2, indicates whether and how a Tag has been recommissioned and/or supports sensors and batteries.

6.4.2.1.2.1 CRC-16 (StoredCRC and PacketCRC)

All Tags shall implement a StoredCRC. Tags that support XPC functionality shall also implement a PacketCRC.

The PC and the UII are protected by the StoredCRC that a Tag backscatters during an inventory operation if PC bits and UII are transmitted to the Interrogator and no additional data, e.g. XPC, is included in the Tag response.

Because Interrogators may issue a *Select* command that includes all or part of this CRC-16 in the mask, and may issue a *Read* command to cause the Tag to backscatter this CRC-16, this CRC-16 is logically mapped

into UII memory. At power-up a Tag shall compute this CRC-16 over (a) the StoredPC and (b) UII specified by the length field in the StoredPC (see 6.4.2.1.2.2) and shall map the computed CRC-16 into UII memory 00_h to $0F_h$, MSB first. Because the {PC+ XPC + UII} is stored in UII memory on word boundaries, this CRC-16 shall be computed on word boundaries. Although Tags include the XI value in the StoredPC in their StoredCRC calculation, regardless of XI value a Tag shall omit XPC_W1 and XPC_W2 from the calculation. A Tag shall finish its StoredCRC computation and memory mapping by the end of interval T_s or T_{hs} (as appropriate) in Figure 3 or Figure 5, respectively. Tags shall not recalculate this CRC-16 for a truncated reply (see 6.4.2.11.1.1).

Table 14 — Tag data and CRC-16 backscattered in response to an ACK command

XI [1]	XEB[1]	Truncation	Tag Backscatter			
			PC	XPC	UII	CRC-16
0	0	Deasserted	StoredPC	None	Full	StoredCRC or PacketCRC
0	0	Asserted	00000_2	None	Truncated	StoredCRC
0	1	Deasserted	Invalid[1]			
0	1	Asserted	Invalid[1]			
1	0	Deasserted	PacketPC	XPC_W1	Full	PacketCRC
1	0	Asserted	00000_2	None	Truncated	StoredCRC
1	1	Deasserted	PacketPC	Both XPC_W1 and XPC_W2	Full	PacketCRC
1	1	Asserted	00000_2	None	Truncated	StoredCRC
[1] See NOTE 1						

NOTE 1 XI is the bitwise logical OR of the 16 bits of XPC_W1, and XEB is the MSB (bit F_h) of XPC_W1, so if XEB=1 then XI=1.

In response to an *ACK* command a Tag backscatters a protocol-control (PC) word (either StoredPC or PacketPC – see 6.4.2.1.2.2), optional XPC word or words (see 6.4.2.1.2.5), UII (see 6.4.2.1.2.3 and 6.4.2.1.2.4), and a CRC-16 that protects the backscattered data. The CRC-16 shall be either (a) the StoredCRC or, alternatively, (b) a PacketCRC that the Tag shall calculate dynamically over the backscattered PC word, optional XPC word or words, and UII. Whether a Tag backscatters its StoredCRC or a PacketCRC shall be as defined in Table 14, depending on the Tag's XI value and whether truncation (see 6.4.2.11.1.1) is asserted or deasserted. Moreover, a PacketCRC shall be used instead of the StoredCRC when a Tag response includes Simple Sensor Data (see 8.3).

Tags that do not support XPC functionality may, but are not required to, implement a PacketCRC.

A Tag shall backscatter its CRC MSB first, regardless of the CRC type.

If XI is asserted then a Tag's PacketCRC is different from its StoredCRC.

As required by 6.4.1.5 an Interrogator shall verify, using a Tag's backscattered CRC-16, the integrity of a received PC word, optional XPC word or words, and UII.

6.4.2.1.2.2 Protocol-control (PC) word (StoredPC and PacketPC)

The PC word contains physical-layer information that a Tag backscatters with its UII during an inventory operation. All Tags shall implement a StoredPC whose fields, comprising UII length, a UMI, an XI, and an NSI, shall be as defined below. Tags that support XPC functionality shall also implement a PacketPC that differs from the StoredPC in its UII length field. The type of PC (StoredPC or PacketPC) that a Tag backscatters in response to an *ACK* shall be as defined in Table 14.

There are 16 PC bits, stored in UII memory at addresses 10_h to $1F_h$, with bit values defined as follows:

Bits 10_h – 14_h: The length of the UII that a Tag backscatters, in words:

00000_2: Zero words.
00001_2: One word (addresses 20h to 2Fh in UII memory).
00010_2: Two words (addresses 20h to 3Fh in UII memory).

•
•
•

11111_2: 31 words (addresses 20h to 20Fh in UII memory).

If a Tag does not support XPC functionality then the maximum value of the UII length field in the StoredPC shall be 11111_2 (allows a 496-bit UII), as shown above. If a Tag supports XPC functionality then the maximum value of the UII length field in the StoredPC shall be 11101_2 (allows a 464-bit UII). A Tag that supports XPC functionality shall ignore a Write or *BlockWrite* command to the StoredPC if the UII length field exceeds 11101_2, and shall instead backscatter an error code (see Annex I).

- Bit 15_h: A User-memory indicator (UMI). If bit 15_h is deasserted then the Tag either does not implement User memory or User memory contains no information. If bit 15_h is asserted then User memory contains information. A Tag may implement the UMI using Method 1 or Method 2 described below, unless the Tag implements block permalocking and/or recommissioning, in which case the Tag shall use Method 1.
    - Method 1: The Tag computes the UMI. At powerup, and prior to calculating its StoredCRC (see 6.4.2.1.2.1), the Tag shall compute the logical OR of bits 03_h to 07_h of User memory and map the computed value into bit 15_h. The Tag shall include the computed UMI value in its StoredCRC calculation (see 6.4.2.1.2.1). If an Interrogator modifies any of bits 03_h to 07_h of User memory then the Tag shall recompute and remap its UMI into bit 15_h. If recommissioning renders User memory inaccessible (see 6.4.2.10) then the Tag shall deassert and remap its UMI into bit 15_h. After remapping the UMI the StoredCRC may be incorrect until the Interrogator power cycles the Tag. The UMI shall not be directly writeable by an Interrogator — when an Interrogator writes the StoredPC, the Tag shall ignore the data value that the Interrogator provides for bit 15_h.
    - Method 2: An Interrogator writes the UMI. If an Interrogator writes a zero value into bits 03_h to 07_h of User memory then it shall deassert bit 15_h. If an Interrogator writes a nonzero value into 03_h to 07_h of User memory then it shall assert bit 15_h. If an Interrogator locks or permalocks UII memory then it shall also lock or permalock, respectively, the word located at address 00_h of User memory, and vice versa. This latter requirement ensures that a condition in which User memory previously contained data but was subsequently erased does not cause a Tag to wrongly indicate the presence of User memory, and vice versa.

NOTE In case the UMI bit is implemented as described in Method 1, then its value will change in dependence of bits 03_h to 07_h of the User Memory independent of whether the UII memory is locked or not.

Bit 16_h: An XPC_W1 indicator (XI). If bit 16_h is deasserted then the Tag either does not implement an XPC_W1 or the XPC_W1 is zero-valued, in which case the Tag shall backscatter its StoredPC or PacketPC, but not an XPC_W1, in response to an *ACK* (see Table 14). If bit 16_h is asserted then the Tag implements an XPC_W1 and one or more bits of XPC_W1 have nonzero values. In this latter case the Tag shall backscatter its XPC_W1 immediately after the StoredPC or PacketPC during inventory.

If a Tag implements an XPC_W1 then at powerup, and prior to calculating its StoredCRC (see 6.4.1.5), the Tag shall compute the bitwise logical OR of its XPC_W1 and map the computed value into bit 16h (i.e. into the XI). A Tag shall use this computed XI value in its StoredCRC calculation. If an Interrogator recommissions the Tag (see 6.4.2.10) then the Tag shall recompute and remap its XI into bit 16h after recommissioning. After recomputing the XI the StoredCRC may be incorrect until the Interrogator power cycles the Tag. The XI bit shall not be directly writeable by an Interrogator — when an Interrogator writes the StoredPC the Tag shall ignore the data value that the Interrogator provides for bit 16h. If an Interrogator reads the PC word using a *Read* command (see 6.4.2.11.3.2) then the Tag's response shall include the XI computed at powerup. If an Interrogator issues a *Select* command (see 6.4.2.11.1.1) with a Mask that includes the XI then the Tag shall use the XI value computed at powerup when determining if Mask is matching or non-matching.

Bits 17_h – $1F_h$: A numbering system identifier (NSI). The MSB of the NSI is stored in memory location 17_h. If bit 17_h contains a logical 0, then the Application is referred to as an EPCglobal Application and PC bits 18_h – $1F_h$ shall be as defined in the EPCglobal Tag Data Standards. If bit 17_h contains a logical 1, then the Application is referred to as an ISO Application and PC bits 18_h – $1F_h$ shall contain the entire AFI defined in ISO/IEC 15961-3. The default value for bits 18_h – $1F_h$ is 00000000_2.

The default (unprogrammed) StoredPC value shall be 0000_h.

During truncated replies a Tag substitutes 00000_2 for the PC word — see 6.4.2.11.1.1.

If an Interrogator modifies the UII length during a memory write, and it wishes the Tag to subsequently backscatter the modified UII, then it must write the length of the new or updated (PC + UII) into the first 5 bits of the Tag's StoredPC, and to ensure a correct StoredCRC, power cycle the Tag. A Tag shall backscatter an error code (see Annex I) if an Interrogator attempts to write a UII length that is not supported by the Tag to the first 5 bits of the Tag's StoredPC.

At power-up a Tag shall compute its StoredCRC over the number of UII words designated by the first 5 bits of the StoredPC word rather than over the length of the entire UII memory (see 6.4.2.11.2.1).

The PacketPC differs from the StoredPC in its UII length field, which a Tag shall adjust to match the length of the backscattered data that follow the PC word. Specifically, if XI is asserted but XEB is not asserted then the Tag backscatters an XPC_W1 before the UII, so the Tag shall add one to (i.e. increment) its UII length field. If both XI and XEB are asserted then the Tag backscatters both an XPC_W1 and an XPC_W2 before the UII, so the Tag shall add two to (i.e. double increment) its UII length field. Because Tags that support XPC functionality have a maximum UII length field of 11101_2, double incrementing will increase the value to 11111_2. A Tag shall not, under any circumstances, allow its UII length field to roll over to 00000_2. Note that incrementing or double incrementing the UII length field does not alter the values stored in bits 10_h – 14_h of UII memory; rather, a Tag increments the UII length field in the backscattered PacketPC but leaves the memory contents unaltered.

Tags that do not support XPC functionality need not implement a PacketPC.

If an Interrogator that does not support an XPC_W1 receives a Tag reply with XI asserted then the Interrogator shall treat the Tag's reply as though its CRC-16 integrity check had failed.

If an Interrogator that does not support an XPC_W2 receives a Tag reply with XEB asserted then the Interrogator shall treat the Tag's reply as though its CRC-16 integrity check had failed.

6.4.2.1.2.3 UII for an EPCglobal Application

The UII structure for an EPCglobal Application shall be as defined in the EPCglobal Tag Data Standards.

6.4.2.1.2.4 UII for an ISO Application

The UII structure for an ISO Application shall be as defined in ISO/IEC 15962.

6.4.2.1.2.5 Extended Protocol Control (XPC) words (optional)

6.4.2.1.2.5.1 General

The optional XPC word shall be used to indicate additional Tag functions.

A Tag may implement an XPC_W1 logically located at addresses 210h to 21Fh of UII memory. If a Tag implements an XPC_W1 then it may additionally implement an XPC_W2 logically located at address 220h to 22Fh of UII memory. A Tag shall not implement an XPC_W2 without also implementing an XPC_W1. If implemented, these XPC words shall be exactly 16 bits in length and are stored MSB first. If a Tag does not support one or both of these optional XPC words then the specified memory locations need not exist.

A Tag shall not implement any non-XPC memory element at UII memory locations 210h to 22Fh, inclusive. This requirement shall apply both to Tags that support an XPC word or words and to those that do not.

If a Tag implements an XPC_W2 then, at powerup, the Tag shall compute the bitwise logical OR of the XPC_W2 and map the computed value into bit 210h of UII memory (i.e. into the most significant bit of XPC_W1). Bit 210h is denoted the XPC Extension Bit (XEB). If a Tag does not implement an XPC_W2 then the XEB shall be zero.

The use of XPC described in this international standard is limited to recommissioning, sensor, and battery functionality. XPC bits not described in this international standard are reserved for future use (RFU).

The use of XPC for recommissioning is described in 6.4.2.1.2.5.2.

The use of XPC for battery functionality is described in Clause 7.

The use of XPC for sensor functionality is described in Clauses 7.6.2 and 8.5.1.

Table 15 describes the allocation of XPC_W1 bits. Bit E_h of XPC_W1 (UII memory location 211h) is reserved for use as a protocol functionality indicator. Furthermore, bits D_h and C_h of XPC_W1 (UII memory locations 212_h and 213_h) are reserved for ISO/IEC 29143.

Table 15 — Allocation of XPC-Bits

UII logical bit address	Bit use	Comment
210	XEB	XPC_W2 indicator
211	RFU	
212	MIIM	Mobile RFID content name indicator (reserved for ISO/IEC 29143)
213		Reserved for ISO/IEC 29143 for future extensions
214	SA	General Alarm
215	SS	Simple Sensor
216	FS	Full Sensor
217	BM	Battery Mode
218		
219	TC	Battery TCRS, see 7.6
21A	RFU	
21B	RFU	
21C	RFU	
21D	Recomm	
21E	Recomm	
21F	Recomm	LSB

XPC_W2 is currently not supported by this international standard and is reserved for future use.

6.4.2.1.2.5.2 XPC for recommissioning

This section assumes that a Tag implements recommissioning.

When this document refers to the 3 least-significant-bits (LSBs) of XPC_W1 it specifically means memory locations $21D_h$, $21E_h$, and $21F_h$ of UII memory.

For virgin Tags the 3 LSBs of the XPC_W1 shall be zero-valued. A Tag writes a nonzero value to one or more of these 3 LSBs during Tag recommissioning (see 6.4.2.10).

A Tag shall not write to the 3 LSBs of XPC_W1 except during Tag recommissioning.

The 3 LSBs of XPC_W1 are not writeable using a *Write* or *BlockWrite*, nor erasable using a *BlockErase*. They can only be asserted by a *Kill* command, meaning that the Tag asserts these bits upon receiving a valid *Kill* command sequence with asserted recommissioning bits (see 6.4.2.11.3.4). If a Tag that does not support XPC receives a *Write*, *BlockWrite*, or *BlockErase* that attempts to write XPC_W1 it responds with an error code (see Annex I).

An Interrogator may issue a *Select* command (see 6.4.2.11.1.1) with a Mask that covers all or part of XPC_W1 and/or XPC_W2. For example, Mask may have the value 000_2 for the 3 LSBs of XPC_W1, in which case recommissioned Tags will be non-matching, as will Tags that do not implement recommissioning; only recommissionable Tags that have not yet been recommissioned will be matching.

An Interrogator may read a Tag's XPC_W1 and XPC_W2 using a *Read* command (see 6.4.2.11.3.2). A recommissionable Tag shall backscatter XPC_W1 regardless of the value of its XI bit. A non-recommissionable Tag shall backscatter an error code (see Annex I).

The following encoding provides a general mapping between the 3 XPC_W1 LSBs and a Tag's recommissioned status. Table 16 provides a detailed mapping between these 3 LSBs and a Tag's recommissioned status:

- **An asserted UII memory location $21F_h$** indicates that block permalocking has been disabled, and any blocks of User memory that were previously block permalocked are no longer block permalocked. An asserted UII memory location $21F_h$ also indicates that the *BlockPermalock* command has been disabled. If a Tag didn't implement block permalocking prior to recommissioning then block permalocking shall remain disabled.
- **An asserted UII memory location $21E_h$** indicates that the User memory bank has been killed. The UII memory location $21E_h$ has precedence over the UII memory location $21F_h$ — if both are asserted then the User memory bank is killed.
- **An asserted UII memory location $21D_h$** indicates that the Tag has unlocked its UII, TID, and User memory banks. The Tag has also write-unlocked its kill and access passwords, and rendered the kill and access passwords permanently unreadable regardless of the values of the Tag's lock bits (see 6.4.2.11.3.5). If an Interrogator subsequently attempts to read the Tag's kill or access passwords the Tag backscatters an error code (see Annex I). Note that portions or banks of Tag memory, if factory set and locked, may not be unlockable regardless of recommissioning. Note also that an Interrogator may subsequently re-lock any memory banks that have been unlocked by recommissioning.
- **XPC_W1 LSBs and a Tag's recommissioned status**

Table 16 — XPC_W1 LSBs and a Tag's recommissioned status

LSBs of XPC_W1	Tag status	Notes
000	1. The Tag has not been recommissioned	1. The Tag's XI bit is deasserted
001	1. Block permalocking has been disabled, and any blocks of User memory that were previously block permalocked are no longer block permalocked 2. The *BlockPermalock* command has been disabled	1. The lock bits are the sole determinant of the User memory bank's lock status (see 6.4.2.11.3.5)
010	1. The User memory bank has been killed 2. A Tag whose XPC_W1 LSBs are 010 acts identically to one whose XPC_W1 LSBs are 011	1. The Tag deasserts its UMI bit 2. User memory is dead, so block permalocking and the *BlockPermalock* command are disabled
011	1. The User memory bank has been killed 2. A Tag whose XPC_W1 LSBs are 011 acts identically to one whose XPC_W1 LSBs are 010	1. The Tag deasserts its UMI bit 2. User memory is dead, so block permalocking and the *BlockPermalock* command are disabled

LSBs of XPC_W1	Tag status	Notes
100	1. The UII, TID, and User memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The kill and access passwords have been rendered permanently unreadable, regardless of the status of the Tag's lock bits (see 6.4.2.11.3.5)	1. If the Tag supports block permalocking then the *BlockPermalock* command remains enabled 2. Any blocks of User memory that were previously block permalocked remain block permalocked, and vice versa 3. Portions or banks of Tag memory, if factory set and locked, may not be unlockable regardless of recommissioning 4. If an Interrogator attempts to read the Tag's kill or access passwords the Tag responds by backscattering an error code (see Annex I) 5. The kill and/or access passwords, as well as one or more memory blocks and/or banks, may be changed and/or relocked after recommissioning
101	1. Block permalocking has been disabled, and any blocks of User memory that were previously block permalocked are no longer block permalocked 2. The *BlockPermalock* command has been disabled 3. The UII, TID, and User memory banks have been unlocked 4. The kill and access passwords have been write-unlocked 5. The kill and access passwords have been rendered permanently unreadable, regardless of the status of the Tag's lock bits (see 6.4.2.11.3.5)	1. The lock bits are the sole determinant of the User memory bank's lock status (see 6.4.2.11.3.5) 2. Portions or banks of Tag memory, if factory set and locked, may not be unlockable regardless of recommissioning 3. If an Interrogator attempts to read the Tag's kill or access passwords the Tag responds by backscattering an error code (see Annex I) 4. The kill and/or access passwords, as well as one or more memory blocks and/or banks, may be changed and/or relocked after recommissioning
110	1. The UII and TID memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The kill and access passwords have been rendered permanently unreadable, regardless of the status of the Tag's lock bits (see 6.4.2.11.3.5) 4. The User memory bank has been killed 5. A Tag whose XPC_W1 LSBs are 110 acts identically to one whose XPC_W1 LSBs are 111	1. Portions or banks of Tag memory, if factory set and locked, may not be unlockable regardless of recommissioning 2. If an Interrogator attempts to read the Tag's kill or access passwords the Tag responds by backscattering an error code (see Annex I) 3. The kill and/or access passwords, as well as one or more memory blocks and/or banks, may be changed and/or relocked after recommissioning 4. The Tag deasserts its UMI bit 5. User memory is dead, so block permalocking and the *BlockPermalock* command are disabled
111	1. The UII and TID memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The kill and access passwords have been rendered permanently unreadable, regardless of the status of the Tag's lock bits (see 6.4.2.11.3.5) 4. The User memory bank has been killed 5. A Tag whose XPC_W1 LSBs are 111 acts identically to one whose XPC_W1 LSBs are 110	1. Portions or banks of Tag memory, if factory set and locked, may not be unlockable regardless of recommissioning 2. If an Interrogator attempts to read the Tag's kill or access passwords the Tag responds by backscattering an error code (see Annex I) 3. The kill and/or access passwords, as well as one or more memory blocks and/or banks, may be changed and/or relocked after recommissioning 4. The Tag deasserts its UMI bit 5. User memory is dead, so block permalocking and the *BlockPermalock* command are disabled

6.4.2.1.3 TID memory bank

TID memory locations 00h to 07h shall contain one of three ISO/IEC 15963 class-identifier values — either $E0_h$,$E2_h$, or $E3_h$. TID memory locations above 07_h shall be defined according to the registration authority defined by this class-identifier value and shall contain, at a minimum, sufficient identifying information for an Interrogator to uniquely identify the custom commands and/or optional features that a Tag supports. TID memory may also contain Tag- and vendor-specific data (for example, a Tag serial number).

NOTE The Tag manufacturer assigns the class-identifier value (i.e. $E0_h$, $E2_h$, or $E3_h$), for which ISO/IEC 15963 defines the registration authorities. The class-identifier does not specify the Application. If the class identifier is $E0_h$, TID memory locations 08_h to $0F_h$ contain an 8-bit manufacturer identifier, TID memory locations 10_h to $3F_h$ contain a 48-bit Tag serial number (assigned by the Tag manufacturer), the composite 64-bit Tag ID (i.e. TID memory 00_h to $3F_h$) is unique among all classes of Tags defined in ISO/IEC 15963, and TID memory is permalocked at the time of manufacture. If the class identifier is $E2_h$, TID memory locations 08_h to 13_h contain a 12-bit Tag mask-designer identifier (obtainable from the registration authority), TID memory locations 14_h to $1F_h$ contain a vendor-defined 12-bit Tag model number, and the usage of Tag memory locations above $1F_h$ is defined in version 1.5 and above of the EPCglobal Tag Data Standards. If the class identifier is E3h, TID memory locations 08h to 0Fh contain an 8-bit manufacturer identifier followed by 2-byte User Memory present and size data, the 48-bit Tag serial number (assigned by the Tag manufacturer), the 1-bit XTID and 15-bit XTID Header data. The composite 80-bit Tag ID (i.e. TID memory 00h to 4Fh) is unique among all classes of Tags defined in ISO/IEC 15963, and TID memory is permalocked at the time of manufacture.

NOTE By convention a number of IC manufacturers use the eight MSBs of the Tag serial number for Allocation Classes E0h as a model number. The eight MSBs of the Tag serial number for Allocation Classes E3h should contain an 8-bit vendor defined model number.

TID memory locations 300_h to $31F_h$ are reserved for ISO/IEC 29143 for the purpose of storing a Mobile RFID Address. Tags that are not claiming conformance to ISO/IEC 29143 shall either set those memory locations to value 0_2 or shall respond to any access command addressing those memory sections by backscattering an according error code, see Annex I.

6.4.2.1.4 User memory bank

A Tag may contain User memory. User memory allows user-specific data storage.

If a Tag's User memory has not yet been programmed then the 5 LSBs of the first word of User memory (i.e. memory addresses $0B_h$ to $0F_h$) shall have the default value 00000_2.

During recommissioning an Interrogator may instruct a Tag to kill its User memory. Killing User memory renders the entire memory bank unreadable, unwriteable, and unselectable. A Tag with killed User memory shall function as though its User memory bank no longer exists.

6.4.2.1.4.1 User memory for an EPCglobal Application

If User memory is included on a Tag then its encoding shall be as defined in the EPCglobal Tag Data Standards.

6.4.2.1.4.2 User memory for an ISO Application

If User memory is included on a Tag then User memory locations 00_h to 07_h shall be the DSFID as defined in ISO/IEC 15961. The encoding of User memory locations above 07_h shall be as defined in ISO/IEC 15962.

6.4.2.2 Sessions and inventoried flags

Interrogators shall support and Tags shall provide 4 sessions (denoted S0, S1, S2, and S3). Tags shall participate in one and only one session during an inventory round. Two or more Interrogators can use sessions to independently inventory a common Tag population. The sessions concept is illustrated in Figure 18.

Tags shall maintain an independent **inventoried** flag for each session. Each of the four **inventoried** flags has two values, denoted *A* and *B*. At the beginning of each and every inventory round an Interrogator chooses to inventory either *A* or *B* Tags in one of the four sessions. Tags participating in an inventory round in one session shall neither use nor modify the **inventoried** flag for a different session. The **inventoried** flags are the only resource a Tag provides separately and independently to a given session; all other Tag resources are shared among sessions.

After singulating a Tag an Interrogator may issue a command that causes the Tag to invert its **inventoried** flag for that session (i.e. $A \rightarrow B$ or $B \rightarrow A$).

The following example illustrates how two Interrogators can use sessions and **inventoried** flags to independently and completely inventory a common Tag population, on a time-interleaved basis:

Interrogator #1 powers-on, then

It initiates an inventory round during which it singulates *A* Tags in session S2 to *B*,

It powers off.

Interrogator #2 powers-on, then

It initiates an inventory round during which it singulates *B* Tags in session S3 to *A*,

It powers off.

This process repeats until Interrogator #1 has placed all Tags in session S2 into *B*, after which it inventories the Tags in session S2 from *B* back to *A*. Similarly, Interrogator #2 places all Tags in session S3 into *A*, after which it inventories the Tags in session S3 from *A* back to *B*. By this multi-step procedure each Interrogator can independently inventory all Tags in its field, regardless of the initial state of their **inventoried** flags.

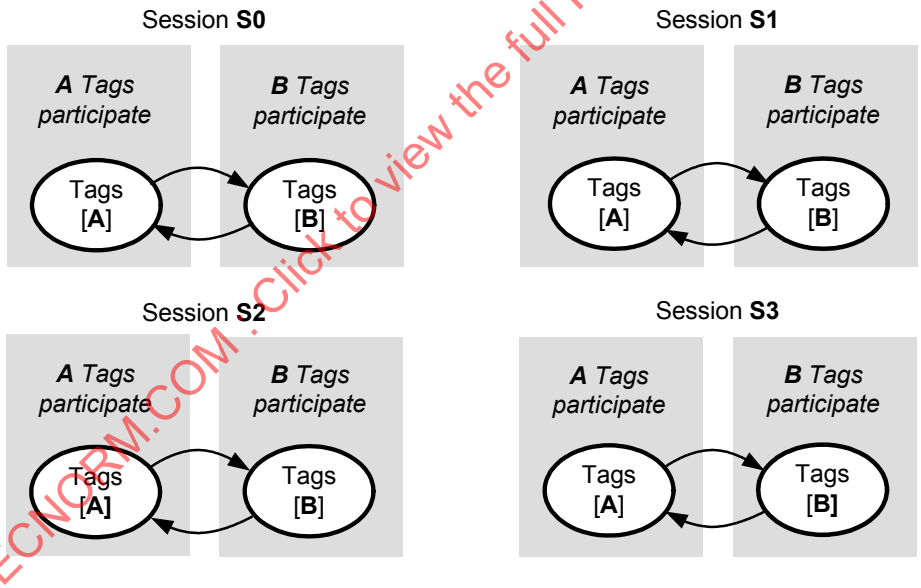


Figure 18 — Session diagram

A Tag's **inventoried** flags shall have the persistence times shown in Table 17. A Tag shall power-up with its **inventoried** flags set as follows:

The S0 **inventoried** flag shall be set to *A*.

The S1 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the flag was set longer in the past than its persistence time, in which case the Tag shall power-up with its S1 **inventoried** flag set to *A*. Because the S1 **inventoried** flag is not automatically refreshed, it may revert from *B* to *A* even when the Tag is powered.

The S2 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the Tag has lost power for a time greater than its persistence time, in which case the Tag shall power-up with the S2 **inventoried** flag set to *A*.

The S3 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the Tag has lost power for a time greater than its persistence time, in which case the Tag shall power-up with its S3 **inventoried** flag set to *A*.

A Tag shall set any of its inventoried flags to either *A* or *B* in 2 ms or less, regardless of the initial flag value. A Tag shall refresh its S2 and S3 flags while powered, meaning that every time a Tag loses power its S2 and S3 **inventoried** flags shall have the persistence times shown in Table 17.

A Tag shall not change the value of its S1 **inventoried** flag from *B* to *A*, as the result of a persistence timeout, while the Tag is participating in an inventory round, is in the midst of being inventoried, or is in the midst of being accessed. If a Tag's S1 flag persistence time expires during an inventory round then the Tag shall change the flag to *A* only as instructed by an Interrogator or at the end of the round; in the latter case, if the subsequent command is a *Query* then the Tag shall invert its S1 flag prior to evaluating the *Query*. If a Tag's S1 flag persistence time expires while the Tag is in the midst of being inventoried or accessed then the Tag shall change the flag to *A* either (i) at the end of the inventory or access operation if the Interrogator instructs it to invert the flag (such as if the Interrogator sends a *QueryRep* with matching session – see 6.4.2.8), or (ii) at the end of the inventory round if the Interrogator instructs it not to invert the flag.

NOTE Battery assisted Tags do not lose power until the battery is depleted and therefore an alternative definition of flag persistence is provided for battery assisted Tags, see 7.3. Moreover, battery assisted Tags may take advantage of the presence of a battery by improving the precision of the maximum flag persistences, see Table 55.

6.4.2.3 Selected flag

The Tag shall implement a **selected** flag, **SL**, which an Interrogator may assert or de-assert using a *Select* command. The Sel parameter in the *Query* command allows an Interrogator to inventory Tags that have **SL** either asserted or de-asserted (i.e. **SL** or ~**SL**), or to ignore the flag and inventory Tags regardless of their **SL** value. **SL** is not associated with any particular session; **SL** may be used in any session and is common to all sessions.

A Tag's **SL** flag shall have the persistence times shown in Table 17. A Tag shall power-up with its **SL** flag either asserted or de-asserted, depending on the stored value, unless the Tag has lost power for a time greater than the **SL** persistence time, in which case the Tag shall power-up with its **SL** flag de-asserted (set to ~**SL**). A Tag shall be capable of asserting or de-asserting its **SL** flag in 2 ms or less, regardless of the initial flag value. A Tag shall refresh its **SL** flag when powered, meaning that every time a Tag loses power its **SL** flag shall have the persistence times shown in Table 17.

Table 17 — Tag flags and persistence values

Flag	Required persistence
S0 **inventoried** flag	Tag energized: Indefinite Tag not energized: None (see NOTE 2)
S1 **inventoried** flag[a]	Tag energized: –40 °C to -25 °C: Not specified –25 °C to +40 °C: 500ms < persistence < 5s +40°C to +65 °C: Not specified Tag not energized: –40 °C to -25 °C: Not specified –25 °C to +40 °C: 500ms < persistence < 5s +40°C to +65 °C: Not specified
S2 **inventoried** flag[a]	Tag energized: Indefinite Tag not energized: –40 °C to -25 °C: Not specified –25 °C to +40 °C: 2s < persistence +40°C to +65 °C: Not specified
S3 **inventoried** flag[a]	Tag energized: Indefinite Tag not energized: –40 °C to -25 °C: Not specified –25 °C to +40 °C: 2s < persistence +40°C to +65 °C: Not specified
Selected (**SL**) flag[a]	Tag energized: Indefinite Tag not energized: –40 °C to -25 °C: Not specified –25 °C to +40 °C: 2s < persistence +40°C to +65 °C: Not specified
NOTE 1 A definition of the meaning of the term “energized” in the context of this document is provided in 4.1. NOTE 2 This corresponds to a value of 0.	

The following requirement applies to those items denoted with a superscript “a” in Table 17: For a randomly chosen and sufficiently large Tag population, 95% of the Tag persistence times shall meet the persistence requirements in Table 17, with a 90% confidence interval.

6.4.2.4 Tag states and slot counter

Tags shall implement the states and the slot counter shown in Figure 19. Annex B shows the associated state-transition tables; Annex C shows the associated command-response tables.

6.4.2.4.1 Ready state

Tags shall implement a **ready** state. **Ready** can be viewed as a “holding state” for energized Tags that are neither killed nor currently participating in an inventory round. Upon entering an energizing RF field a Tag that is not killed shall enter **ready**. The Tag shall remain in **ready** until it receives a *Query* command (see 6.4.2.11.2.1) whose inventoried parameter (for the session specified in the *Query*) and sel parameter match its current flag values. Matching Tags shall draw a *Q*-bit number from their RNG (see 6.4.2.5), load this number into their slot counter, and transition to the **arbitrate** state if the number is nonzero, or to the **reply** state if the number is zero. If a Tag in any state except **killed** loses power it shall return to **ready** upon regaining power.

6.4.2.4.2 Arbitrate state

Tags shall implement an **arbitrate** state. **Arbitrate** can be viewed as a "holding state" for Tags that are participating in the current inventory round but whose slot counters (see 6.4.2.4.8) hold nonzero values. A Tag in **arbitrate** shall decrement its slot counter every time it receives a *QueryRep* command (see 6.4.2.11.2.3) whose session parameter matches the session for the inventory round currently in progress, and it shall transition to the **reply** state and backscatter an RN16 when its slot counter reaches 0000_h. Tags that return to **arbitrate** (for example, from the **reply** state) with a slot value of 0000_h shall decrement their slot counter from 0000_h to $7FFF_h$ at the next *QueryRep* (with matching session) and, because their slot value is now nonzero, shall remain in **arbitrate**.

6.4.2.4.3 Reply state

Tags shall implement a **reply** state. Upon entering **reply** a Tag shall backscatter an RN16. If the Tag receives a valid acknowledgement (*ACK*) it shall transition to the **acknowledged** state, backscattering the reply shown in Table 14. If the Tag fails to receive an *ACK* within time $T_{2(max)}$, or receives an invalid *ACK* or an *ACK* with an erroneous RN16, it shall return to **arbitrate**. Tag and Interrogator shall meet all timing requirements specified in Table 13.

6.4.2.4.4 Acknowledged state

Tags shall implement an **acknowledged** state. A Tag in **acknowledged** may transition to any state except **killed**, depending on the received command (see Figure 19). If a Tag in the **acknowledged** state receives a valid *ACK* containing the correct RN16 it shall re-backscatter the reply shown in Table 14. If a Tag in the **acknowledged** state fails to receive a valid command within time $T_{2(max)}$ it shall return to **arbitrate**. Tag and Interrogator shall meet all timing requirements specified in Table 13.

6.4.2.4.5 Open state

Tags shall implement an **open** state. A Tag in the **acknowledged** state whose access password is nonzero shall transition to **open** upon receiving a *Req_RN* command, backscattering a new RN16 (denoted handle) that the Interrogator shall use in subsequent commands and the Tag shall use in subsequent replies. Tags in the **open** state can execute all access commands except *Lock* and *BlockPermalock*. A Tag in **open** may transition to any state except **acknowledged**, depending on the received command (see Figure 19). If a Tag in the **open** state receives a valid *ACK* containing the correct handle it shall re-backscatter the reply shown in Table 14. Tag and Interrogator shall meet all timing requirements specified in Table 13 except $T_{2(max)}$; in the **open** state the maximum delay between Tag response and Interrogator transmission is unrestricted.

6.4.2.4.6 Secured state

Tags shall implement a **secured** state. A Tag in the **acknowledged** state whose access password is zero shall transition to **secured** upon receiving a *Req_RN* command, backscattering a new RN16 (denoted handle) that the Interrogator shall use in subsequent commands and the Tag shall use in subsequent replies. A Tag in the **open** state whose access password is nonzero shall transition to **secured** upon receiving a valid *Access* command sequence, maintaining the same handle that it previously backscattered when it transitioned from the **acknowledged** state to the **open** state. Tags in the **secured** state can execute all access commands. A Tag in **secured** may transition to any state except **open** or **acknowledged**, depending on the received command (see Figure 19). If a Tag in the **secured** state receives a valid *ACK* containing the correct handle it shall re-backscatter the reply shown in Table 14. Tag and Interrogator shall meet all timing requirements specified in Table 13 except $T_{2(max)}$; in the **secured** state the maximum delay between Tag response and Interrogator transmission is unrestricted.

6.4.2.4.7 Killed state

Tags shall implement a **killed** state. A Tag in either the **open** or **secured** states shall enter the **killed** state upon receiving a valid *Kill* command sequence (see 6.4.2.11.3.4) with a valid nonzero kill password, zero-valued Recom bits (see 6.4.2.10), and valid handle. If a Tag does not implement recommissioning then it treats nonzero Recom bits as though Recom = 0. *Kill* permanently disables a Tag. Upon entering the **killed** state a Tag shall notify the Interrogator that the kill operation was successful, and shall not respond to an Interrogator thereafter. Killed Tags shall remain in the **killed** state under all circumstances, and shall immediately enter killed upon subsequent power-ups. Killing a Tag is irreversible.

6.4.2.4.8 Slot counter

Tags shall implement a 15-bit slot counter. Upon receiving a *Query* or *QueryAdjust* command a Tag shall preload into its slot counter a value between 0 and 2^Q–1, drawn from the Tag's RNG (see 6.4.2.5). *Q* is an integer in the range (0,15). A *Query* specifies *Q*; a *QueryAdjust* may modify *Q* from the prior *Query*.

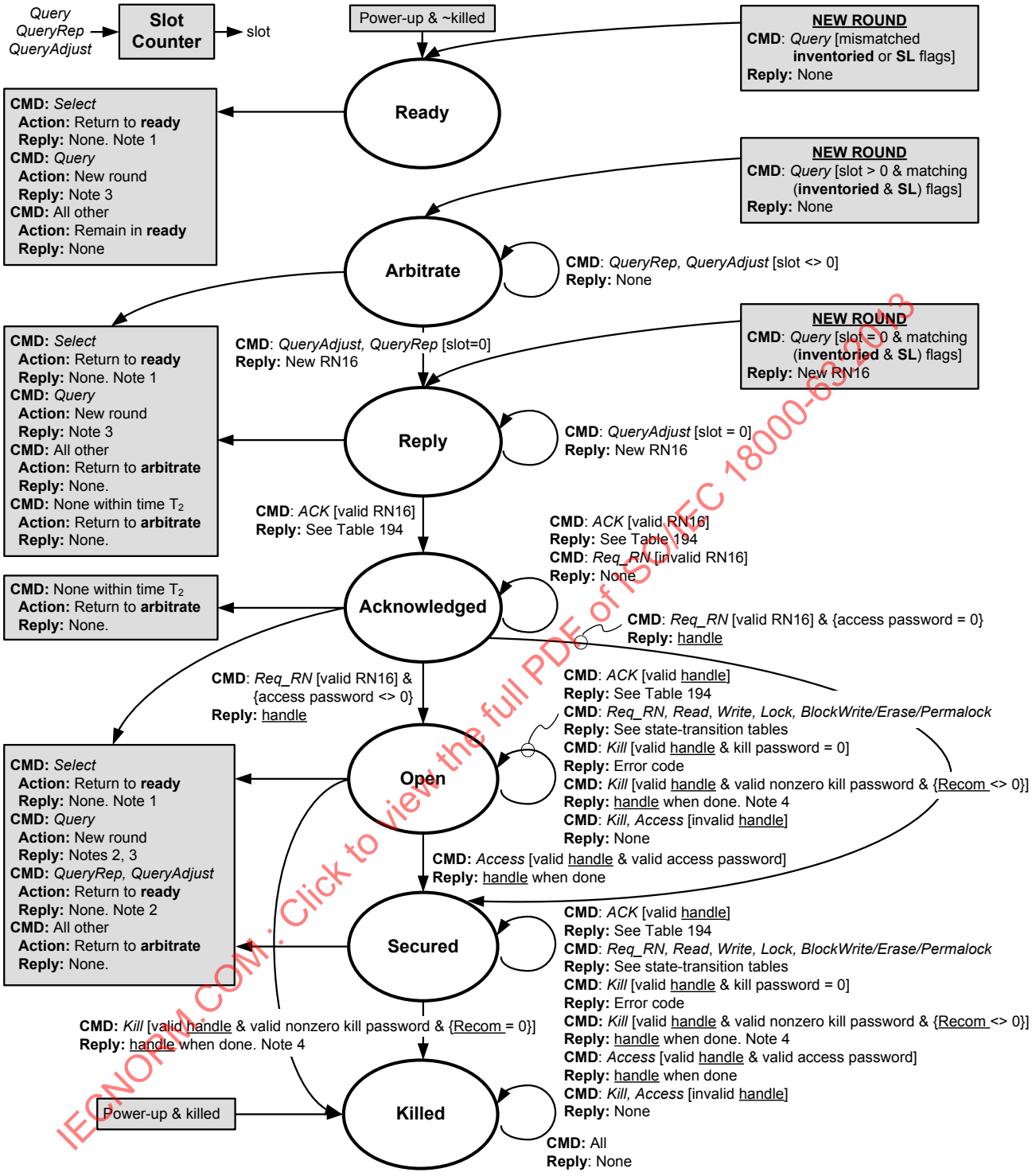
Tags in the **arbitrate** state decrement their slot counter every time they receive a *QueryRep* with matching session, transitioning to the **reply** state and backscattering an RN16 when their slot counter reaches 0000_h. Tags whose slot counter reached 0000_h, who replied, and who were not acknowledged (including Tags that responded to an original *Query* and were not acknowledged) shall return to **arbitrate** with a slot value of 0000_h and shall decrement this slot value from 0000_h to $7FFF_h$ at the next *QueryRep*. The slot counter shall be capable of continuous counting, meaning that, after the slot counter rolls over to $7FFF_h$ it begins counting down again, thereby effectively preventing subsequent replies until the Tag loads a new random value into its slot counter. See also Annex J.

6.4.2.5 Tag random or pseudo-random number generator

Tags shall implement a random or pseudo-random number generator (RNG). The RNG shall meet the following randomness criteria independent of the strength of the energizing field, the R=>T link rate, and the data stored in the Tag (including but not limited to the StoredPC, XPC word or words, UII, and StoredCRC). Tags shall generate 16-bit random or pseudo-random numbers (RN16) using the RNG, and shall have the ability to extract *Q*-bit subsets from an RN16 to preload the Tag's slot counter (see 6.4.2.4.8). Tags shall have the ability to temporarily store at least two RN16s while powered, to use, for example, as a handle and a 16-bit cover-code during password transactions (see Figure 23 or Figure 25).

> **Probability of a single RN16:** The probability that any RN16 drawn from the RNG has value RN16 = *j*, for any *j*, shall be bounded by $0.8/2^{16} < P(RN16 = j) < 1.25/2^{16}$.
>
> **Probability of simultaneously identical sequences:** For a Tag population of up to 10,000 Tags, the probability that any two or more Tags simultaneously generate the same sequence of RN16s shall be less than 0.1%, regardless of when the Tags are energized.
>
> **Probability of predicting an RN16:** An RN16 drawn from a Tag's RNG 10ms after the end of T_r in Figure 3 shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from the RNG, performed under identical conditions, are known.

This document recommends that Interrogators wait 10ms after T_r in Figure 3 or T_{hr} in Figure 5 before issuing passwords to Tags.



NOTES

1. *Select*: Assert/deassert **SL** or set **inventoried** to *A* or *B*.
2. *Query*: *A*→ *B* or *B*→ *A* if the new session matches the prior session; otherwise no change to the **inventoried** flag.
   QueryRep/QueryAdjust: *A*→ *B* or *B*→ *A* if the session matches the prior *Query*; otherwise, the command is invalid and ignored by the Tag.
3. *Query* starts a new round and may change the session. Tags may go to **ready**, **arbitrate**, or **reply**.
4. If a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.

Figure 19 — Tag state diagram

6.4.2.6 Managing Tag populations

Interrogators manage Tag populations using the three basic operations shown in Figure 20. Each of these operations comprises one or more commands. The operations are defined as follows:

Select: The process by which an Interrogator selects a Tag population for inventory and access. Interrogators may use one or more *Select* commands to select a particular Tag population prior to inventory.

Inventory: The process by which an Interrogator identifies Tags. An Interrogator begins an inventory round by transmitting a *Query* command in one of four sessions. One or more Tags may reply. The Interrogator detects a single Tag reply and requests the PC word, optional first XPC word XPC_W1 (if XI is asserted), optional second XPC word XPC_W2 (if XEB is asserted), UII, and CRC-16 from the Tag. An inventory round operates in one and only one session at a time. Annex E shows an example of an Interrogator inventorying and accessing a single Tag.

Access: The process by which an Interrogator transacts with (reads from or writes to) individual Tags. An individual Tag must be uniquely identified prior to access. Access comprises multiple commands, some of which employ one-time-pad based cover-coding of the R=>T link.

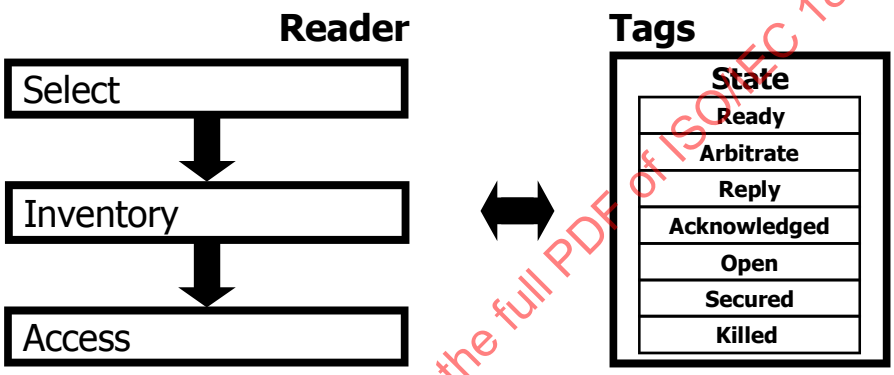


Figure 20 — Interrogator/Tag operations and Tag state

6.4.2.7 Selecting Tag populations

The selection process employs a single command, *Select*, which an Interrogator may apply successively to select a particular Tag population based on user-defined criteria, enabling union (U), intersection (∩), and negation (~) based Tag partitioning. Interrogators perform U and ∩ operations by issuing successive *Select* commands. *Select* can assert or de-assert a Tag's **SL** flag, or it can set a Tag's **inventoried** flag to either *A* or *B* in any one of the four sessions. *Select* contains the parameters Target, Action, MemBank, Pointer, Length, Mask, and Truncate.

Target and Action indicate whether and how a *Select* modifies a Tag's **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. A *Select* that modifies **SL** shall not modify **inventoried**, and vice versa.

MemBank specifies if the mask applies to UII, TID, or User memory. *Select* commands apply to a single memory bank. Successive *Selects* may apply to different memory banks.

Pointer, Length, and Mask: Pointer and Length describe a memory range. Mask, which must be Length bits long, contains a bit string that a Tag compares against the specified memory range.

Truncate specifies whether a Tag backscatters its entire UII, or only that portion of the UII immediately following Mask. Truncated UIIs are always followed by the Tag's StoredCRC (see Table 14). A Tag does not re-calculate its StoredCRC for a truncated reply. Truncated replies never include the XPC bits specified in 6.4.2.1.2.5.

By issuing multiple identical *Select* commands an Interrogator can asymptotically single out all Tags matching the selection criteria even though Tags may undergo short-term RF fades.

A *Query* command uses **inventoried** and **SL** to decide which Tags participate in an inventory. Interrogators may inventory and access **SL** or **~SL** Tags, or they may choose to ignore the **SL** flag entirely.

6.4.2.8 Inventorying Tag populations

The inventory command set includes *Query*, *QueryAdjust*, *QueryRep*, *ACK*, and *NAK*. *Query* initiates an inventory round and decides which Tags participate in the round (“inventory round” is defined in 6.2.2).

Query contains a slot-count parameter *Q*. Upon receiving a *Query* participating Tags pick a random value in the range (0, 2^Q-1), inclusive, and load this value into their slot counter. Tags that pick a zero transition to the **reply** state and reply immediately. Tags that pick a nonzero value transition to the **arbitrate** state and await a *QueryAdjust* or a *QueryRep* command. Assuming that a single Tag replies, the query-response algorithm proceeds as follows:

The Tag backscatters an RN16 as it enters **reply**,

The Interrogator acknowledges the Tag with an *ACK* containing this same RN16,

The acknowledged Tag transitions to the **acknowledged** state, backscattering the reply shown in Table 14,

The Interrogator issues a *QueryAdjust* or *QueryRep,* causing the identified Tag to invert its **inventoried** flag (i.e. *A*→*B* or *B*→*A*) and transition to **ready**, and potentially causing another Tag to initiate a query-response dialog with the Interrogator, starting in step (a), above.

If the Tag fails to receive the *ACK* in step (b) within time T_2 (see Figure 16), or receives the *ACK* with an erroneous RN16, it returns to **arbitrate**.

If multiple Tags reply in step (a) but the Interrogator, by detecting and resolving collisions at the waveform level, can resolve an RN16 from one of the Tags, the Interrogator can *ACK* the resolved Tag. Unresolved Tags receive erroneous RN16s and return to **arbitrate** without backscattering the reply shown in Table 14.

If the Interrogator sends a valid *ACK* (i.e. an *ACK* containing the correct RN16) to the Tag in the **acknowledged** state the Tag re-backscatters the reply shown in Table 14.

At any point the Interrogator may issue a *NAK*, in response to which all Tags in the inventory round that receive the *NAK* return to **arbitrate** without changing their **inventoried** flag.

After issuing a *Query* to initiate an inventory round, the Interrogator typically issues one or more *QueryAdjust* or *QueryRep* commands. *QueryAdjust* repeats a previous *Query* and may increment or decrement *Q*, but does not introduce new Tags into the round. *QueryRep* repeats a previous *Query* without changing any parameters and without introducing new Tags into the round. An inventory round can contain multiple *QueryAdjust* or *QueryRep* commands. At some point the Interrogator will issue a new *Query*, thereby starting a new inventory round.

Tags in the **arbitrate** or **reply** states that receive a *QueryAdjust* first adjust *Q* (increment, decrement, or leave unchanged), then pick a random value in the range (0, 2^Q-1), inclusive, and load this value into their slot counter. Tags that pick zero transition to the **reply** state and reply immediately. Tags that pick a nonzero value transition to the **arbitrate** state and await a *QueryAdjust* or a *QueryRep* command.

Tags in the **arbitrate** state decrement their slot counter every time they receive a *QueryRep*, transitioning to the **reply** state and backscattering an RN16 when their slot counter reaches 0000_h. Tags whose slot counter reached 0000_h, who replied, and who were not acknowledged (including Tags that responded to the original *Query* and were not acknowledged) return to **arbitrate** with a slot value of 0000_h and decrement this slot value from 0000_h to $7FFF_h$ at the next *QueryRep*, thereby effectively preventing subsequent replies until the Tag loads a new random value into its slot counter.

Although Tag inventory is based on a random protocol, the *Q*-parameter affords network control by allowing an Interrogator to regulate the probability of Tag responses. *Q* is an integer in the range (0,15); thus, the associated Tag-response probabilities range from $2^0 = 1$ to $2^{-15} = 0.000031$.

Annex D describes an exemplary Interrogator algorithm for choosing *Q*.

The scenario outlined above assumed a single Interrogator operating in a single session. However, as described in 6.4.2.2, an Interrogator can inventory a Tag population in one of four sessions. Furthermore, as described in 6.4.2.11.2, the *Query*, *QueryAdjust*, and *QueryRep* commands each contain a session parameter. How a Tag responds to these commands varies with the command, session parameter, and Tag state, as follows:

Query: A *Query* command starts an inventory round and chooses the session for the round. Tags in any state except **killed** execute a *Query*, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure 19).

If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter matches the prior session it inverts its **inventoried** flag (i.e. *A*→*B* or *B*→*A*) for the session before it evaluates whether to transition to **ready**, **arbitrate**, or **reply**.

If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter does not match the prior session it leaves its **inventoried** flag for the prior session unchanged as it evaluates whether to transition to **ready**, **arbitrate**, or **reply**.

QueryAdjust, QueryRep: Tags in any state except **ready** or **killed** execute a *QueryAdjust* or *QueryRep* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the Tag is not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively). Tags ignore a *QueryAdjust* or *QueryRep* with mismatched session.

If a Tag in the **acknowledged**, **open**, or **secured** states receives a *QueryAdjust* or *QueryRep* whose session parameter matches the session parameter in the prior *Query*, and the Tag is not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively), it inverts its **inventoried** flag (i.e. *A*→*B* or *B*→*A*) for the current session then transitions to **ready**.

To illustrate an inventory operation, consider a specific example. Assume a population of 64 powered Tags in the **ready** state. An Interrogator first issues a *Select* to select a subpopulation of Tags. Assume that 16 Tags match the selection criteria. Further assume that 12 of the 16 selected Tags have their **inventoried** flag set to *A* in session S0. The Interrogator issues a *Query* specifying (**SL**, *Q* = 4, S0**,** *A*). Each of the 12 Tags picks a random number in the range (0,15) and loads the value into its slot counter. Tags that pick a zero respond immediately. The *Query* has 3 possible outcomes:

No Tags reply: The Interrogator may issue another *Query*, or it may issue a *QueryAdjust* or *QueryRep*.

One Tag replies (see Figure 21): The Tag transitions to the **reply** state and backscatters an RN16. The Interrogator acknowledges the Tag by sending an *ACK*. If the Tag receives the *ACK* with a correct RN16 it backscatters the reply shown in Table 14 and transitions to the **acknowledged** state. The diagram in Figure 21 assumes that XI is deasserted. If the Tag receives the *ACK* with an incorrect RN16 it transitions to **arbitrate**. Assuming a successful *ACK*, the Interrogator may either access the acknowledged Tag or issue a *QueryAdjust* or *QueryRep* with matching session parameter to invert the Tag's **inventoried** flag from *A*→*B* and send the Tag to **ready** (a *Query* with matching prior-round session parameter will also invert the **inventoried** flag from *A*→*B*).

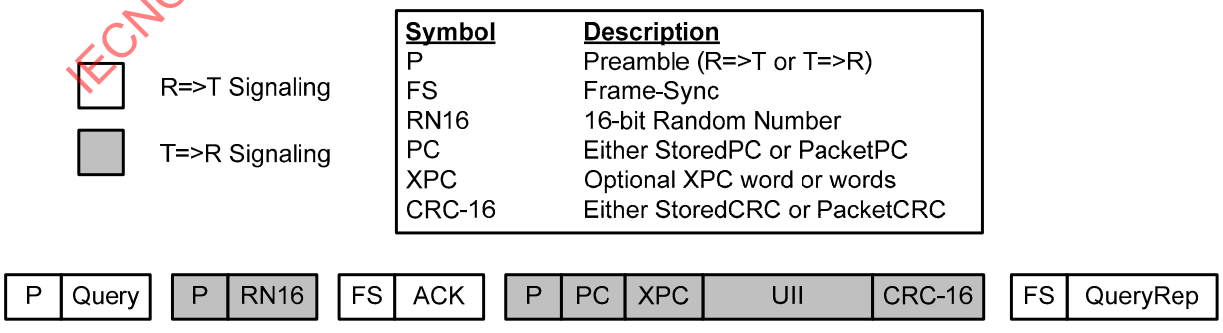


Figure 21 — One Tag reply

Multiple Tags reply: The Interrogator observes a backscattered waveform comprising multiple RN16s. It may try to resolve the collision and issue an *ACK*; not resolve the collision and issue a *QueryAdjust, QueryRep*, or *NAK*; or quickly identify the collision and issue a *QueryAdjust* or *QueryRep* before the collided Tags have finished backscattering. In the latter case the collided Tags, not observing a valid reply within T_2 (see Figure 16), return to **arbitrate** and await the next *Query* or *QueryAdjust* command.

6.4.2.9 Accessing individual Tags

After acknowledging a Tag, an Interrogator may choose to access it. The access command set comprises *Req_RN, Read, Write*, *Kill*, *Lock*, *Access*, *BlockWrite*, and *BlockErase*, and *BlockPermalock*. The Tag executes access commands in the states according to Table 18.

Table 18 — Access commands and Tag states in which they are permitted

Command	Acknowledged State	Open State	Secured State	Remark
Req_RN	Permitted	Permitted	Permitted	
Read		Permitted	Permitted	
Write		Permitted	Permitted	Requires prior *Req_RN*
Kill		Permitted	Permitted	Requires prior *Req_RN*
Lock			Permitted	
Access		Permitted	Permitted	Optional command; Requires prior *Req_RN*
BlockWrite		Permitted	Permitted	Optional command
BlockErase		Permitted	Permitted	Optional command
BlockPermalock			Permitted	Optional command

An Interrogator accesses a Tag in the **acknowledged** state as follows:

Step 1. The Interrogator issues a *Req_RN* to the acknowledged Tag.

Step 2. The Tag generates and stores a new RN16 (denoted handle), backscatters the handle, and transitions to the **open** state if its access password is nonzero, or to the **secured** state if its access password is zero. The Interrogator may now issue further access commands.

All access commands issued to a Tag in the **open** or **secured** states include the Tag's handle as a parameter in the command. When in either of these two states, Tags verify that the handle is correct prior to executing an access command, and ignore access commands with an incorrect handle. The handle value is fixed for the entire duration of a Tag access.

Tags in the **open** state can execute all access commands except *Lock* and *BlockPermalock*. Tags in the **secured** state can execute all access commands. A Tag's response to an access command includes, at a minimum, the Tag's handle; the response may include other information as well (for example, the result of a *Read* operation).

An Interrogator may issue an *ACK* to a Tag in the **open** or **secured** states, causing the Tag to backscatter the reply shown in Table 14.

Interrogator and Tag can communicate indefinitely in the **open** or **secured** states. The Interrogator may terminate the communications at any time by issuing a *Query*, *QueryAdjust*, *QueryRep*, or a *NAK*. The Tag's response to a *Query*, *QueryAdjust*, or *QueryRep* is described in 6.4.2.8. A *NAK* causes all Tags in the inventory round to return to **arbitrate** without changing their **inventoried** flag.

The *Write*, *Kill*, and *Access* commands send 16-bit words (either data or half-passwords) from Interrogator to Tag. These commands use one-time-pad based link cover-coding to obscure the word being transmitted, as follows:

Step 1. The Interrogator issues a *Req_RN,* to which the Tag responds by backscattering a new RN16. The Interrogator then generates a 16-bit ciphertext string comprising a bit-wise EXOR of the 16-bit word to be transmitted with this new RN16, both MSB first, and issues the command with this ciphertext string as a parameter.

Step 2. The Tag decrypts the received ciphertext string by performing a bit-wise EXOR of the received 16-bit ciphertext string with the original RN16.

If an Interrogator issues a command containing cover-coded data or a half-password and fails to receive a response from the Tag, the Interrogator may reissue the command unchanged. If the Interrogator issues a command with new data or new half-password, then Interrogator shall first issue a *Req_RN* to obtain a new RN16 and shall use this new RN16 for the cover-coding.

To reduce security risks, this document recommends that (1) Tags use unique kill passwords and (2) memory writes be performed in a secure location.

The *BlockWrite* command (see 6.4.2.11.3.7) communicates multiple 16-bit words from Interrogator to Tag. Unlike *Write*, *BlockWrite* does not use link cover-coding.

A Tag responds to a command that writes or erases memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite*, and *BlockErase* and *BlockPermalock* with Read/Lock=1 (see 6.4.2.11.3.9)) by backscattering its handle, indicating that the operation was successful, or by backscattering an error code (see Annex I), indicating that the operation was unsuccessful. The Tag reply uses the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. the Tag replies as if TRext=1 regardless of the TRext value specified in the *Query* command that initiated the inventory round). See 6.4.2.11.3 for detailed descriptions of a Tag's reply to each particular access command.

Issuing an access password to a Tag is a multi-step procedure, described in 6.4.2.11.3.6 and outlined in Figure 25.

Tag memory may be unlocked or locked. The lock status may be changeable or permalocked (i.e. permanently unlocked or permanently locked). Recommissioning the Tag may change the lock status, even if the memory was previously permalocked. An Interrogator may write to unlocked memory from either the **open** or **secured** states. An Interrogator may write to locked memory that is not permalocked from the **secured** state only. See Table 44, 6.4.2.10, 6.4.2.11.3.5, 6.4.2.11.3.9, Table 44, and Table 51 for a detailed description of memory lock, permalock, recommissioning, and the Tag state required to modify memory.

This protocol recommends that Interrogators avoid powering-off while a Tag is in the **reply**, **acknowledged**, **open** or **secured** states. Rather, Interrogators should end their dialog with a Tag before powering off, leaving the Tag in either the **ready** or **arbitrate** state.

6.4.2.10 Killing or recommissioning a Tag

Killing or recommissioning a Tag is a multi-step procedure, described in 6.4.2.11.3.4 and shown in Figure 23, in which an Interrogator sends two successive *Kill* commands to a Tag. The first *Kill* command contains the first half of the kill password, and the second *Kill* command contains the second half. Each *Kill* command also contains 3 RFU/Recom bits. In the first *Kill* command these bits are RFU and zero valued; in the second *Kill* command they are called *recommissioning* (or Recom) bits and may be nonzero valued. The procedures for killing or recommissioning a Tag are identical, except that the recommissioning bits in the second *Kill* command are zero when killing a Tag and are nonzero when recommissioning it. Regardless of the intended operation, a Tag shall not kill or recommission itself without first receiving the correct kill password by the procedure shown in Figure 23.

If a Tag does not implement recommissioning then it shall ignore the recommissioning bits and treat them as though they were zero, meaning that, if the Tag receives a properly formatted *Kill* command sequence with the correct kill password it shall kill itself dead regardless of the values of the recommissioning bits. The remainder of this section 6.4.2.10 assumes that a Tag implements recommissioning.

Upon receiving a properly formatted *Kill* command sequence with the correct kill password and one or more nonzero recommissioning bits, a Tag shall assert those LSBs of its XPC_W1 that are asserted in the recommissioning bits (for example, if a Tag receives 100_2 for the recommissioning bits then it asserts the UII memory location $21D_h$ of its XPC_W1 word). The XPC_W1 LSBs shall be one-time-writeable, meaning that they cannot be deasserted after they are asserted. By storing the Tag's recommissioned status in the first XPC word a subsequent Interrogator can know how the Tag was recommissioned (see Table 16). A Tag shall perform the following operations based on the recommissioning bit values it receives (see also 6.4.2.1.2.5):

- **Asserted UII memory location $21F_h$:** The Tag shall disable block permalocking and unlock any blocks of User memory that were previously permalocked. The Tag shall disable support for the *BlockPermalock* command. If the Tag didn't implement block permalocking prior to recommissioning then block permalocking shall remain disabled. The lock status of User memory shall be determined solely by the lock bits (see 6.4.2.11.3.5).
- **Asserted UII memory location $21E_h$:** The Tag shall kill its User memory, rendering the memory bank unreadable, unwriteable, and unselectable (i.e. the Tag functions as though its User memory bank no longer exists).The UII memory location $21E_h$ has precedence over the UII memory location $21F_h$ — if both are asserted then the User memory bank shall be killed.
- **Asserted UII memory location $21D_h$:** The Tag shall unlock its UII, TID, and User memory banks, regardless of whether these banks were locked or permalocked. Portions of User memory that were block permalocked shall remain block permalocked, and vice versa, unless the UII memory location $21F_h$ is also asserted, in which case the Tag shall unlock its permalocked blocks. The Tag shall write-unlock its kill and access passwords, and shall render the kill and access passwords permanently unreadable regardless of the values of the Tag's lock bits (see 6.4.2.11.3.5). If an Interrogator subsequently attempts to read the Tag's kill or access passwords the Tag shall backscatter an error code (see Annex I). A Tag that receives a subsequent *Lock* command with pwd-read/write=0 shall lock or permalock the indicated password(s) in the writeable state, but the passwords shall still remain unreadable.

A Tag shall not execute any of the above recommissioning operations more than once. As one example, a Tag does not allow any of its memory banks to be unlocked more than once by recommissioning.

A Tag may execute multiple *Kill* command sequences, depending on the nature and ordering of the operations specified in these command sequences. Specifically:

- A Tag that is killed dead shall not allow a subsequent recommissioning.
- A Tag that has been recommissioned may be subsequently killed dead.
- A Tag that receives a properly formatted *Kill* command sequence with the correct kill password but with redundant recommissioning bits (for example, the recommissioning bits are 100_2 but the Tag's XPC_W1 already contains 100_2) shall not perform the requested recommissioning operation again. Instead, the Tag shall merely verify that its XPC word contains the asserted values and respond affirmatively to the Interrogator. An Interrogator may choose to send a *Kill* sequence with redundant recommissioning bits if, for example, it had sent a prior *Kill* sequence but did not observe an affirmative response from the Tag.
- A Tag that receives a properly formatted *Kill* command sequence with the correct kill password and with newly asserted recommissioning bits shall perform the recommissioning operation indicated by the newly asserted bits, responding affirmatively to the Interrogator when done. A Tag shall compute the logical OR of the UII memory location $21F_h$ to $21D_h$ of its current XPC_W1 and the recommissioning bits and store the resulting value into its XPC_W1. For example, if a Tag whose LSB XPC_W1 bits are 100_2 receives a *Kill* sequence whose recommissioning bits are 010_2, the Tag renders its User memory bank inaccessible and stores 110_2 into its XPC_W1 LSBs.

An Interrogator may subsequently re-lock any of the memory banks or passwords that have been unlocked by recommissioning.

Portions or entire banks of Tag memory, if factory locked, may not be unlockable by recommissioning. An Interrogator may determine whether a Tag supports recommissioning, and if so which (if any) memory portions are not recommissionable, by reading the Tag's TID memory prior to recommissioning.

A Tag that does not implement a kill password, or a Tag whose kill password is zero, shall not execute a kill or recommissioning operation. Such a Tag shall respond with an error code (as shown in Figure 23) to a *Kill* command sequence regardless of the RFU or recommissioning bit settings.

A Tag shall accept all eight possible combinations of the 3 recommissioning bits, executing those portions that it is capable of executing, ignoring those it cannot, and responding affirmatively to the Interrogator when done. Several examples of operations that a Tag may be incapable or partially capable of executing are:

- A Tag that does not have User memory cannot unlock it
- A Tag that does not implement an Access password cannot unlock it for writing
- A Tag in which a portion of TID memory is factory locked cannot unlock this portion

6.4.2.11 Interrogator commands and Tag replies

Interrogator-to-Tag commands shall have the format shown in Table 19.

QueryRep and *ACK* have 2-bit command codes beginning with 0_2.

Query, *QueryAdjust*, and *Select* have 4-bit command codes beginning with 10_2.

All other base commands have 8-bit command codes beginning with 110_2.

All extended commands have 16-bit command codes beginning with 1110_2.

QueryRep, *ACK*, *Query*, *QueryAdjust,* and *NAK* have the unique command lengths shown in Table 19. No other commands shall have these lengths. If a Tag receives one of these commands with an incorrect length it shall ignore the command.

Query, QueryAdjust, and *QueryRep* contain a session parameter.

Query is protected by a CRC-5, shown in Table 12 and detailed in Annex F.

Select, *Req_RN*, *Read*, *Write*, *Kill*, *Lock*, *Access*, *BlockWrite, BlockErase*, and *BlockPermalock* are protected by a CRC-16, defined in 6.4.1.5 and detailed in Annex F.

R=>T commands begin with either a preamble or a frame-sync, as described in 6.4.1.2.8. The command-code lengths specified in Table 19 do not include the preamble or frame-sync.

Tags shall ignore invalid commands. In general, "invalid" means a command that (1) is incorrect given the current Tag state, (2) is unsupported by the Tag, (3) has incorrect parameters, (4) has a CRC error, (5) specifies an incorrect session, (6) is inappropriately interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively, or (7) is in any other way not recognized or not executable by the Tag. The actual definition of "invalid" is state-specific and defined, for each Tag state, in Annex B and Annex C.

Table 19 — Commands

Command	Code	Length (bits)	Mandatory?	Protection
QueryRep	**0**0	4	Yes	Unique command length
ACK	**0**1	18	Yes	Unique command length
Query	**10**00	22	Yes	Unique command length and a CRC-5
QueryAdjust	**10**01	9	Yes	Unique command length
Select	**10**10	> 44	Yes	CRC-16
Reserved for future use	**10**11	–	–	–
NAK	**110**00000	8	Yes	Unique command length
Req_RN	**110**00001	40	Yes	CRC-16
Read	**110**00010	> 57	Yes	CRC-16
Write	**110**00011	> 58	Yes	CRC-16
Kill	**110**00100	59	Yes	CRC-16
Lock	**110**00101	60	Yes	CRC-16
Access	**110**00110	56	No	CRC-16
BlockWrite	**110**00111	> 57	No	CRC-16
BlockErase	**110**01000	> 57	No	CRC-16
BlockPermalock	**110**01001	> 66	No	CRC-16
Reserved for future use	**110**01010 … **110**11111	–	–	Some commands thereof are used for optional extension as defined in Clause 7.
Reserved for custom commands	**1110**0000 00000000 … **1110**0000 11111111	–	–	Manufacturer specified
Reserved for proprietary commands	**1110**0001 00000000 … **1110**0001 11111111	–	–	Manufacturer specified
Reserved for future use	**1110**0010 00000000 … **1110**1111 11111111	–	–	–

6.4.2.11.1 Select commands

The Select command set comprises a single command: *Select*.

6.4.2.11.1.1 *Select* (mandatory)

Select selects a particular Tag population based on user-defined criteria, enabling union (U), intersection (∩), and negation (~) based Tag partitioning. Interrogators perform U and ∩ operations by issuing successive *Select* commands. *Select* can assert or de-assert a Tag's **SL** flag, which applies across all four sessions, or it can set a Tag's **inventoried** flag to either *A* or *B* in any one of the four sessions.

Interrogators and Tags shall implement the *Select* command shown in Table 20. Target shall indicate whether the *Select* modifies a Tag's **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. Action shall elicit the Tag response shown in Table 21. The criteria for determining whether a Tag is matching or non-matching are specified in the MemBank, Pointer, Length and Mask fields. Truncate indicates whether a Tag's backscattered reply shall be truncated to include only those UII and StoredCRC bits that follow Mask. *Select* passes the following parameters from Interrogator to Tags:

Target indicates whether the *Select* command modifies a Tag's **SL** flag or its **inventoried** flag, and in the case of **inventoried** it further specifies one of four sessions. A *Select* command that modifies **SL** does not modify **inventoried**, and vice versa. Tags shall ignore *Select* commands whose Target is 101_2, 110_2, or 111_2.

Action indicates whether matching Tags assert or de-assert **SL**, or set their **inventoried** flag to *A* or to *B*. Tags conforming to the contents of the MemBank, Pointer, Length, and Mask fields are considered matching. Tags not conforming to the contents of these fields are considered non-matching.

MemBank specifies whether Mask applies to UII, TID, or User memory. *Select* commands shall apply to a single memory bank. Successive *Selects* may apply to different banks. MemBank shall not specify Reserved memory; if a Tag receives a *Select* specifying MemBank = 00_2 it shall ignore the *Select*. MemBank parameter value 00_2 is reserved for future use (RFU).

Pointer, Length, and Mask: Pointer and Length describe a memory range. Pointer references a memory bit address (Pointer is not restricted to word boundaries) and uses EBV formatting (see Annex A). Length is 8 bits, allowing Masks from 0 to 255 bits in length. Mask, which is Length bits long, contains a bit string that a Tag compares against the memory location that begins at Pointer and ends Length bits later. If Pointer and Length reference a memory location that does not exist on the Tag then the Tag shall consider the *Select* to be non-matching. If Length is zero then all Tags shall be considered matching, unless Pointer references a memory location that does not exist on the Tag or Truncate = 1 and Pointer is outside the UII specified in the StoredPC, in which case the Tag shall consider the *Select* to be non-matching.

Truncate: If an Interrogator asserts Truncate, and if a subsequent *Query* specifies Sel=10 or Sel=11, then a matching Tag shall truncate their reply to an *ACK* to that portion of the UII immediately following Mask, followed by the StoredCRC in UII memory 00_h to $0F_h$. If an Interrogator asserts Truncate, it shall assert it:

in the last *Select* that the Interrogator issues prior to sending a *Query*,

only if the *Select* has Target = 100_2, and

only if Mask ends in the UII.

These constraints *do not* preclude an Interrogator from issuing multiple *Select* commands that target the **SL** and/or **inventoried** flags. They *do* require that an Interrogator that is requesting Tags to truncate their replies assert Truncate in the last *Select*, and that this last *Select* targets the **SL** flag. Tags shall power-up with Truncate de-asserted.

Tags shall decide whether to truncate their backscattered UII on the basis of the most recently received *Select*. If a Tag receives a *Select* with Truncate=1 but Target<>100_2 the Tag shall ignore the *Select*. If a Tag receives a *Select* in which Truncate=1 but MemBank<>01, the Tag shall consider the *Select* to be invalid. If a Tag receives a *Select* in which Truncate=1, MemBank=01, but Mask ends outside the UII specified in the StoredPC, the Tag shall consider the *Select* to be not matching.

Mask may end at the last bit of the UII, in which case a selected Tag shall backscatter its StoredCRC.

Truncated replies never include PC word or the XPC words, because Mask must end in the UII.

A Tag shall preface its truncated reply with five leading zeros (00000_2) inserted between the preamble and the truncated reply. Specifically, when truncating its replies a Tag backscatters 00000_2, then the portion of its UII following Mask, and then its StoredCRC. See Table 14. A Tag does not recalculate its StoredCRC for a truncated reply.

A recommissioned Tag shall not truncate its replies. A recommissioned Tag that receives a Select with Truncate=1 shall evaluate the Select normally, but when replying to a subsequent *ACK* it shall backscatter its PacketPC, XPC_W1, optionally its XPC_W2 (if XEB is asserted), an UII whose length is as specified in the UII length field in the StoredPC, and its PacketCRC.

Interrogators can use a *Select* command to reset all Tags in a session to **inventoried** state *A*, by issuing a *Select* with Action = 000_2 and a Length value of zero.

Because a Tag stores its StoredPC and StoredCRC in UII memory, a Select command may select on them. Because a Tag computes its PacketPC and PacketCRC dynamically and does not store them in memory, a Select command is unable to select on them.

Because a Tag may compute its PC and/or CRC dynamically, its response to a Select command whose Pointer, Length, and Mask include the StoredPC or StoredCRC may produce unexpected behaviour. Specifically, a Tag's backscattered reply may appear to not match Mask even though the Tag's behaviour indicates matching, and vice versa. For example, suppose an Interrogator sends a Select to match a 00100_2 UII length field in the StoredPC. Further assume that a Tag matches, but has an asserted XI. The Tag will dynamically increment its UII length field to 00101_2 when responding to an *ACK*, and will backscatter this incremented value in the PacketPC. The Tag was matching, but the backscattered UII length field appears to be non-matching.

Interrogators shall prepend a *Select* with a frame-sync (see 6.4.1.2.8). The CRC-16 that protects a Select is calculated over the first command-code bit to the Truncate bit.

Tags shall not reply to a *Select*.

Table 20 — *Select* command

	Command	Target	Action	MemBank	Pointer	Length	Mask	Truncate	CRC-16
# of bits	4	3	3	2	EBV	8	Variable	1	16
description	1010	000: **Inventoried** (S0) 001: **Inventoried** (S1) 010: **Inventoried** (S2) 011: **Inventoried** (S3) 100: **SL** 101: RFU 110: RFU 111: RFU	See Table 21	00: RFU 01: UII 10: TID 11: User	Starting Mask address	Mask length (bits)	Mask value	0: Disable truncation 1: Enable truncation	

Table 21 — Tag response to Action parameter

Action	Matching	Non-matching
000	assert **SL** or **inventoried** → *A*	de-assert **SL** or **inventoried** → *B*
001	assert **SL** or **inventoried** → *A*	do nothing
010	do nothing	de-assert **SL** or **inventoried** → *B*
011	negate **SL** or (*A* → *B*, *B* → *A*)	do nothing
100	de-assert **SL** or **inventoried** → *B*	assert **SL** or **inventoried** → *A*
101	de-assert **SL** or **inventoried** → *B*	do nothing
110	do nothing	assert **SL** or **inventoried** → *A*
111	do nothing	negate **SL** or (*A* → *B*, *B* → *A*)

6.4.2.11.2 Inventory commands

The inventory command set comprises *Query*, *QueryAdjust, QueryRep, ACK*, and *NAK*.

6.4.2.11.2.1 Query (mandatory)

Interrogators and Tags shall implement the *Query* command shown in Table 22. *Query* initiates and specifies an inventory round. *Query* includes the following fields:

DR (TRcal divide ratio) sets the T=>R link frequency as described in 6.4.1.2.8 and Table 9.

M (cycles per symbol) sets the T=>R data rate and modulation format as shown in Table 10.

TRext chooses whether the T=>R preamble is pre-pended with a pilot tone as described in 6.4.1.3.2.2 and 6.4.1.3.2.4, however, a Tag's reply to access commands that write to memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite* and *BlockErase*, and *BlockPermalock*) always use a pilot tone regardless of the value of TRext.

Sel chooses which Tags respond to the *Query* (see 6.4.2.11.1.1 and 6.4.2.8).

Session chooses a session for the inventory round (see 6.4.2.8).

Target selects whether Tags whose **inventoried** flag is *A* or *B* participate in the inventory round. Tags may change their **inventoried** flag from *A* to *B* (or vice versa) as a result of being singulated.

Q sets the number of slots in the round (see 6.4.2.8).

Interrogators shall prepend a *Query* with a preamble (see 6.4.1.2.8).

The CRC-5 that protects a *Query* is calculated over the first command-code bit to the last Q bit. If a Tag receives a *Query* with a CRC-5 error it shall ignore the command.

Upon receiving a *Query*, Tags with matching Sel and Target shall pick a random value in the range (0, $2^{Q}-1$), inclusive, and shall load this value into their slot counter. If a Tag, in response to the *Query*, loads its slot counter with zero, then its reply to a *Query* shall be as shown in Table 23; otherwise the Tag shall remain silent.

A *Query* may initiate an inventory round in a new session, or in the prior session. If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter matches the prior session it shall invert its **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$) for the session before it evaluates whether to transition to **ready**, **arbitrate**, or **reply**. If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter does not match the prior session it shall leave its **inventoried** flag for the prior session unchanged when beginning the new round.

Tags shall support all DR and M values specified in Table 9 and Table 10, respectively.

Tags in any state other than **killed** shall execute a *Query* command, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure 19). Tags in the **killed** state shall ignore a *Query*.

Table 22 — *Query* command

	Command	DR	M	TRext	Sel	Session	Target	Q	CRC-5
# of bits	4	1	2	1	2	2	1	4	5
description	1000	0: DR=8 1: DR=64/3	00: M=1 01: M=2 10: M=4 11: M=8	0: No pilot tone 1: Use pilot tone	00: All 01: All 10: ~**SL** 11: **SL**	00: S0 01: S1 10: S2 11: S3	0: *A* 1: *B*	0–15	

Table 23 — Tag reply to a *Query* command

	Response
# of bits	16
description	RN16

6.4.2.11.2.2 *QueryAdjust* (mandatory)

Interrogators and Tags shall implement the *QueryAdjust* command shown in Table 24. *QueryAdjust* adjusts *Q* (i.e. the number of slots in an inventory round – see 6.4.2.8) without changing any other round parameters.

QueryAdjust includes the following fields:

Session corroborates the session number for the inventory round (see 6.4.2.8 and 6.4.2.11.2.1). If a Tag receives a *QueryAdjust* whose session number is different from the session number in the *Query* that initiated the round it shall ignore the command.

UpDn determines whether and how the Tag adjusts *Q*, as follows:

110: Increment *Q* (i.e. $Q = Q + 1$).

000: No change to *Q*.

011: Decrement *Q* (i.e. $Q = Q - 1$).

If a Tag receives a *QueryAdjust* with an UpDn value different from those specified above it shall ignore the command. If a Tag whose *Q* value is 15 receives a *QueryAdjust* with UpDn = 110 it shall change UpDn to 000 prior to executing the command; likewise, if a Tag whose *Q* value is 0 receives a *QueryAdjust* with UpDn = 011 it shall change UpDn to 000 prior to executing the command.

Tags shall maintain a running count of the current *Q* value. The initial *Q* value is specified in the *Query* command that started the inventory round; one or more subsequent *QueryAdjust* commands may modify *Q*.

A *QueryAdjust* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

Upon receiving a *QueryAdjust* Tags first update *Q*, then pick a random value in the range $(0, 2^Q - 1)$, inclusive, and load this value into their slot counter. If a Tag, in response to the *QueryAdjust*, loads its slot counter with zero, then its reply to a *QueryAdjust* shall be shown in Table 25; otherwise, the Tag shall remain silent. Tags shall respond to a *QueryAdjust* only if they received a prior *Query*.

Tags in any state except **ready** or **killed** shall execute a *QueryAdjust* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the Tag is not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively).

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryAdjust* whose session parameter matches the session parameter in the prior *Query*, and who are not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively), shall invert their **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$, as appropriate) for the current session and transition to **ready**.

Table 24 — *QueryAdjust* command

	Command	Session	UpDn
# of bits	4	2	3
description	1001	00: S0 01: S1 10: S2 11: S3	110: $Q = Q + 1$ 000: No change to *Q* 011: $Q = Q - 1$

Table 25 — Tag reply to a *QueryAdjust* command

	Response
# of bits	16
description	RN16

6.4.2.11.2.3 *QueryRep* (mandatory)

Interrogators and Tags shall implement the *QueryRep* command shown in Table 26. *QueryRep* instructs Tags to decrement their slot counters and, if slot = 0 after decrementing, to backscatter an RN16 to the Interrogator.

QueryRep includes the following field:

Session corroborates the session number for the inventory round (see 6.4.2.8 and 6.4.2.11.2.1). If a Tag receives a *QueryRep* whose session number is different from the session number in the *Query* that initiated the round it shall ignore the command.

A *QueryRep* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

If a Tag, in response to the *QueryRep*, decrements its slot counter and the decremented slot value is zero, then its reply to a *QueryRep* shall be as shown in Table 27; otherwise the Tag shall remain silent. Tags shall respond to a *QueryRep* only if they received a prior *Query*.

Tags in any state except **ready** or **killed** shall execute a *QueryRep* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the Tag is not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively).

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryRep* whose session parameter matches the session parameter in the prior *Query*, and who are not in the middle of a *Kill* or *Access* command sequence (see 6.4.2.11.3.4 or 6.4.2.11.3.6 respectively), shall invert their **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$, as appropriate) for the current session and transition to **ready**.

Table 26 — *QueryRep* command

	Command	Session
# of bits	2	2
description	00	00: S0 01: S1 10: S2 11: S3

Table 27 — Tag reply to a *QueryRep* command

	Response
# of bits	16
description	RN16

6.4.2.11.2.4 *ACK* (mandatory)

Interrogators and Tags shall implement the *ACK* command shown in Table 28. An Interrogator sends an *ACK* to acknowledge a single Tag. *ACK* echoes the Tag's backscattered RN16.

If an Interrogator issues an *ACK* to a Tag in the **reply** or **acknowledged** states, then the echoed RN16 shall be the RN16 that the Tag previously backscattered as it transitioned from the **arbitrate** state to the **reply** state. If an Interrogator issues an *ACK* to a Tag in the **open** or **secured** states, then the echoed RN16 shall be the Tag's handle (see 6.4.2.11.3.1).

An *ACK* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

The Tag reply to a successful *ACK* shall be as shown in Table 29. As described in 6.4.2.11.1.1, the reply may be truncated, in which case the Tag reply is 00000_2 followed by the truncated UII followed by the StoredCRC computed by the Tag at power-up. A Tag that receives an *ACK* with an incorrect RN16 or an incorrect handle

(as appropriate) shall return to **arbitrate** without responding, unless the Tag is in **ready** or **killed**, in which case it shall ignore the *ACK* and remain in its present state.

If a Tag does not support XPC functionality then the maximum length of its backscattered UII is 496 bits. If a Tag supports XPC functionality then the maximum length of its backscattered UII is reduced by two words to accommodate the XPC_W1 and optional XPC_W2 - see 6.4.2.1.2.5, so is 464 bits. If additionally Simple Sensor Data is transmitted subsequent to the UII, see 8.3 the maximum length of the UII decreases accordingly. In either case a Tag's reply to an *ACK* shall not exceed 528 bits in length.

Table 28 — *ACK* command

	Command	RN
# of bits	2	16
description	01	Echoed RN16 or handle

Table 29 — Tag reply to a successful *ACK* command – For exceptions in respect to sensors see Clause 7

	Response
# of bits	21 to 528
description	See Table 14

6.4.2.11.2.5 *NAK* (mandatory)

Interrogators and Tags shall implement the *NAK* command shown in Table 30. Any Tag that receives a *NAK* shall return to the **arbitrate** state without changing its **inventoried** flag, unless the Tag is in **ready** or **killed**, in which case it shall ignore the *NAK* and remain in its current state.

A *NAK* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

Tags shall not reply to a *NAK*.

Table 30 — *NAK* command

	Command
# of bits	8
description	11000000

6.4.2.11.3 Access commands

The set of access commands comprises *Req_RN, Read, Write*, *Kill*, *Lock*, *Access*, *BlockWrite*, *BlockErase* and *BlockPermalock*. As described in 6.4.2.9, Tags execute *Req_RN* from the **acknowledged**, **open**, or **secured** states. Tags execute *Read*, *Write*, *BlockWrite*, and *BlockErase* from the **secured** state; if allowed by the lock status of the memory location being accessed, they may also execute these commands from the **open** state. Tags execute *Access* and *Kill* from the **open** or **secured** states. Tags execute *Lock* and *BlockPermalock* only from the **secured** state.

All access commands issued to a Tag in the **open** or **secured** states include the Tag's handle as a parameter in the command. When in either of these two states, the Tag shall verify that the handle is correct prior to executing an access command, and the Tag shall ignore access commands with an incorrect handle. The handle value is fixed for the entire duration of a Tag access.

A Tag's reply to all access commands that read or write memory (i.e. *Read, Write*, *Kill*, *Lock*, *BlockWrite*, and *BlockErase* and *BlockPermalock*) includes a 1-bit header. Header=0 indicates that the operation was successful and the reply is valid; header=1 indicates that the operation was unsuccessful and the reply is an error code.

A Tag's reply to all access commands that write memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite*, *BlockErase*, and *BlockPermalock* with Read/Lock=1 (see 6.4.2.11.3.9) shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. the Tag shall reply as if TRext=1 regardless of the TRext value specified in the *Query* command that initiated the inventory round).

After an Interrogator writes to a Tag's StoredPC or UII, or changes bits $0B_h$ to $0F_h$ of User memory from zero to a nonzero value (or vice versa), or recommissions the Tag, the StoredCRC may be incorrect until the Interrogator powers-down and then re-powers-up its energizing RF field, because a Tag calculates its StoredCRC at powerup.

If an Interrogator attempts to write to UII memory locations 00_h to $0F_h$ (i.e. to the XPC_W1 or XPC_W2) using a *Write*, *BlockWrite*, or *BlockErase* command the Tag shall ignore the command and respond with an error code (see Annex I for error-code definitions and for the reply format).

If an Interrogator attempts to write to UII memory locations 210_h to $22F_h$ (i.e. to the XPC words) of a Passive Tag using a *Write*, *BlockWrite*, or *BlockErase* command the Tag shall ignore the command and respond with an error code (see Annex I for error-code definitions and for the reply format).

If an Interrogator attempts to write to a memory bank or password that is permalocked unwriteable, or to a memory bank or password that is locked unwriteable and the Tag is not in the **secured** state, or to a memory block that is permalocked, using a *Write*, *BlockWrite*, or *BlockErase* command, or if a portion of the Data in a *Write* command overlaps a permalocked memory block, the Tag shall ignore the command and respond with an error code (see Annex I for error-code definitions and for the reply format).

Req_RN, Read, Write, *Kill*, and *Lock* are required commands; *Access*, *BlockWrite*, *BlockErase*, and *BlockPermalock* are optional. A Tag shall ignore optional access commands that it does not support.

See Annex K for an example of a data-flow exchange during which an Interrogator accesses a Tag and reads its kill password.

6.4.2.11.3.1 *Req_RN* (mandatory)

Interrogators and Tags shall implement the *Req_RN* command shown in Table 31. *Req_RN* instructs a Tag to backscatter a new RN16. Both the Interrogator's command, and the Tag's response, depend on the Tag's state:

Acknowledged state: When issuing a *Req_RN* command to a Tag in the **acknowledged** state, an Interrogator shall include the Tag's last backscattered RN16 as a parameter in the *Req_RN*. The *Req_RN* command is protected by a CRC-16 calculated over the command bits and the RN16. If the Tag receives the *Req_RN* with a valid CRC-16 and a valid RN16 it shall generate and store a new RN16 (denoted handle), backscatter this handle, and transition to the **open** or **secured** state. The choice of ending state depends on the Tag's access password, as follows:

Access password <> 0: Tag transitions to **open** state.

Access password = 0: Tag transitions to **secured** state.

If the Tag receives the *Req_RN* command with a valid CRC-16 but an invalid RN16 it shall ignore the *Req_RN* and remain in the **acknowledged** state.

Open or **secured** states: When issuing a *Req_RN* command to a Tag in the **open** or **secured** states, an Interrogator shall include the Tag's handle as a parameter in the *Req_RN*. The *Req_RN* command is protected by a CRC-16 calculated over the command bits and the handle. If the Tag receives the *Req_RN* with a valid CRC-16 and a valid handle it shall generate and backscatter a new RN16. If the Tag receives the *Req_RN* with a valid CRC-16 but an invalid handle it shall ignore the *Req_RN*. In either case the Tag shall remain in its current state (**open** or **secured**, as appropriate).

If an Interrogator wishes to ensure that only a single Tag is in the **acknowledged** state it may issue a *Req_RN*, causing the Tag or Tags to each backscatter a handle and transition to the **open** or **secured** state (as appropriate). The Interrogator may then issue an *ACK* with handle as a parameter. Tags that receive an *ACK* with an invalid handle shall return to **arbitrate** (Note: If a Tag receives an *ACK* with an invalid handle it returns to **arbitrate**, whereas if it receives an access command with an invalid handle it ignores the command).

The first bit of the backscattered RN16 shall be denoted the MSB; the last bit shall be denoted the LSB.

A *Req_RN* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

The Tag reply to a *Req_RN* shall be as shown in Table 32. The RN16 or handle are protected by a CRC-16.

Table 31 — *Req_RN* command

	Command	RN	CRC-16
# of bits	8	16	16
description	11000001	Prior RN16 or handle	

Table 32 — Tag reply to a *Req_RN* command

	RN	CRC-16
# of bits	16	16
description	handle or new RN16	

6.4.2.11.3.2 *Read* (mandatory)

Interrogators and Tags shall implement the *Read* command shown in Table 34. *Read* allows an Interrogator to read part or all of a Tag's Reserved, UII, TID, or User memory. *Read* has the following fields:

MemBank specifies whether the *Read* accesses Reserved, UII, TID, or User memory. *Read* commands shall apply to a single memory bank. Successive *Reads* may apply to different banks.

WordPtr specifies the starting word address for the memory read, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).

WordCount specifies the number of 16-bit words to be read. If WordCount = 00_h the Tag shall backscatter the contents of the chosen memory bank starting at WordPtr and ending at the end of the bank, unless MemBank = 01, in which case the Tag shall backscatter the memory contents specified in Table 33.

Table 33 – Tag backscatter when WordCount=00_h and MemBank=01_2

WordPtr memory word address	Tag implements XPC_W1?	Tag implements XPC_W2?	What the Tag backscatters
Within the StoredCRC, StoredPC, or the UII specified by bits 10_h–14_h of the StoredPC	Don't care	Don't care	UII memory starting at WordPtr and ending at the UII length specified by bits 10_h–14_h of the StoredPC
Within physical UII memory but above the UII specified by bits 10_h–14_h of the StoredPC	No	N/A. See NOTE 1	UII memory starting at WordPtr and ending at the end of physical UII memory
Within physical UII memory but above the UII specified by bits 10_h–14_h of the StoredPC	Yes	No	UII memory starting at WordPtr and ending at the end of physical UII memory. Includes the XPC_W1 if WordPtr is less than or equal to 210_h and physical UII memory extends to or above address 210_h
Within physical UII memory but above the UII specified by bits 10_h–14_h of the StoredPC	Yes	Yes	UII memory starting at WordPtr and ending at the end of physical UII memory. Includes the XPC_W1 and XPC_W2 if WordPtr is less than or equal to 210_h and physical UII memory extends to or above address 210_h. Includes the XPC_W2 if WordPtr is equal to 220_h and physical UII memory extends to or above 220_h.
210_h. Above physical UII memory	No	N/A. See NOTE 1	Error code
210_h. Above physical UII memory	Yes	No	XPC_W1
210_h. Above physical UII memory	Yes	Yes	XPC_W1 and XPC_W2
220_h. Above physical UII memory	No	N/A. See NOTE 1	Error code
220_h. Above physical UII memory	Yes	No	Error code
220_h. Above physical UII memory	Yes	Yes	XPC_W2
Not 210_h or 220_h. Above physical UII memory.	Don't care	Don't care	Error code
NOTE 1 If a Tag doesn't implement an XPC_W1 then it doesn't implement an XPC_W2. See 6.4.2.1.2.5.			

The *Read* command also includes the Tag's handle and a CRC-16. The CRC-16 is calculated over the first command-code bit to the last handle bit.

If a Tag receives a *Read* with a valid CRC-16 but an invalid handle it shall ignore the *Read* and remain in its current state (**open** or **secured**, as appropriate).

A *Read* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

If all of the memory words specified in a *Read* exist and none are read-locked, the Tag reply to the *Read* shall be as shown in Table 35. The Tag responds by backscattering a header (a 0-bit), the requested memory words, and its handle. The reply includes a CRC-16 calculated over the 0-bit, memory words, and handle.

If one or more of the memory words specified in the *Read* command either do not exist or are read-locked, the Tag shall backscatter an error code, within time T_1 in Table 13, rather than the reply shown in Table 35 (see Annex I for error-code definitions and for the reply format).

Table 34 — *Read* command

	Command	MemBank	WordPtr	WordCount	RN	CRC-16
# of bits	8	2	EBV	8	16	16
description	11000010	00: Reserved 01: UII 10: TID 11: User	Starting word address pointer	Number of words to read	handle	

Table 35 — Tag reply to a successful *Read* command

	Header	Memory Words	RN	CRC-16
# of bits	1	Variable	16	16
description	0	Data	handle	

6.4.2.11.3.3 *Write* (mandatory)

Interrogators and Tags shall implement the *Write* command shown in Table 36. *Write* allows an Interrogator to write a word in a Tag's Reserved, UII, TID, or User memory. *Write* has the following fields:

MemBank specifies whether the *Write* occurs in Reserved, UII, TID, or User memory.

WordPtr specifies the word address for the memory write, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).

Data contains a 16-bit word to be written. Before each and every *Write* the Interrogator shall first issue a *Req_RN* command; the Tag responds by backscattering a new RN16. The Interrogator shall cover-code the data by EXORing it with this new RN16 prior to transmission.

The *Write* command also includes the Tag's handle and a CRC-16. The CRC-16 is calculated over the first command-code bit to the last handle bit.

If a Tag in the **open** or **secured** states receives a *Write* with a valid CRC-16 but an invalid handle, or it receives a *Write* before which the immediately preceding command was not a *Req_RN*, it shall ignore the *Write* and remain in its current state.

A *Write* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

After issuing a *Write* an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's *Write* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *Write*, depending on the success or failure of the Tag's memory-write operation:

The *Write* succeeds: After completing the *Write* a Tag shall backscatter the reply shown in Table 37 and Figure 22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *Write* completed successfully.

The Tag encounters an error: The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 37 (see Annex I for error-code definitions and for the reply format).

The *Write* does not succeed: If the Interrogator does not observe a reply within 20ms then the *Write* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *Write* command.

Upon receiving a valid *Write* command a Tag shall write the commanded Data into memory. The Tag's reply to a *Write* shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 36 — *Write* command

	Command	MemBank	WordPtr	Data	RN	CRC-16
# of bits	8	2	EBV	16	16	16
description	11000011	00: Reserved 01: UII 10: TID 11: User	Word address pointer	RN16 ⊗ word to be written	handle	

Table 37 — Tag reply to a successful *Write* command

	Header	RN	CRC-16
# of bits	1	16	16
description	0	handle	

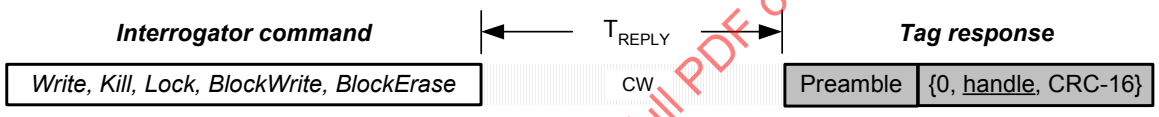


Figure 22 — Successful *Write* sequence

6.4.2.11.3.4 *Kill* (mandatory)

Interrogators and Tags shall implement the *Kill* command shown in Table 38 and Table 39. *Kill* allows an Interrogator to permanently disable a Tag. *Kill* also allows an Interrogator to recommission recommissionable Tags.

To kill or recommission a Tag, an Interrogator shall follow the multi-step kill procedure outlined in Figure 23. Briefly, an Interrogator issues two *Kill* commands, the first containing the 16 MSBs of the Tag's kill password EXORed with an RN16, and the second containing the 16 LSBs of the Tag's kill password EXORed with a different RN16. Each EXOR operation shall be performed MSB first (i.e. the MSB of each half-password shall be EXORed with the MSB of its respective RN16). Just prior to issuing each *Kill* command the Interrogator first issues a *Req_RN* to obtain a new RN16.

Tags shall incorporate the necessary logic to successively accept two 16-bit subportions of a 32-bit kill password. Interrogators shall not intersperse commands other than *Req_RN* between the two successive *Kill* commands. If a Tag, after receiving a first *Kill*, receives any valid command other than *Req_RN* before the second *Kill* it shall return to **arbitrate**, unless the intervening command is a *Query*, in which case the Tag shall execute the *Query* (inverting its **inventoried** flag if the session parameter in the *Query* matches the prior session).

Kill contains 3 RFU/Recom bits. In the first *Kill* command these bits are RFU. Interrogators shall set them to 000_2 when communicating with Passive Tags, and Passive Tags shall ignore them. Higher-functionality Tags may use these bits to expand the functionality of the *Kill* command. As described in 6.4.2.10, in the second *Kill* command these bits are called *recommissioning* (or Recom) bits and may be nonzero. The procedures for killing or recommissioning a Tag are identical, except that the recommissioning bits in the second *Kill* command are zero when killing a Tag and are nonzero when recommissioning it. Regardless of the intended operation, a Tag does not kill or recommission itself without first receiving the correct kill password by the procedure shown in Figure 23.

If a Tag does not implement recommissioning then it ignores the recommissioning bits and treats them as though they were zero, meaning that, if the Tag receives a properly formatted *Kill* command sequence with the correct kill password it kills itself dead regardless of the values of the recommissioning bits.

Tags whose kill password is zero do not execute a kill or a recommissioning operation; if such a Tag receives a *Kill* command it ignores the command and backscatters an error code (see Figure 23).

The Tag reply to the first *Kill* command shall be as shown in Table 39. The reply shall use the TRext value specified in the *Query* command that initiated the round.

After issuing the second *Kill* command an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's second *Kill* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *Kill* command sequence, depending on the success or failure of the Tag's kill or recommissioning operation:

- **The *Kill* or recommissioning succeeds:** After completing the operation the Tag shall backscatter the reply shown in Table 39 and Figure 22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. Immediately after this reply the Tag shall render itself silent and shall not respond to an Interrogator thereafter. If the Interrogator observes this reply within 20 ms then the operation completed successfully. If the Tag is killed dead then immediately after this reply the Tag shall render itself silent and shall not respond to an Interrogator thereafter.

- **The Tag encounters an error:** The **Tag** shall backscatter an error code during the CW period rather than the reply shown in Table 39 (Annex I for error-code definitions and for the reply format).

- **The *Kill* or recommissioning does not succeed:** If the Interrogator does not observe a reply within 20ms then the operation did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reinitiate the multi-step kill procedure outlined in Figure 23.

A *Kill* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

Upon receiving a valid *Kill* command sequence a Tag shall render itself killed or recommissioned, as appropriate. The Tag's reply to the second *Kill* command shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 38 — *Kill* command

	Command	Password	RFU/Recom	RN	CRC-16
# of bits	8	16	3	16	16
description	11000100	(½ kill password) ⊗ RN16	000_2	handle	

Table 39 — Second *Kill* command

	Command	Password	RFU/Recom	RN	CRC-16
# of bits	8	16	3	16	16
description	11000100	(½ kill password) ⊗ RN16	Recommissioning bits	handle	

Table 40 – Tag reply to the first *Kill* command

	RN	CRC-16
# of bits	16	16
description	handle	

Table 41 — Tag reply to a successful *Kill* procedure

	Header	RN	CRC-16
# of bits	1	16	16
description	0	handle	

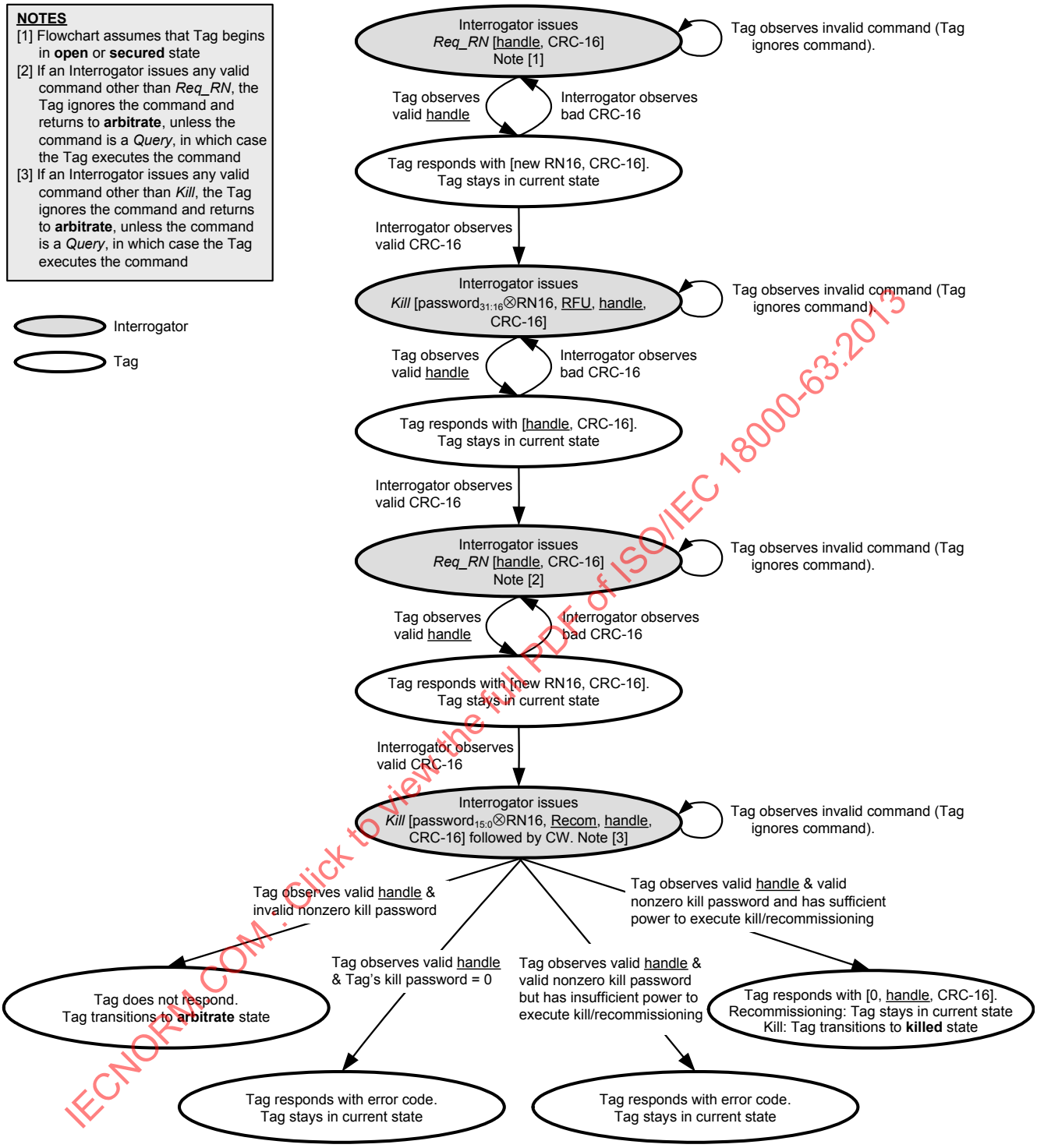


Figure 23 — *Kill* procedure

6.4.2.11.3.5 *Lock* (mandatory)

Interrogators and Tags shall implement the *Lock* command shown in Table 42 and Figure 24. Only Tags in the **secured** state shall execute a *Lock* command. *Lock* allows an Interrogator to:

- Lock individual passwords, thereby preventing or allowing subsequent reads and writes of that password,

- Lock individual memory banks, thereby preventing or allowing subsequent writes to that bank, and
- Permalock (make permanently unchangeable) the lock status for a password or memory bank.

Lock contains a 20-bit payload defined as follows:

The first 10 payload bits are Mask bits. A Tag shall interpret these bit values as follows:

Mask = 0: Ignore the associated Action field and retain the current lock setting.

Mask = 1: Implement the associated Action field and overwrite the current lock setting.

The last 10 payload bits are Action bits. A Tag shall interpret these bit values as follows:

Action = 0: De-assert lock for the associated memory location.

Action = 1: Assert lock or permalock for the associated memory location.

The functionality of the various Action fields is described in Table 44.

The payload of a *Lock* command shall always be 20 bits in length.

If an Interrogator issues a *Lock* command whose Mask and Action fields attempt to change the lock status of a nonexistent memory bank or nonexistent password, the Tag shall ignore the entire *Lock* command and instead backscatter an error code (see Annex I).

The *Lock* command differs from the optional *BlockPermalock* command in that *Lock* reversibly or permanently locks a password or an entire UII, TID, or User memory bank in a writeable or unwriteable state, whereas *BlockPermalock* permanently locks blocks of User memory in an unwriteable state. Table 51 specifies how a Tag shall react to a *Lock* command that follows a prior *BlockPermalock* command, or vice versa.

Permalock bits, once asserted, cannot be de-asserted except by recommissioning the Tag (see 6.4.2.10). If a Tag receives a *Lock* whose payload attempts to de-assert a previously asserted permalock bit, the Tag shall ignore the *Lock* and backscatter an error code (see Annex I). If a Tag receives a *Lock* whose payload attempts to reassert a previously asserted permalock bit, the Tag shall simply ignore this particular Action field and implement the remainder of the *Lock* payload.

A Tag's lock bits cannot be read directly; they can be inferred by attempting to perform other memory operations.

All Tags shall implement memory locking, and all Tags shall implement the *Lock* command. However, Tags need not support all the Action fields shown in Figure 24, depending on whether the password location or memory bank associated with an Action field exists and is lockable and/or unlockable. Specifically, if a Tag receives a *Lock* it cannot execute because one or more of the passwords or memory banks do not exist, or one or more of the Action fields attempt to change a permalocked value, or one or more of the passwords or memory banks are either not lockable or not unlockable, the Tag shall ignore the entire *Lock* and instead backscatter an error code (see Annex I). The only exception to this general rule relates to Tags whose only lock functionality is to permanently lock **all** memory (i.e. all memory banks and all passwords) at once; these Tags shall execute a *Lock* whose payload is $FFFFF_h$, and shall backscatter an error code for any payload other than $FFFFF_h$.

A *Lock* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

After issuing a *Lock* an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's *Lock* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *Lock*, depending on the success or failure of the Tag's memory-write operation:

The *Lock* succeeds: After completing the *Lock* the Tag shall backscatter the reply shown in Table 43 and Figure 22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *Lock* completed successfully.

The Tag encounters an error: The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 43 (see Annex I for error-code definitions and for the reply format).

The *Lock* does not succeed: If the Interrogator does not observe a reply within 20ms then the *Lock* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *Lock*.

Upon receiving a valid *Lock* command a Tag shall perform the commanded lock operation. The Tag's reply to a *Lock* shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 42 — Lock command

	Command	Payload	RN	CRC-16
# of bits	8	20	16	16
description	11000101	Mask and Action Fields	handle	

Table 43 — Tag reply to a Lock command

	Header	RN	CRC-16
# of bits	1	16	16
description	0	handle	

Lock-Command Payload

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Kill Mask		Access Mask		UII Mask		TID Mask		User Mask		Kill Action		Access Action		UII Action		TID Action		User Action	

Masks and Associated Action Fields

	Kill pwd		Access pwd		UII memory		TID memory		User memory	
	19	18	17	16	15	14	13	12	11	10
Mask	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write	skip/ write
	9	8	7	6	5	4	3	2	1	0
Action	pwd read/ write	perma lock	pwd read/ write	perma lock	pwd write	perma lock	pwd write	perma lock	pwd write	perma lock

Figure 24 — Lock payload and usage

Table 44 — Lock Action-field functionality

pwd-write	permalock	Description
0	0	Associated memory bank is writeable from either the **open** or **secured** states.
0	1	Associated memory bank is permanently writeable from either the **open** or **secured** states and may never be locked.
1	0	Associated memory bank is writeable from the **secured** state but not from the **open** state.
1	1	Associated memory bank is not writeable from any state.
pwd-read/write	**permalock**	**Description**
0	0	Associated password location is readable and writeable from either the **open** or **secured** states.
0	1	Associated password location is permanently readable and writeable from either the **open** or **secured** states and may never be locked.
1	0	Associated password location is readable and writeable from the **secured** state but not from the **open** state.
1	1	Associated password location is not readable or writeable from any state.

6.4.2.11.3.6 *Access* (optional)

Interrogators and Tags may implement an *Access* command; if they do, the command shall be as shown in Table 45. *Access* causes a Tag with a nonzero-valued access password to transition from the **open** to the **secured** state (a Tag with a zero-valued access password is never in the **open** state — see Figure 19) or, if the Tag is already in the **secured** state, to remain in **secured**.

To access a Tag, an Interrogator shall follow the multi-step procedure outlined in Figure 25. Briefly, an Interrogator issues two *Access* commands, the first containing the 16 MSBs of the Tag's access password EXORed with an RN16, and the second containing the 16 LSBs of the Tag's access password EXORed with a different RN16. Each EXOR operation shall be performed MSB first (i.e. the MSB of each half-password shall be EXORed with the MSB of its respective RN16). Just prior to issuing each *Access* command the Interrogator first issues a *Req_RN* to obtain a new RN16.

Tags shall incorporate the necessary logic to successively accept two 16-bit subportions of a 32-bit access password. Interrogators shall not intersperse commands other than *Req_RN* between the two successive *Access* commands. If a Tag, after receiving a first *Access*, receives any valid command other than *Req_RN* before the second *Access* it shall return to **arbitrate**, unless the intervening command is a *Query*, in which case the Tag shall execute the *Query* (inverting its **inventoried** flag if the session parameter in the *Query* matches the prior session).

An *Access* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

The Tag reply to an *Access* command shall be as shown in Table 46. If the *Access* is the first in the sequence, then the Tag backscatters its handle to acknowledge that it received the command. If the *Access* is the second in the sequence and the entire received 32-bit access password is correct, then the Tag backscatters its handle to acknowledge that it has executed the command successfully and has transitioned to the **secured** state; otherwise the Tag does not reply and returns to **arbitrate**. The reply includes a CRC-16 calculated over the handle.

Table 45 — *Access* command

	Command	Password	RN	CRC-16
# of bits	8	16	16	16
description	11000110	(½ access password) ⊗ RN16	handle	

Table 46 — Tag reply to an *Access* command

	RN	CRC-16
# of bits	16	16
description	handle	

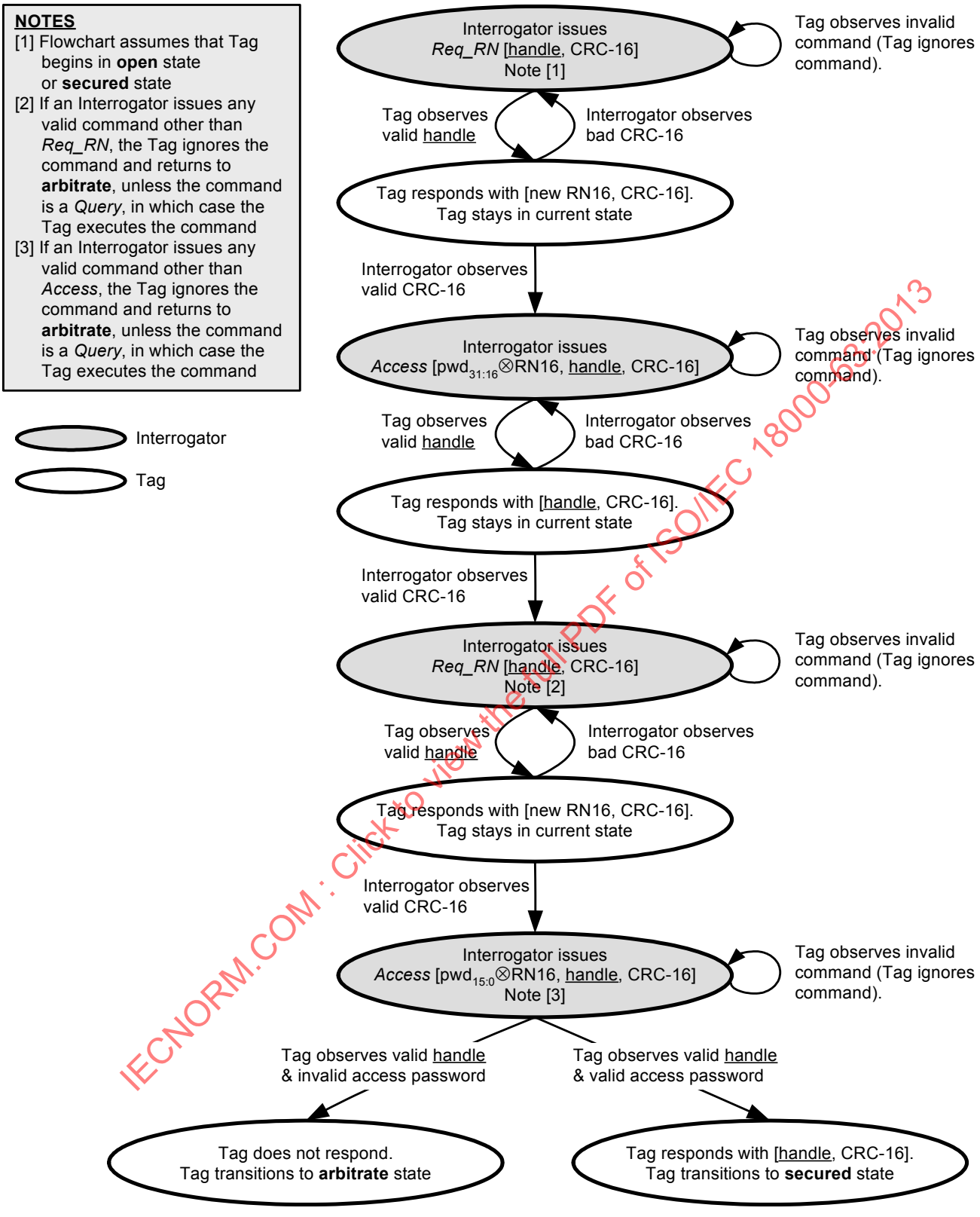


Figure 25 — *Access* procedure

6.4.2.11.3.7 *BlockWrite* (optional)

Interrogators and Tags may implement a *BlockWrite* command; if they do, they shall implement it as shown in Table 47. *BlockWrite* allows an Interrogator to write multiple words in a Tag's Reserved, UII, TID, or User memory using a single command. *BlockWrite* has the following fields:

MemBank specifies whether the *BlockWrite* occurs in Reserved, UII, TID, or User memory. *BlockWrite* commands shall apply to a single memory bank. Successive *BlockWrites* may apply to different banks.

WordPtr specifies the starting word address for the memory write, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).

WordCount specifies the number of 16-bit words to be written. If WordCount = 00_h the Tag shall ignore the *BlockWrite*. If WordCount = 01_h the Tag shall write a single data word.

Data contains the 16-bit words to be written, and shall be 16×WordCount bits in length. Unlike a *Write*, the data in a *BlockWrite* are not cover-coded, and an Interrogator need not issue a *Req_RN* before issuing a *BlockWrite*.

The *BlockWrite* command also includes the Tag's handle and a CRC-16. The CRC-16 is calculated over the first command-code bit to the last handle bit.

If a Tag receives a *BlockWrite* with a valid CRC-16 but an invalid handle it shall ignore the *BlockWrite* and remain in its current state (**open** or **secured**, as appropriate).

A *BlockWrite* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

After issuing a *BlockWrite* an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's *BlockWrite* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *BlockWrite*, depending on the success or failure of the Tag's memory-write operation:

The *BlockWrite* succeeds: After completing the *BlockWrite* a Tag shall backscatter the reply shown in Table 48 and Figure 22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockWrite* completed successfully.

The Tag encounters an error: The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 48 (see Annex I for error-code definitions and for the reply format).

The *BlockWrite* does not succeed: If the Interrogator does not observe a reply within 20ms then the *BlockWrite* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *BlockWrite*.

Upon receiving a valid *BlockWrite* command a Tag shall write the commanded Data into memory. The Tag's reply to a *BlockWrite* shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 47 — *BlockWrite* command

	Command	MemBank	WordPtr	WordCount	Data	RN	CRC-16
# of bits	8	2	EBV	8	Variable	16	16
description	11000111	00: Reserved 01: UII 10: TID 11: User	Starting word address pointer	Number of words to write	Data to be written	handle	

Table 48 — Tag reply to a successful *BlockWrite* command

	Header	RN	CRC-16
# of bits	1	16	16
description	0	handle	

6.4.2.11.3.8 *BlockErase* (optional)

Interrogators and Tags may implement a *BlockErase* command; if they do, they shall implement it as shown in Table 49. *BlockErase* allows an Interrogator to erase multiple words in a Tag's Reserved, UII, TID, or User memory using a single command. *BlockErase* has the following fields:

MemBank specifies whether the *BlockErase* occurs in Reserved, UII, TID, or User memory. *BlockErase* commands shall apply to a single memory bank. Successive *BlockErases* may apply to different banks.

WordPtr specifies the starting word address for the memory erase, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).

WordCount specifies the number of 16-bit words to be erased. If WordCount = 00_h the Tag shall ignore the *BlockErase*. If WordCount = 01_h the Tag shall erase a single data word.

The *BlockErase* command also includes the Tag's handle and a CRC-16. The CRC-16 is calculated over the first command-code bit to the last handle bit.

If a Tag receives a *BlockErase* with a valid CRC-16 but an invalid handle it shall ignore the *BlockErase* and remain in its current state (**open** or **secured**, as appropriate).

A *BlockErase* shall be pre-pended with a frame-sync (see 6.4.1.2.8).

After issuing a *BlockErase* an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's *BlockErase* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *BlockErase*, depending on the success or failure of the Tag's memory-erase operation:

The *BlockErase* succeeds: After completing the *BlockErase* a Tag shall backscatter the reply shown in Table 50 and Figure 22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockErase* completed successfully.

The Tag encounters an error: The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 50 (see Annex I for error-code definitions and for the reply format).

The *BlockErase* does not succeed: If the Interrogator does not observe a reply within 20ms then the *BlockErase* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *BlockErase*.

Upon receiving a valid *BlockErase* command a Tag shall erase the commanded memory words. The Tag's reply to a *BlockErase* shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 49 — *BlockErase* command

	Command	MemBank	WordPtr	WordCount	RN	CRC-16
# of bits	8	2	EBV	8	16	16
description	11001000	00: Reserved 01: UII 10: TID 11: User	Starting word address pointer	Number of words to erase	handle	

Table 50 — Tag reply to a successful *BlockErase* command

	Header	RN	CRC-16
# of bits	1	16	16
description	0	Handle	

6.4.2.11.3.9 *BlockPermalock (optional)*

Interrogators and Tags may implement a *BlockPermalock* command; if they do, they shall implement it as shown in Table 52. *BlockPermalock* allows an Interrogator to:

- Permalock one or more blocks (individual sub-portions) in a Tag's User memory, or
- Read the permalock status of the memory blocks in a Tag's User memory.

A single *BlockPermalock* command can permalock between 0 and 4080 blocks of User memory. The block size is vendor-defined. The memory blocks need not be contiguous.

Only Tags in the **secured** state shall execute a *BlockPermalock* command.

The *BlockPermalock* command differs from the *Lock* command in that *BlockPermalock* permanently locks blocks of User memory in an unwriteable state, whereas *Lock* reversibly or permanently locks a password or an entire memory bank in a writeable or unwriteable state. Table 51 specifies how a Tag shall react to a *BlockPermalock* command (with Read/Lock = 1) that follows a prior *Lock* command, or vice versa.

Table 51 — Precedence for *Lock* and *BlockPermalock* commands

First command			Second command			Tag Action and response to 2nd command
Lock	**pwd-write**	**permalock**	*BlockPermalock* (Read/Lock = 1)			
	0	0				Permalock the blocks indicated by Mask; respond as described in this section 6.4.2.11.3.9
	0	1				Reject the *BlockPermalock*; respond with an error code, see also NOTE.
	1	0				Permalock the blocks indicated by Mask; respond as described in this section 6.4.2.11.3.9
	1	1				Permalock the blocks indicated by Mask; respond as described in this section 6.4.2.11.3.9
BlockPermalock (Read/Lock = 1)			*Lock*	**pwd-write**	**permalock**	
				0	0	Implement the *Lock*, but do not un-permalock any blocks that were previously permalocked; respond as described in 6.4.2.11.3.5
				0	1	Reject the *Lock*; respond with an error code
				1	0	Implement the *Lock*, but do not un-permalock any blocks that were previously permalocked; respond as described in 6.4.2.11.3.5
				1	1	Implement the *Lock*; respond as described in 6.4.2.11.3.5
NOTE: Under these conditions the BlockPermalock shall only be rejected if it is attempting to lock within the just unlocked memory bank, i.e. in case Lock and BlockPermalock refer to the same memory bank, otherwise the command shall be obeyed.						

The *BlockPermalock* command has the following fields:

- MemBank specifies whether the *BlockPermalock* applies to UII, TID, or User memory. *BlockPermalock* commands shall apply to a single memory bank. Successive *BlockPermalocks* may apply to different memory banks. Passive Tags shall only execute a *BlockPermalock* command if MemBank = 11 (User memory); if a Passive Tag receives a *BlockPermalock* with MemBank<>11 it shall ignore the command and instead backscatter an error code (see Annex I), remaining in the **secured** state. Higher-functionality Tags may use the other MemBank values to expand the functionality of the *BlockPermalock* command.

- Read/Lock specifies whether a Tag backscatters the permalock status of, or permalocks, one or more blocks within the memory bank specified by MemBank. A Tag shall interpret the Read/Lock bit as follows:
  - Read/Lock = 0: A Tag shall backscatter the permalock status of blocks in the specified memory bank, starting from the memory block located at BlockPtr and ending at the memory block located at BlockPtr+(16×BlockRange)–1. A Tag shall backscatter a "0" if the memory block corresponding to that bit is not permalocked and a "1" if the block is permalocked. An Interrogator omits Mask from the *BlockPermalock* command when Read/Lock = 0.
  - Read/Lock = 1: A Tag shall permalock those blocks in the specified memory bank that are specified by Mask, starting at BlockPtr and ending at BlockPtr+(16×BlockRange)–1.

- BlockPtr specifies the starting block address for Mask, in units of 16 blocks. For example, BlockPtr = 00_h indicates block 0, BlockPtr = 01_h indicates block 16, BlockPtr = 02_h indicates block 32. BlockPtr uses EBV formatting (see Annex A).

- BlockRange specifies the range of Mask, starting at BlockPtr and ending (16×BlockRange)–1 blocks later. If BlockRange == 00_h then a Tag shall ignore the *BlockPermalock* command and instead backscatter an error code (see Annex I), remaining in the **secured** state.

- Mask specifies which memory blocks a Tag permalocks. Mask depends on the Read/Lock bit as follows:
  - Read/Lock = 0: The Interrogator shall omit Mask from the *BlockPermalock* command.
  - Read/Lock = 1: The Interrogator shall include a Mask of length 16×BlockRange bits in the *BlockPermalock* command. The Mask bits shall be ordered from lower-order block to higher (i.e. if BlockPtr == 00_h then the leading Mask bit refers to block 0). The Tag shall interpret each bit of Mask as follows:
    - Mask bit = 0: Retain the current permalock setting for the corresponding memory block.
    - Mask bit = 1: Permalock the corresponding memory block. If a block is already permalocked then the Tag shall retain the current permalock setting. A memory block, once permalocked, cannot be un-permalocked except by recommissioning the Tag (see 6.4.2.10).

The following examples illustrate the usage of Read/Lock, BlockPtr, BlockRange, and Mask:

- If Read/Lock=1, BlockPtr=01_h, and BlockRange=01_h the Tag operates on sixteen blocks starting at block 16 and ending at block 31, permalocking those blocks whose corresponding bits are asserted in Mask.

The *BlockPermalock* command contains 8 RFU bits. Interrogators shall set these bits to 00h when communicating with Passive Tags. If a Tag receives a BlockPermalock command containing nonzero RFU bits it shall ignore the command and instead backscatter an error code (see Annex I), remaining in the **secured** state. Higher-functionality Tags may use these bits to expand the functionality of the *BlockPermalock* command.

The *BlockPermalock* command also includes the Tag's handle and a CRC-16. The CRC-16 is calculated over the first command-code bit to the last handle bit. If a Tag receives a *BlockPermalock* with a valid CRC-16 but an invalid handle it shall ignore the *BlockPermalock* and remain in the **secured** state.

If a Tag receives a *BlockPermalock* command that it cannot execute because User memory does not exist, or in which the LSB and/or the 2 LSB of a Tag's XPC_W1 is/are asserted (see Table 15), or in which one of the asserted Mask bits references a non-existent block, then the Tag shall ignore the *BlockPermalock* command and instead backscatter an error code (see Annex I), remaining in the **secured** state. A Tag shall treat as invalid a *BlockPermalock* command in which Read/Lock=1 but Mask has a length that is not equal to 16xBlockRange bits (see 6.4.2.11 for the definition of invalid commands).

Certain Tags, depending on the Tag manufacturer's implementation, may be unable to execute a *BlockPermalock* command with certain BlockPtr and BlockRange values, in which case the Tag shall ignore the *BlockPermalock* command and instead backscatter an error code (see Annex I), remaining in the **secured** state. Because a Tag contains information in its TID memory that an Interrogator can use to uniquely identify the optional features that the Tag supports (see 6.4.2.1.3 and 7.6.3), this specification recommends that Interrogators read a Tag's TID memory prior to issuing a *BlockPermalock* command.

If an Interrogator issues a *BlockPermalock* command in which BlockPtr and BlockRange specify one or more nonexistent blocks, but Mask only asserts permalocking on existent blocks, then the Tag shall execute the command.

A *BlockPermalock* shall be prepended with a frame-sync (see 6.4.1.2.8).

After issuing a *BlockPermalock* command an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20ms, where T_{REPLY} is the time between the Interrogator's *BlockPermalock* and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from a *BlockPermalock* command, depending on the value of the Read/Lock bit in the command and, if Read/Lock = 1, the success or failure of the Tag's memory-lock operation:

- **Read/Lock = 0 and the Tag is able to execute the command:** The Tag shall backscatter the reply shown in Table 53, within time T_1 in Table 13, comprising a header (a 0-bit), the requested permalock bits, the Tag's handle, and a CRC-16 calculated over the 0-bit, permalock bits, and handle. The Tag's reply shall use the preamble specified by the TRext value in the *Query* that initiated the round.

- **Read/Lock = 0 and the Tag is unable to execute the command:** the Tag shall backscatter an error code, within time T_1 in Table 13, rather than the reply shown in Table 53 (see Annex I for error-code definitions and for the reply format). The Tag's reply shall use the preamble specified by the TRext value in the *Query* that initiated the round. An example of an unexecutable command is a Passive Tag receiving a *BlockPermalock* with MemBank<>11.

- **Read/Lock = 1 and The *BlockPermalock* succeeds:** After completing the *BlockPermalock* the Tag shall backscatter the reply shown in Table 54 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockPermalock* completed successfully. The Tag's reply shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. the Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

- **Read/Lock = 1 and the Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 54 (see Annex I for error-code definitions and for the reply format). The Tag's reply shall use the extended preamble shown in Figure 11 or Figure 15, as appropriate (i.e. the Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

- **Read/Lock = 1 and the *BlockPermalock* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *BlockPermalock* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *BlockPermalock*.

Table 52 — *BlockPermalock* command

	Command	RFU	Read/Lock	MemBank	BlockPtr	BlockRange	Mask	RN	CRC-16
# of bits	8	8	1	2	EBV	8	Variable	16	16
description	11001001	00_h	0: Read 1: Permalock	00: RFU 01: UII 10: TID 11: User	Mask starting block address, specified in units of 16 blocks	Mask range, specified in units of 16 blocks	0: Retain current permalock setting 1: Assert permalock	handle	

Table 53 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 0

	Header	Data	RN	CRC-16
# of bits	1	Variable	16	16
description	0	Permalock bits	handle	

Table 54 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 1

	Header	RN	CRC-16
# of bits	1	16	16
description	0	handle	

Upon receiving a valid *BlockPermalock* command a Tag shall perform the commanded operation, unless the Tag does not support block permalocking, in which case it shall ignore the command.

7 Battery Assisted Passive (BAP) Interrogator Talks First Type C systems (optional)

7.1 Applicability

In case an Interrogator or Tag supports any command, response or feature of Clause 7 then this Interrogator or Tag shall support all mandatory commands, responses or features, and it may support the specified combinations of optional commands, responses or features. If an Interrogator or Tag does not support any command, response, or feature of Clause 7, then Clause 7 does not apply for that device.

There are no new mandatory commands in Clause 7 simply to have a battery assisted Tag, but there are required modifications of flag persistence with batteries, and required relationships between optional commands and features.

7.2 General overview, definitions, and requirements of BAP

The benefit of BAP is to improve the robustness and performance of the air interface of a passive backscatter system. Specific benefits include better Tag sensitivity and resistance to multipath fade. These improvements can result in improved operating range or better penetration of reading zone into closely packed objects such as a pallet of goods. BAP also provides power for sensors which may then operate out of the field of an Interrogator. See Annex P for an informative tutorial overview of BAP as implemented in this standard.

This clause defines the requirements and features of Battery Assisted Passive Type C systems. It covers these types of BAP Tags:

BAP PIE: The air interface of Clause 6 supplemented with an on-Tag battery and optionally supplemented with battery life extending Battery Saver functionality referred to as Battery Saver Mode. This BAP PIE mode may be provided as the only mode a Tag supports. Battery Saver Mode is basically a duty cycled mode of manufacturer selected form where the Tag receiver is either OFF or in a very low power state, or switches between both, for the majority of the time. The Battery Saver Mode option may be implemented with a low power RF threshold detection function or alternatively with a successful command decode function as the trigger to activate the Tag and temporarily hold it in a fully operational state. Battery Saver Mode may feature either a low power continuous "**listen**" capability, or a duty cycled **listen** capability (typically at improved sensitivity compared to continuous), or both. A Tag that supports BAP PIE shall support being inventoried in parallel with passive PIE Tags without special actions required on the part of the Interrogator. It may also support being inventoried in special battery only Query rounds by use of the optional *Flex_Query* command. The *Flex_Query* command combines the function of a *Select* command and *Query* command, which allows taking targeted types of Tags into Query rounds with a single command.

For BAP PIE Tags the beginning of passive persistence **inventoried** flag and state machine state timeout are controlled by an on-Tag timer function of some type, and not dictated by loss of signal as occurs with passive Tags. This timer function allows the BAP PIE Tag to survive brief signal fades without unnecessarily losing state machine state or **inventoried** flag state, in particular on the S0 flag with its passive persistence definition of "None".

Manchester: The Manchester mode features an improved forward link using Manchester modulation that is capable of taking the performance of BAP Tags to the limits of the physically possible. A Manchester Tag shall also support PIE, though this support requirement is discontinuous as described below in this clause. The Manchester forward mode is orthogonal to the PIE mode in the sense that Tags of each mode do not accidentally decode commands of the other mode, thus providing some degree of interference resistance. The Manchester mode also features a **hibernate** state where a low data rate *Activation* command may bring the Tag up to a fully operational mode.

Interrogator BAP support requirements: Type C Interrogators have no requirements to support any commands or features as described in this clause. Their base requirements are as given in Clause 6. However, if an Interrogator that is specified by its manufacturer as explicitly supporting BAP PIE Tags, then it should also support the *Flex_Query* command of 7.4.1 (see Annex P for additional recommendations).

If an Interrogator is specified by its manufacturer as explicitly supporting Manchester, then it shall also support the physical modulation form and mandatory commands of 7.5 and its sub-clauses.

BAP Type C Tag PIE support requirements: Type C Interrogator-Talks-First Battery Assisted Passive Tags are required to support PIE. The Tag may satisfy this requirement with either the continuously available PIE of Clause 6, or with the battery supported and optionally duty cycled "Battery Assisted Passive PIE" or BAP PIE of clause 7.4. Though it is not required for BAP PIE to support better sensitivity than passive PIE, it is typical for the battery to allow improved sensitivity. If BAP PIE is duty cycled, the duty cycle maximum **sleep** time shall meet the requirements of 7.4.

The requirement for PIE support is temporarily suspended for Manchester Tags that have been activated in Manchester mode, for programmable Tags that are programmed differently by the user, and for Tags that are authorized to self-adjust their duty cycle in case of a depleted battery state. Also, a Tag that has a user controlled ON-OFF switch or other power-down control method outside of the standard air interface may when powered down be completely unresponsive.

BAP Tags with depleted batteries may be unresponsive, or may function in a purely passive PIE mode as described in Clause 6 where they are completely powered by the Interrogator RF field. Whether or not a BAP Tag will reply with a depleted battery is referred to as the Dead Battery Response (DBR) capability.

BAP Tags have the option to recalculate their storedCRC (see 6.4.2.1.2.1) upon entering the **battery ready** state. Alternatively, they may use a manufacturer specific implementation that meets the specifically BAP Tag modification to 6.4.2.1.2.1 to have the correct storedCRC at the time that the Tag may receive a *Query* command that brings it into a query round. For example, in a BAP Tag with continuous power a flag in the Tag

may tell its controller that this update has been made and there is no current need to recalculate the storedCRC.

BAP Tags may but are not required to support Extended Protocol Control (XPC). Tags that support XPC have the further option of supporting advanced capabilities and setting controls through a memory managed system referred to as the Tag Capabilities Reporting and Settings (TCRS). The TCRS system allows the Tag to report its full feature set, and for Interrogator control of some features such as the duty cycles it uses to listen for Interrogator signals. Tags may periodically check their settings file for Interrogator updates, such as upon preparing to enter the **battery ready** state (see sub-clause 7.3.1).

BAP Tags may but are not required to support the optional Access commands of *Access* (clause 6.4.2.11.3.6), *BlockWrite* (clause 6.4.2.11.3.7), *BlockErase* (clause 6.4.2.11.3.8), and *BlockPermalock* (clause 6.4.2.11.3.9).

BAP Tag physics and degree of power consumption allows for Tag sensitivity that ranges from approximately -10 dBm to -60 dBm without an RF low noise amplifier, thus leading to wide range of performance and interference control requirements (see Annex P and Annex Q for more detail). This performance may vary with battery state or particular Tag programming.

The term “Low Power Receive” or “LPR” is defined to mean a Tag that features a BAP PIE receive mode that is lower power and sensitivity than an additional receive mode that Tag also possesses. For example, when in LPR mode a Tag might not make use of an optional on-Tag amplifier that provides better sensitivity but consumes additional power when it is on. LPR is not the same as Dead Battery Response, as DBR implies that the Tag fully operates from energy harvested from the RF field. LPR only means that the BAP Tag has an ultra-low power mode in addition to a higher power consumption mode that may be valuable in operational planning. If the Tag supports TCRS, then the capability to support LPR shall be indicated in TCRS.

7.3 Battery Assisted Passive inventoried flag and state machine behaviour modifications

7.3.1 Modification to ready state and power-down support for BAP Tags

All Interrogator-Talks-First Type C BAP Tags shall support a variation of the passive **ready** state referred to as the **“battery ready”** state. The **battery ready** state is logically identical to the passive **ready** state with these two exceptions:

1. The **battery ready** state is battery supported and does not require the presence of an RF field.

2. In the case of BAP PIE Tags that do not support Battery Saver Mode, the **battery ready** state supports **inventoried** and **selected** flag persistence timer operation (see sub-clause 7.4.2) instead of these operations being performed in the implied deenergized state as is done for passive Tags.

Manchester Tags conduct their own form of persistence timing in the “**stateful hibernate**” state (see Figure 34). This is not the “inventory state holding” persistence of clause 6.4.2.2 and Table 17. Instead it is a persistence to reject new activations aimed at Tags that have not been recently inventoried. If the Tag was not successfully inventoried before an INACT_T or Global Timeout took the Tag back to the **hibernate** state, then the Tag shall be receptive to activations aimed at recently uninventoried Tags.

In BAP PIE Tags that support a Battery Saver Mode the persistence timing operations occur in similar “stateful” variations of the other states in the “**sleep**-**listen** mode”.

The Tag types that support various “low power modes”, meaning BAP PIE with Battery Saver Mode and Manchester with its support of hibernation, are exempted from the timing requirements of sub-clause 6.4.1.3.4 when in their associated low power mode. Instead they have more relaxed but still constrained timing limits, such as are described in sub-clause 7.4.2 for BAP PIE. BAP PIE Tags that do not implement Battery Saver Mode will usually be in the **battery ready** state when entering the field of an Interrogator, and in that state they are subject to sub-clause 6.4.1.3.4. All Tags in the **battery ready** state shall support the timing requirement of sub-clause 6.4.1.3.4.

It is highly desirable for battery Tags to have a means of avoiding a permanent powered condition which would rapidly deplete their batteries. Necessary support for this in the case of BAP PIE Tags is provided in

sub-clause 7.4.2 on Battery Saver Mode. Tags supporting any of the low power modes (**hibernate** or **sleep-listen**) shall support at least one of the "return to low power" timers INACT_T and Global Timeout (see sub-clause 7.3.2).

7.3.2 Signal loss tolerance via timer (mandatory)

7.3.2.1 Signal loss immunity requirements

Received signal strength as viewed at the Tag suffers strong time variation due to multipath fade, resulting in rapid signal loss and return of signal. The time period of these signal nulls under typical operating conditions is in the range of approximately 1 to 100 ms (see Annex P for detailed information) and suffers attenuations of up to tens of dB. Tags may also temporarily completely lose Interrogator signal due to deliberate Interrogator actions such as frequency hopping.

The passive Tag generally loses its operating power within a few forward symbol periods of loss of signal. It then must lose its state machine position (go to the implied de-energized state). At this time passive Tags begin their inventory persistence (a timing function controlling flag state) operation so that they temporarily retain **inventoried** state.

BAP Tags do not lose power until the battery is depleted, hence this forced passive Tag behaviour does not apply. The BAP Tag may take appropriate advantage of its battery by showing a general behaviour whereby the Tag holds its operating state (both **inventoried** flag and state machine state) over temporary loss of signal relative to its sensitivity level. A still more sophisticated behaviour the Tag may display is to hold its state for an interval of time from the last valid command received (environmental validation), so that the Tag recognizes that signals above threshold may actually be from interfering sources that the Tag should decline to allow holding the Tag active. It should be noted that these two behaviours have an opposite behaviour with regards to when timers are in operation. When a timer is used to time loss of signal relative to a threshold, the timer begins when signal is lost and so is normally not running. When a timer is used to measure time from last valid command or preamble, the timer begins upon entering the **battery ready** state, and is reset to begin again on every valid command or preamble, so the timer is always running. Note also that a timer function that validates the environment can avoid the Tag being trapped in an on state in the presence of a strong interferer.

Tag reaction to signal loss is controlled by one or both of the below specified timers, which are applicable to all the BAP Tag types and which may have more advanced behaviours of command recognition. These are the INACT_T and Global Timeout timer functions. The INACT_T function is the primary timer function that holds the Tag in the state it is in for some period of loss of signal or absence of valid commands. Some form of INACT_T function is mandatory, though it may be specified as a reaction time that applies without requiring an explicit timer. The Global Timeout is an optional timer that can recover the Tag from being trapped on in the presence of an interfering carrier. It is thus most useful for situations where environmental validation is not performed, since a single timer with environmental validation can suffice to both survive fades and not be trapped on by interference. These timers are described in detail in the following sub-clauses.

7.3.2.2 INACT_T timer

The period of time the PIE or Manchester BAP Tag maintains its state within the singulation and access state machine and by which it delays the initiation of flag persistence timers without received RF signal (or optionally without correctly decoded signals) is governed by a timer whose timeout period is "INACT_T". This timer is defined for both BAP PIE and Manchester Tags. Values for INACT_T range from near zero (microseconds) to a few seconds, with 100 ms being a typical value to survive multipath fades typical in the modest velocities of RFID systems. INACT_T also allows for interrogation rounds to span over multiple frequency hopping intervals as used in countries where frequency hopping is a regulatory requirement.

In general there are two modes of INACT_T timer operation, specified as follows:

1. Threshold INACT_T. The Threshold INACT_T form begins timer operation upon loss of signal relative to the manufacturer determined threshold. After the manufacturer specified INACT_T period the Tag returns to the **stateful sleep** or **stateful low power listen** state for BAP PIE Tags featuring Battery Saver Mode, or to the **hibernate** state for Manchester (see Figure 34). With this simple form of INACT_T behaviour there is the chance that an interfering source of power above the threshold can cause a Tag to remain in a higher power

mode. This is referred to as an “interference trap”. For BAP PIE Tags using Threshold INACT_T the manufacturer has the option of providing a separate “Global Timeout” timer (see 7.3.2.3) that will defeat the interference trap and return the Tag to the **battery ready** state. For Manchester Tags using Threshold INACT_T the inclusion of a Global Timeout is mandatory, and it shall return Manchester Tags to **hibernate** in the presence of interference.

If a BAP Tag implements Threshold INACT_T, its timing actions need not be with respect to a single RF signal level. Hysteresis may be employed where the Tag starts the timer running (loss of signal) on one signal level and resets the timer (reacquisition of signal) on a different signal level. It is recommended to implement hysteresis between an “RF signal detected” level and an “RF signal absent” level to avoid bounce at the threshold level.

If a BAP Tag implements Threshold INACT_T, then the monitoring of signal need not be continuous. The Tag may sample signal strength at discrete times, and internally consider signal to be present or absent after a number of such samples are taken and weighed according to an algorithm of the manufacturer’s choice. The total time period applicable to INACT_T then becomes the sum of the individual periods of the number of samples taken to reach this decision.

Some INACT_T behaviour is specified as mandatory even if the Tag does not have actual timers, as there is always some greater than zero time delay before the Tag reacts to signal loss. If the Tag does not possess an actual INACT_T timer, then there is a small period of time generally in the range of microseconds before the Tag reacts to signal loss, and this time shall be specified by the Tag manufacturer as a fixed Threshold INACT_T period. When INACT_T is not supported with a timer, Tags will eventually reset from an inventory or access state to a low power mode via a failsafe “Global Timeout” timer (see 7.3.2.3).

If INACT_T is not supported with a timer then it shall be considered as INACT_T = 0 (or near zero as specified by the Tag manufacturer) for purposes of persistence (see Table 55). The no explicit timer case shall be considered as INACT_T = Infinity for purposes of how long a Tag holds a state upon loss of signal or loss of valid Type C preambles or commands, with the Global Timeout function then recovering from a BAP Tag being trapped in an ON state.

2. Validated INACT_T. This INACT_T form starts timer operation upon the Tag entering the **battery ready** state, and resets the timer to zero to begin again upon reception of a valid command or preamble. In other words, this INACT_T form validates the environment before resetting the INACT_T timer. If the INACT_T timer expires without a preamble or command causing reset, then the Tag returns to either the **stateful sleep** or **stateful low power listen** state for BAP PIE, or to the **hibernate** state for Manchester (see Figure 34). It is left to manufacturer’s options what it chooses to constitute a valid command or how these may vary as a function of Tag state. For example, if a Manchester Tag is awakened from **hibernate** with Session Locking (see **Table 64** and its notes) in effect, then the may consider that only Interrogator commands with a **inventoried** flag match constitute valid commands for resetting the Validated INACT_T timer.

BAP PIE Tags that support Battery Saver Mode shall implement at least one of INACT_T or Global Timeout. Further details of the behaviour of BAP PIE Tags upon expiration of the INACT_T timer are described in sub-clause 7.4.2. Since Global Timeout is specifically defined to apply to return to a low power mode, BAP PIE Tags that do not support Battery Saver Mode shall support INACT_T to provide a means to force them back to the **battery ready** state upon a sufficient period of loss of signal or loss of Type C commands. The expiration of the INACT_T or Global Timeout functions may override the responses to T2 timeout described in 6.4.1.6.

If the Tag supports TCRS, then the value of INACT_T shall be indicated in TCRS, and may optionally be reprogrammable via TCRS. A BAP Tag that is programmed to INACT_T greater than 500 ms shall apply a maximum of 500 ms delay to the start of its inventory persistence timers when operating in BAP PIE mode (see Table 55).

The INACT_T timer shall display an accuracy of +/-40% accuracy over nominal temperature range of -25 to + 40 °C, and (if supported) a +/-50% accuracy over extended temperature range of -40 to +65 °C.

7.3.2.3 Global Timeout timer and Selective Global Timeout

This optional timer causes the Tag to fall back to a low power state (**sleep**, **low power listen**, or **hibernate** as appropriate) a relatively long time after it was activated to the Normal Mode. The Global Timeout may at the option of the Tag manufacturer be of either of these two forms:

1. Global Timeout. This form forces the Tag back to its low power mode regardless of Tag state, such as the Tag currently being involved in a legitimate inventory round. If implemented in this way, then in order to reduce the odds of an intended inventory round from being interrupted, this Global Timeout shall be 4 seconds or more. Global Timeout begins running when the Tag enters the **battery ready** state, and upon expiration and regardless of what the Tag is doing takes the Tag back to the **stateful sleep** or **stateful low power listen** state for BAP PIE, or to the **hibernate** state for Manchester (see Figure 34). It may thus affect a Tag engaged in a legitimate inventory operation.

2. Selective Global Timeout. This form is selective as to Tag state. Selective Global Timeout has the behaviour that it begins running when the Tag enters the **battery ready** state, but that the timer refreshes upon the Tag detecting legitimate commands. This generally avoids causing a Tag engaged legitimate operations from timing out. An example of selective behaviour is resetting Selective Global Timeout upon receiving valid commands, with different options possible as to what is considered a valid command. These options are at the choice of the Tag manufacturer, but should be described in product data sheets. The Selective Global Timeout, if implemented, shall have a minimum duration of at least 50 ms.

If the Tag supports TCRS, then the value of Global Timeout and whether it is of the Selective form shall be indicated in TCRS, and may optionally be reprogrammable via TCRS.

The Global Timeout shall display an accuracy of +/-40% accuracy over nominal temperature range of -25 to +40 °C, and (if supported) +/-50% accuracy over extended temperature range of -40 to +65 °C.

BAP Tags shall support at least one of the INACT_T or Global Timeout timers. The only currently specified requirements as to choices are that:

1. BAP PIE Tags that do not implement Battery Saver Mode shall implement INACT_T (such Tags do not have a low power state for Global Timeout to apply to).

2. For Manchester Tags the INACT_T timer is optional if Global Timeout is provided, but if INACT_T is provided then it shall be of the Validated INACT_T form, and the minimum time of INACT_T shall be 50 ms. Note that Validated INACT_T allows a single timer to avoid the interference trap problem, and so with Validated INACT_T a Global Timeout is usually not necessary (an exception could be for Tags that spend long periods of time near active Interrogators). More detailed behaviour of Manchester Tags is described in sub-clause 7.5.3.6.

7.3.3 Modified persistence of BAP PIE inventory flags (optional)

BAP PIE Tags shall provide for persistence behaviour either identical to that of passive Tags or enhanced with immunity to fade via the INACT_T timer and more precise persistence maximum values. Manchester Tags shall support persistence in a different way, via returning to the **hibernate** state and running an accurate persistence timer that (unlike passive or BAP PIE) has identical behaviour for all four **inventoried** flags.

For BAP PIE Tags the optional specification on maximum flag persistence can reduce the incidence of needing to inventory Tags from **inventoried** flag state *A*→*B* and then again from *B*→*A*. The specification fits within and is backward compatible to the persistences defined for pure passive in Clause 6 in Table 17. Tags that have Dead Battery Response may in the case of a dead battery continue to support the persistences of Table 55, or may revert to meeting the more relaxed requirements of Table 17 of Clause 6.

If TCRS is supported, then the adoption or not of these optional maximums shall be documented in TCRS.

The values for optional BAP PIE flag persistence are as shown in Table 55. The flags in this table shall have the same functions as the passive **inventoried** and **selected** flags, simply with the specified maximums and delay time INACT_T that may be chosen to apply while the Tag has battery power. If a BAP PIE Tag that

supports Battery Saver Mode does not support INACT_T, then for the purposes of Table 55, INACT_T shall be considered as zero.

Table 55 — Optional improved Battery Assisted Passive PIE flag persistence values (the Tag manufacturer may comply with the persistence values of this table, but is only required to comply to the more relaxed passive persistence values of Table 17)

Flag	Persistence (See NOTES 1-5)
S0 **inventoried** flag	Above sensitivity threshold or successful decoding: Indefinite Below sensitivity threshold or failure to decode : Zero (following INACT_T, see NOTE 1)
S1 **inventoried** flag[a]	Above sensitivity threshold or successfully decoding: –25 °C to +40 °C: 500 ms < persistence < 5 s (beginning after INACT_T) –40 °C to -25 °C and +40 °C to +65 °C: Not specified if Tag does not cover extended temp range, but 500 ms < persistence < 5 s if Tag does cover extended temp range (beginning after INACT_T) Below sensitivity threshold or failure to decode: –25 °C to +40 °C: 500 ms < persistence < 5 s (beginning after INACT_T) –40 °C to -25 °C and +40 °C to +65 °C: Not specified if Tag does not cover extended temp range, but 500 ms < persistence < 5 s if Tag does cover extended temp range (beginning after INACT_T) See NOTE 3 for the special case of this flag timing out in the **battery ready** state.
S2 **inventoried** flag[a]	Above sensitivity threshold or successfully decoding: Indefinite Below sensitivity threshold or failure to decode: –25 °C to +40 °C: 2 s < persistence < 20 s (beginning after INACT_T) –40 °C to -25 °C and +40 °C to +65 °C: Not specified if Tag does not cover extended temp range, but 2 s < persistence < 20 s if Tag does cover extended temp range (beginning after INACT_T)
S3 **inventoried** flag[a]	Above sensitivity threshold or successfully decoding: Indefinite Below sensitivity threshold for failing to decode: –25 °C to +40 °C: 2 s < persistence < 20 s (beginning after INACT_T) –40 °C to -25 °C and +40 °C to +65 °C: Not specified if Tag does not cover extended temp range, but 2 s < persistence < 20 s if Tag does cover extended temp range (beginning after INACT_T)
selected (**SL**) flag[a]	Above sensitivity threshold or successfully decoding: Indefinite Below sensitivity threshold or failing to decode: –25 °C to +40 °C: 2 s < persistence< 20 s (beginning after INACT_T) –40 °C to -25 °C and +40 °C to +65 °C: Not specified if Tag does not cover extended temp range, but 2 s < persistence < 20 s if Tag does cover extended temp range (beginning after INACT_T)

NOTE 1 The beginnings of the persistence time periods for S0, S1, S2, S3, and **SL** are all delayed by manufacturer specified signal loss or failure to decode tolerance time INACT_T. For the purposes of this table, if INACT_T is not supported then it shall be regarded as zero (or near zero hardware delay as specified by the Tag manufacturer), in which case the persistence timing begins just as it does for passive Tags.

NOTE 2 A BAP Tag that is programmed to INACT_T greater than 500 ms and complying with this table shall apply a maximum of 500 ms delay to the start of its **inventoried** persistence timers when operating in BAP PIE mode.

NOTE 3 Though the S1 flag may run its persistence timer in the **battery ready** state (if it enters **battery ready** with S1 in B) and may time out in the **battery ready** state, while in a Query round it suspends persistence timer operation (it will not time out and invert its state from *B*→*A* while in the inventory and access process). The other flags all have indefinite

persistence, meaning they hold their *B* or **SL** state until RF carrier loss (defined as carrier strength below manufacturer defined threshold) or decode loss have occurred for the period defined by INACT_T.

NOTE 4 This standard uses *A* and *B* as proxy logic states for **inventoried** flags with the understanding that the electrical logic state is implementation dependent. Similarly the **selected** flag uses proxy states **SL** (asserted) and **~SL** (deasserted). The default state (following persistence) of the **inventoried** flags is state *A*, and that of the **selected** flag is **~SL**. Thus inventory timers controlling persistence need only operate if the state is *B* or SL, and upon expiration of the timer these flags change state from *B*→*A* and from **SL** to **~SL**.

NOTE 5 The accuracy of these flags is required to at least match the passive Tag case, which is as follows. Those flags denoted with a superscript "a" in Table 55 shall for a randomly chosen and sufficiently large Tag population shall behave such that at least 95% of the Tag persistence times shall meet the persistence requirements in Table 17, with at least a 90% confidence interval. Tags that have accurately specified tolerances on persistence timers may meet these requirements by reducing their range of persistence to allow for timer error. See P.11 for analysis.

7.4 Battery Assisted Passive PIE (optional)

7.4.1 *Flex_Query* command (optional)

The basic function of this optional command is to provide a "Tag Type Select" function that allows for selecting the categories of Tags to be included in the query round without requiring a separate *Select* command. This selection allows for the Interrogator to bring nearly any combination of Tag types into Query rounds.

Tags that support Dead Battery Response and *Flex_Query* may continue to support *Flex_Query* with a depleted battery.

Table 56 — *Flex_Query* command (optional) (See NOTES 1-3)

	Command	Tag Type Select	SS Resp NOTE 2	MRFID Response	DR	M NOTE 3	TRext	Sel	Session	Target	Q	CRC-5
# of bits	8	12	1	1	1	4	1	2	2	1	4	5
description	11001111	NOTE 1	0: No 1: Yes	0: Disable 1: Enable As defined in ISO/IEC 29143	0: DR=8 1: DR=64/3	0000: M=1 0001: M=2 0010: M=4 0011: M=8 0100: M=16 0101: M=32 0110: M=64 0111 to 1111: RFU	0: No pilot tone 1: Use pilot tone	00: All 01: All 10: **~SL** 11: **SL**	00: S0 01: S1 10: S2 11: S3	0: *A* 1: *B*	0–15	

NOTE 1 See Table 57 for full definition of Tag types to be pulled into the Query round. Possible implementation of *Flex_Query* by passive Tags is a future option.

NOTE 2 The Simple Sensor Response flag authorizes automatic transmission of Simple Sensor data in cases where sensor Tags have been selected. Tags without Simple Sensors shall ignore the SS Response flag.

NOTE 3 M values of 16, 32, and 64 are optional for Tags implementing the *Flex_Query* command. A Tag that receives a *Flex_Query* command for an unsupported M value shall ignore the command.

Table 57 — *Flex_Query* command Tag Type Select field (see NOTES 1-2)

Interpretation NOTE 2	RFU	MRFID	Sensor Alarm	Full Function Sensor	Simple Sensor	RFU	RFU	RFU	Battery Assisted Passive	RFU	Passive NOTE 1 (RFU)
1	1	1	1	1	1	1	1	1	1	1	1
0: Inclusive 1: Exclusive NOTE 2		0: Disable 1: Enable As defined in ISO/IEC 29143	0: Disable 1: Enable	0: Disable 1: Enable	0: Disable 1: Enable				0: Disable 1: Enable		0: Disable 1: Enable

NOTE 1 *Flex_Query* is not currently defined for passive Tags, but the Passive select field is noted for when it becomes defined.

NOTE 2 This field switches the interpretation of the following selection fields between "inclusive" (the Tag responds if it matches any selection) and "exclusive" (the Tag responds only if it matches all selected criteria).

BAP PIE Tags that implement *Flex_Query* shall respond to *Flex_Query* by beginning a new Query round from the **battery ready**, **arbitrate**, **reply**, **acknowledged**, **open**, and **secured** states.

Just as in the normal *Query* command, the *Flex_Query* command is preceded by R=>T Preamble and not R=>T Frame-Sync.

There is no reply to *Flex_Query* unless the Tag rolls zero on its slot counter, in which case it shall reply with its RN16 as in Table 58.

Table 58 — Tag reply to a *Flex_Query* command (only if slot counter = 0)

	Response
# of bits	16
description	RN16

7.4.2 BAP PIE detailed operation including optional Battery Saver Mode

This sub-clause provides detailed specification of BAP PIE mode both with and without Battery Saver functionality. A state machine description is used to provide a logical framework applicable to both of these cases and which integrates with the passive Type C state diagram of Figure 19. This passive state diagram is a subset of the BAP PIE state diagram. Whether the Tag formally follows a state machine according to Figure 26 is an implementation issue, but from the Interrogator point of view the Tag implementing BAP PIE shall act as if it does according to the functional behaviours and times specified in this sub-clause.

A Tag providing BAP PIE mode may employ Battery Saver functionality to reduce the average battery current drain when the Tag is not in the field of an Interrogator. This mode has two main options of which a Tag using this mode shall provide at least one and may provide both.

The first option is the use of Low Duty Cycle (LDC) mode. LDC is defined as the following ratio:

"**listen** state" time / cycle time, where cycle time = "**listen** state" time + "**sleep** state" time.

In **listen** state the Tag samples the presence of Interrogator RF radiation, generally with higher sensitivity and relatively high current drain. In **sleep** state all Tag circuits are disabled, except the duty cycle timer, draining very low current from the battery.

Neglecting small changes due to processing within Query rounds, the average battery current drain is approximated by:

sleep state current * (1- LDC) + (**listen** state current) * LDC

LDC causes a delay in the Tag "Power-up" time defined in clause 6.4.1.3.4. The maximum power-up delay of the BAP PIE Tag in Low Duty Cycle Battery Saver Mode is given by the sum of the cycle time and the appropriate delay as described in clause 6.4.1.3.4.

The second option is a low power mode with a continuous built in listen capability (typically not as sensitive as a part time **listen** mode). If the Tag supports this continuous low power listen capability as its lowest power state, then this state is referred to as "**low power listen**" instead of "**sleep**". See Figure 34 for a suggested state machine diagram.

The Tag has relaxed timing requirements before being required to be in the **battery ready** state following initiation of Interrogator RF carrier while using Battery Saver Mode. This relaxed time to **battery ready** allows for these more sophisticated Tags to provide for settling time of regulators and clock sources that may be off while in the **sleep**-**listen** mode (see below in this sub-clause), and for fetching key data from TCRS memory and setting up the Tag accordingly.

The possible BAP PIE modes are specified in detail as follows and with reference to Figure 34. This figure provides a complete state description of BAP PIE Tags that both support or do not support a Battery Saver Mode. If a Battery Saver Mode is supported, this figure also describes both duty cycled and non-duty cycled behaviour. The figure also covers use of timers for INACT_T and Global Timeout.

The states in Figure 34 are defined as follows:

sleep: A near zero power mode where the Tag is not responsive to RF signals. The Tag transitions from **sleep** to **listen** only under control of an internal timer, and does so within a manufacturer selected **sleep** to **listen** time S_to_L. It is in **sleep** for the manufacturer chosen period **sleep** time or ST. ST may also be a range of times that may change under operating conditions, such as to avoid interference induced false wake-ups. If the Tag supports TCRS, ST or its range will be documented in TCRS. ST may also be optionally programmed via TCRS.

stateful sleep: This is the **sleep** state with the addition of one or more **inventoried** or **selected** flag persistence timers in operation. The Tag may also cycle through the **stateful sleep** state while briefly checking if such timers are in operation and if not then transition to **sleep**. It is comparable to the **stateful hibernate** modes of Manchester. It responds to internal duty cycle timer control as does **sleep**.

low power listen: A replacement for **sleep** where the Tag is continuously responsive to RF signals, though generally using low power and thus only achieving modest sensitivity. It is typically used without duty cycling to a more sensitive state, but may be used in combination with duty cycling. The Tag shall transition out of **low power listen** to **battery ready** upon an RF signal greater than its manufacturer defined threshold within manufacturer selected **low power listen** to **battery ready** time LPL_to_R of 20 ms or less. The Tag may optionally also transition from **low power listen** to **listen** under control of an internal timer, and if so then the Tag does this within **low power listen** to **listen** time LPL_to_L.

stateful low power listen: This is the **low power listen** state with the addition of one or more **inventoried** or **selected** flag persistence timers either in operation or being checked for operation. It is also comparable to the **stateful hibernate** mode of Manchester. It responds to RF signals as does **low power listen**. The Tag is considered to transition from **stateful low power listen** to **listen** upon confirming expiration of the last of its **inventoried** or **selected** flag persistence timers.

listen: A part time state where the Tag is responsive to RF signals, typically but not necessarily with improved sensitivity. This state is used in a Duty Cycled Battery Saver Mode. The Tag is not required to decode signals

in the **listen** state, but shall at least detect their presence. The Tag shall transition from **listen** to **battery ready** upon either receiving a signal above a manufacturer defined threshold or if validating the environment upon correctly decoding either a PIE preamble or command (the degree of environmental validation is up to the manufacturer). The transition from **listen** to **battery ready** shall occur within manufacturer selected **listen** to **battery ready** time L_to_R of 20 ms or less. The Tag shall remain in **listen** for manufacturer selected **listen** time LT, and if no RF is detected or no correct PIE symbol or command is decoded the Tag shall then return to **sleep**, **stateful sleep**, **low power listen**, or **stateful low power listen** as appropriate. Transitions to these low power states shall be within manufacturer chosen **listen** to **sleep** time L_to_S. If the Tag supports TCRS, LT will be documented in TCRS. LT may also be optionally programmed via TCRS.

stateful listen: Identical to **listen** except at least one persistence timer is in operation. Upon expiration of the persistence timers the **stateful listen** state is considered to transition to the **listen** state.

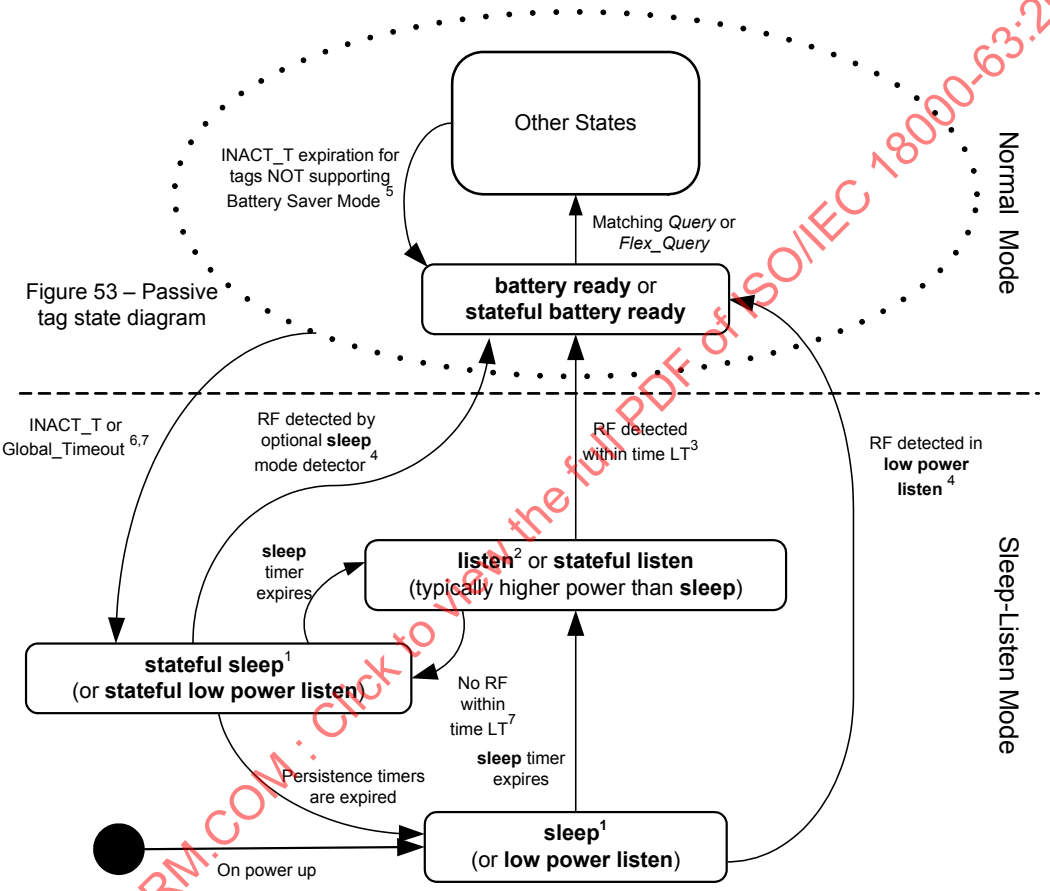


Figure 53 – Passive tag state diagram

Notes:

1. **Sleep** may optionally also "listen", though normally at reduced sensitivity, in which case the name of the state changes to **low power listen**.
2. If **sleep** has its own "listen", then the separate **listen** state (generally at improved sensitivity) is optional. The listen timer period may be increased and/or **sleep** period decreased by the Tag in response to recent activity. They may also be adjusted by the tag near the end of battery charge in order to increase battery life.
3. A transition from **listen** or **stateful listen** to **battery ready** occurs within L_to_R time <= 20 ms.
4. A transition from **low power listen** or stateful low power listen to **battery ready** (if supported) occurs within LPL_to_R time <=20 ms.
5. If the BAP Simple PIE tag does NOT support Battery Saver Mode, then it runs its persistence timers (holds inventory state) in the **battery ready** state.
6. If the BAP Simple PIE tag does support Battery Saver Mode, then it runs its persistence timers (holds inventory state) in the **stateful sleep** or **stateful low power listen** state, and upon expiration transitions to the full **sleep** state or **low power listen** state. These timers only operate if the inventory state is B or the **selected** state is **SL**, and upon expiration the states change to A or **~SL** as appropriate.
7. The "stateful" states mean timers are running or that the tag is checking if they are running. The tag manufacturer has the option of going through the stateful states to check if timers are running, or checking in the prior state and going straight to sleep or **low power listen**.

Figure 26 — BAP PIE and Battery Saver Mode state diagram

Battery ready is specified in sub-clause 7.3.1. The remainder of the states under the block called “Other States” are the passive Tag states other than **ready** shown in Figure 19. Being in the **battery ready** or any of the passive Tag states is referred to as “Normal Mode”. Being in any of **sleep**, **stateful sleep**, **low power listen**, **stateful low power listen**, or **listen**, is referred to as being in “Sleep-Listen Mode”.

If a BAP PIE Tag transitions from any stateful state to the Normal Mode, it shall reset its persistence timers to zero and halt their operation while holding its current flag states indefinitely (while RF or decoded RF has not been absent longer than INACT_T) with the exception of the S1 **inventoried** flag. For the S1 flag its persistence timers continue to operate and hold the state in *B* (it would not be running if the S1 flag was already in A) while the timer is operating in the **battery ready** state (see S1 description and related note of Table 55). If the S1 timer expires in **battery ready**, the S1 flag change from *B*→*A*. If the S1 flag is set to *B* via a *Select* command, then the persistence timer is “refreshed” (set to zero and restarted with the Tag in the **battery ready** state). If the Tag is taken into a Query round based on the S1 flag being in state *B*, then the timer is set to zero and not restarted. The Tag then holds S1 in state *B* while in the Query round, until it is appropriately changed to *A* via an inventory action.

The transitions between states in the Normal Mode are identical to that exhibited for passive Tags with the exception of delays and return to Sleep-Listen Mode as dictated by the manufacturer’s implementation of INACT_T. INACT_T may be implemented with respect to RF signal strength in relation to a threshold, or with relation to correct receipt of valid Type C preambles or commands.

INACT_T normally delays the start of persistence timeout and state machine state changes upon loss of RF or decoded RF. However, according to sub-clause 6.4.1.6 on Link Timing, the maximum time of the T_2 timer and its specified state changes apply instead of INACT_T while the Tag is in the **reply** or **acknowledged** states.

Global Timeout does not take RF signalling or Tag position in the Normal Mode state machine into account, but Selective Global Timeout may take these factors into account. See sub-clause 7.3.2.3.

Environmental validation for BAP PIE: If INACT_T is supported relative to valid Type C preambles or commands (Validation INACT_T), this allows the opportunity to validate the radio environment that awakened a Tag as a legitimate Type C environment. Validation INACT_T shall be conducted as follows: The Tag transitioning from a Sleep-Listen Mode to **battery ready** shall in **battery ready** listen for valid Type C preambles or commands at the manufacturer’s choice. If the Tag does not decode valid Type C signals within INACT_T, it shall return to the **sleep** or **low power listen** state as appropriate. The Tag may upon manufacturer choice increase its **sleep** time ST to avoid non-Type C signals causing further false wake ups. If it does detect valid Type C signals within INACT_T, the Tag shall reset the INACT_T timer. The Tag may then perform one of two actions:

1. Continue to reset the INACT_T timer indefinitely upon receipt of valid preambles or commands. In this case Global Timeout would be recommended to eventually force the Tag back to the **sleep-listen** mode.

2. Wait for valid *Select* and *Query* commands (or *Flex_Query* command) to take it into a Query round, while resetting the INACT_T timer for a limited number of manufacturer selected times. If the Tag has not entered a Query round upon expiration of INACT_T for the manufacturer selected number of times, it may return to the Sleep-Listen Mode. If the Tag is taken into a Query round it shall again reset the INACT_T timer.

While a Tag is in a Query round it shall upon receipt of further valid Type C preambles or commands continue to reset the INACT_T timer. See Annex P for description of various applicable command validation methods. If signal is lost the Tag shall hold state machine state and not begin persistence timer operation (if the Tag supports BAP PIE persistence as defined in Table 55) until the expiration of the INACT_T timer (with the exception of the requirements of sub-clause 6.4.1.6). Tags that support a Battery Saver Mode shall then transition to the **sleep**, **stateful sleep**, **low power listen**, or **stateful low power listen** state as appropriate (the stateful states are used if any **inventoried** flag is in state *B* or if **selected** is in state **SL**). Tags that do not support Battery Saver Mode shall upon expiration of INACT_T transition to the **battery ready** state and (if the Tag supports BAP PIE persistence as defined in Table 55) begin persistence timer operation if any **inventoried** flags are in state *B* or if **selected** is in state **SL**. That condition may be considered as a **stateful battery ready** state, and upon expiration of persistence timers the **inventoried** and **SL** flag state will change as appropriate.

Environment not validated: In this case INACT_T is not supported or if supported is responsive to signal strength without regard to decoding of valid Type C signals (Threshold INACT_T). The Tag transitioning from a Sleep-Listen Mode state to the Normal Mode that supports Threshold INACT_T shall in any Normal Mode state listen for RF signals above its manufacturer defined threshold. When such are detected the INACT_T timer is reset (held in reset if RF signal is continuously detected above threshold), and when signal drops below threshold (optionally a different threshold than that necessary to reset the INACT_T timer) then the INACT_T timer will begin. If the INACT_T timer expires the Tag returns to the Sleep-Listen Mode (if a Battery Saver Mode is supported) or to the **battery ready** state (if a Battery Saver Mode is not supported) as appropriate.

Now the various BAP PIE cases may be completely specified as follows.

Battery Saver Mode not supported: In this case the BAP PIE Tag is in the **battery ready** state when not in the field of an Interrogator, and behaves nearly identically to a passive Tag (though typically with better sensitivity). The Tag responds to the application of RF carrier within the timing limits of sub-clause 6.4.1.3.4 (1.5 ms to be ready to detect a preamble). Since the Tag does not automatically reset to the **battery ready** state upon loss of power, the Tag shall implement INACT_T to force the Tag back to the **battery ready** or **stateful battery ready** (when persistence timer functions are in operation) state when it is not in range of an Interrogator. The Tag manufacturer may support the persistence requirements of Table 55 for BAP PIE Tags (persistence delayed by INACT_T and with the specified maximums), or may support the more relaxed persistence requirements of passive Tags as given in Table 17. The Global Timeout is not applicable to this case.

Battery Saver Mode supported via low power listen only: In this case the Tag is not supporting duty cycling of **listen** and maintains a continuous low power **listen** mode. It shall transition to the **battery ready** state upon detecting RF field above its manufacturer determined threshold within LPL_to_R time of 20 ms. This longer allowed time to be in the **battery ready** state is what distinguishes this case from Battery Saver Mode not supported. Since the Tag does not automatically reset to the **low power listen** state upon loss of power, the Tag shall implement one of INACT_T or Global Timeout to force the Tag back to the **low power listen** state when it is not in range of an Interrogator. When the Tag departs Normal Mode it shall transition to either the **stateful low power listen** state (if no timers running is not confirmed before leaving Normal Mode) or **low power listen** state (if no timers running is confirmed before departing Normal Mode). The reason for allowing the Tag the option of briefly transitioning to **stateful low power listen** for the timer check operation is to simplify state machine or firmware operation. The Tag shall transition from the **stateful low power listen** state to the **low power listen** state upon confirming expiration of all persistence timers.

Battery Saver Mode supported via duty cycling only: The Tag shall **listen** (generally in a relatively high sensitivity state) for time LT and **sleep** for time ST. It shall transition to the **battery ready** or **stateful battery ready** state upon detecting RF field while in the **listen** state above its manufacturer determined threshold within L_to_R time of 20 ms. Since the Tag does not automatically reset to the **sleep** state upon loss of power, the Tag shall implement one of INACT_T or Global Timeout to force the Tag back to the **sleep** state when it is not in range of an Interrogator. Upon departing Normal Mode, the Tag shall transition to either the **stateful sleep** (if timers running or to confirm if timers running) or **sleep** state (if it confirms no timers running). The Tag will continue to operate its duty cycle control to transition to the **stateful listen** mode and back while in **stateful sleep**. The Tag shall transition from the **stateful sleep** state to the **sleep** state upon confirming expiration of all persistence timers.

In the **listen** mode the Tag is standing by to enter the **battery ready** state upon application of an RF carrier. The **listen** Time (LT) that the Tag maintains itself in the **listen** state in the absence of RF signal may be selected by the manufacturer, or in the case of Tags supporting TCRS it may optionally be programmable. If the Tag does not receive a signal above a manufacturer defined threshold level during the LT interval, then the Tag shall transition back to the **sleep** state. If the Tag does receive any signal above the manufacturer defined sensitivity threshold, then the Tag shall transition to the **battery ready** state and will do so within the timing limit of **listen** to **battery ready** time L_to_R <= 20 ms.

In the **sleep** state the Tag is running a timer of duration **sleep** time (ST) and is waiting to return to the **listen** state upon expiration of the ST timer. ST may be fixed by the manufacturer. In the case of Tags that support TCRS the ST shall be documented in TCRS if fixed, and may optionally be programmable. ST is typically on the order of 10 ms to 1 s. ST is chosen shorter for higher Tag velocity environments or longer for slow moving environments, with ~100 ms being a typically chosen or programmed value. Though manufacturers are free to select any ST, they should specify their chosen ST within their product documentation. Programmable ST ranges are as shown in sub-clause 7.6.3.

Battery Saver Mode supported via duty cycling and low power listen: In this case the Tag is both supporting duty cycling of **listen** (generally in a higher sensitivity state) and maintaining a continuous low power **listen** mode (generally in a lower sensitivity state) when not in **listen**. It shall transition to the **battery ready** or **stateful battery ready** state upon detecting RF field above its manufacturer determined thresholds within L_to_R or S_to_R time as appropriate. Since the Tag does not automatically reset to the **low power listen** state upon loss of power, the Tag shall implement one of INACT_T or Global Timeout to force the Tag back to the Sleep-Listen Mode when it is not in range of an Interrogator. While in **low power listen** the Tag also operates its **listen-sleep** timer to periodically transition to the **listen** mode to listen in a higher sensitivity state than in the **low power listen** state. Other operation is as described above for duty cycle operation only or low power listen operation only.

Table 59 below provides a summary of the various delay times and timers that apply to BAP PIE Tags.

Table 59 —Battery Saver Mode timing parameters

Timing parameter	Description	Effects & comments
LT	**listen** time, the time the Tag is listening for any RF above a threshold before the Tag goes back to the **sleep** or **low power listen** state.	Manufacturer defined or user selected via TCRS.
ST	**sleep** time, the low power consumption time the Tag is either not listening or is only listening at low sensitivity.	Manufacturer defined or user selected via TCRS. If Tag only supports a low power continuous “listen” and is not duty cycled to a higher power more sensitive **listen** state, then this time is set to infinity or not applicable.
S_to_L	**sleep** to **listen** time, also used for **low power listen** to **listen** time.	Manufacturer selected, typically << ST but not required to be a particular value.
L_to_R	**listen** to **battery ready** time, a maximum of 20 ms following signal exceeding threshold.	20 ms max, actual value documented in data sheet. Only applicable if Tag supports optional duty cycled **listen** state. This time also applies to **stateful listen** to **battery ready**.
L_to_S	**listen** to **sleep** time.	Manufacturer selected, typically << ST but not required to be a particular value. This time also applies to the **stateful listen** to **stateful sleep**, **listen** to **low power listen**, and **stateful listen** to **stateful low power listen** times.
LPL_to_R	**low power listen** to **battery ready** time, a maximum of 20 ms following signal exceeding threshold.	20 ms max, actual value chosen by manufacturer. Only applicable if Tag supports optional continuous listen while in the **low power listen** state as a replacement for the **sleep** state. This time also applies to the transition from **stateful low power listen** to **battery ready**.

Timing parameter	Description	Effects & comments
N_to_S	Normal to **sleep** time	Time from any normal state to **sleep**, **stateful sleep**, **low power listen**, or **stateful low power listen**, following expiration of INACT_T or Global Timeout. Normally << than INACT_T or Global Timeout, but not required to be any particular value.
INACT_T	A timer that allows a BAP PIE Tag to temporarily hold its state machine state in the absence of RF signal or in the absence of valid Type C preambles or commands. Is continually reset upon either signal above threshold or decoding of a valid preamble or command. Can apply to Tags that support or do not support Battery Saver Mode.	See for responses for different BAP PIE modes with respect to support of Battery Saver Mode. At least one of INACT_T or Global Timeout shall be supported for BAP PIE and Manchester Tags.
Global Timeout	A timer that allows a BAP PIE Tag to temporarily hold its state machine state in the absence of RF signal or in the absence of valid Type C preambles or commands. Is initiated by Tag transitioning from a **sleep-listen** Mode to the **battery ready** state, and is NOT reset by RF signal strength or correct decoding of preambles or commands. Only applies to Tags that support a Battery Saver Mode.	See for responses for different BAP PIE modes with respect to support of Battery Saver Mode. At least one of INACT_T or Global Timeout shall be supported. At least one of INACT_T or Global Timeout shall be supported for BAP PIE and Manchester Tags.

According to the above table, the total time of the **sleep** and **listen** cycle is ST + S_to_L + LT + L_to_S. This would normally be dominated by the **sleep** and **listen** time intervals.

When a BAP PIE Tag also supports Manchester, that Tag shall maintain any **hibernate** mode timer values and inventory states until it validates the environment as a legitimate PIE environment. That validation shall be to at least properly decode PIE commands. Upon proper validation of environment, the Tag shall clear any **hibernate** mode timers, set all **inventoried** flag states to A, set **selected** flag state to **~SL**, and stand by for Interrogator commands.

7.5 Manchester mode Battery Assisted operation protocol extensions

7.5.1 Introduction

This clause defines optional extensions of the ISO/IEC 18000-6 Type C protocol for Battery Assisted Passive RFID utilizing a Manchester mode forward link. The Manchester mode is optional, but if a Tag or Interrogator uses this optional mode, all requirements in this clause are mandatory. The extensions defined here are intended to maximize Interrogator-to-Tag read ranges and battery life. This is achieved by using DC balanced symbols in the forward link which enable the use of AC coupled inputs to high gain preamplifiers. Tags may be designed with medium to extremely sensitive receivers. For the more sensitive Tags, the Manchester command preamble length provides for sufficient time to switch between two receiver dynamic ranges to cover the large input signal dynamic range. The longer range also requires higher symbol frequency accuracy at the lower data rates which results in higher values of M. Higher accuracy is enabled at lower signal levels in Tags equipped with battery powered oscillators by providing configuration bits in the forward link commands that specify the BLF and that are not subject to measurement accuracies of TRcal and RTcal.

Compatibility with ISO/IEC 18000-6 Type C Interrogators and passive Tags is maximized through the use of methods for channel access similar to ISO/IEC 18000-6 Type C, ensuring effective channel sharing in mixed mode environments. The Interrogator talks first, and the Tags backscatter responses with random slot times identical to ISO/IEC 18000-6 Type C. The Interrogator awakens battery assisted Tags with a specific activation sequence, followed by a selection field which identifies which Tags need to respond. Tags not selected by the *Activation* command quickly return to the lower power search (**hibernate**) mode. The battery assist activation is distinctly different from ISO/IEC 18000-6 Type C commands specified in Clause 6 and thus ISO/IEC 18000-6 Type C passive Tags will not respond, and conversely battery assisted Tags compliant to this clause will not respond to ISO/IEC 18000-6 Type C passive Interrogators in battery assisted mode.

Battery assisted Interrogators currently use the same channel definitions and similar bandwidths as ISO/IEC 18000-6 Type C Interrogators. Therefore, RF coordination and channel sharing concepts apply across both battery assisted and passive modes of ISO/IEC 18000-6 Type C. The UII numbering and formatting, as well as all memory bank definitions and structure, are identical to the ISO/IEC 18000-6 Type C definition.

For battery assisted mode, the higher sensitivities and longer ranges create a higher likelihood of interference between multiple Interrogators and the Tags they are communicating with. The activation mechanism defined in this section provides for the use of Interrogator power level control. Tags can be activated and operated on in sub-groups using lower power levels for Tags that are closer. Tags that have already been singulated or accessed at lower power levels can be silenced for commandable periods while the higher power operations on Tags that are farther away from Interrogators are performed. The use of power levelling is recommended in applications that would allow such system interference reducing techniques.

NOTE As PIE is a required mode and Manchester is an optional mode for Type C, Manchester Tags must support PIE. The definition of “support PIE” is given in clause 7.2.

7.5.2 Physical layer

7.5.2.1 Interrogator-to-Tag (R=>T) communications

7.5.2.1.1 Modulation

Interrogator shall communicate with the Tag using amplitude shift keyed Manchester encoded DSB-ASK, SSB-ASK, or PR-ASK modulation in the forward air-interface link. Tags shall demodulate all three modulation types.

7.5.2.1.2 Data rates

Interrogators shall communicate with Tags using one of the following data rates, while Tags shall support all of the data rates:

8 kbit/s, 16 kbit/s, 32 kbit/s, 64 kbit/s, and 128 kbit/s.

7.5.2.1.3 Frequency accuracy

Interrogator frequency accuracy shall be +/– 10 ppm over the Nominal temperature range of -25 °C to +40 °C, and +/- 12 ppm over the extended temperature range of -40 to +65 °C. These accuracies shall apply for a manufacturer specified time interval following sale of the product. If local regulatory requirements specify tighter accuracy, the Interrogator frequency accuracy shall meet the local regulatory requirements.

7.5.2.1.4 Data encoding

The R=>T link shall use Manchester data encoding, shown in Figure 27.

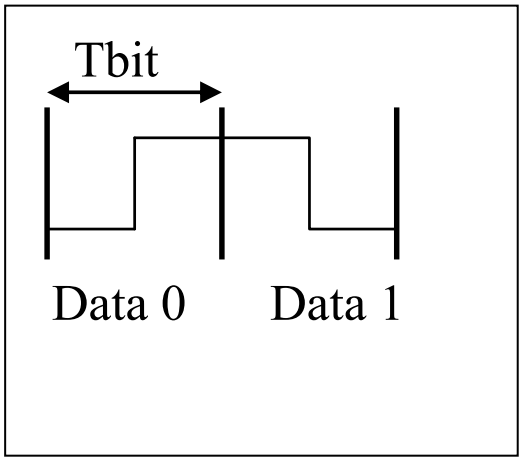


Figure 27 — Manchester symbols

7.5.2.1.5 Interrogator to Tag RF Envelope

The RF envelope (in electric field) of the Manchester modulation shall be within the limits specified in Figure 28 and Table 60.

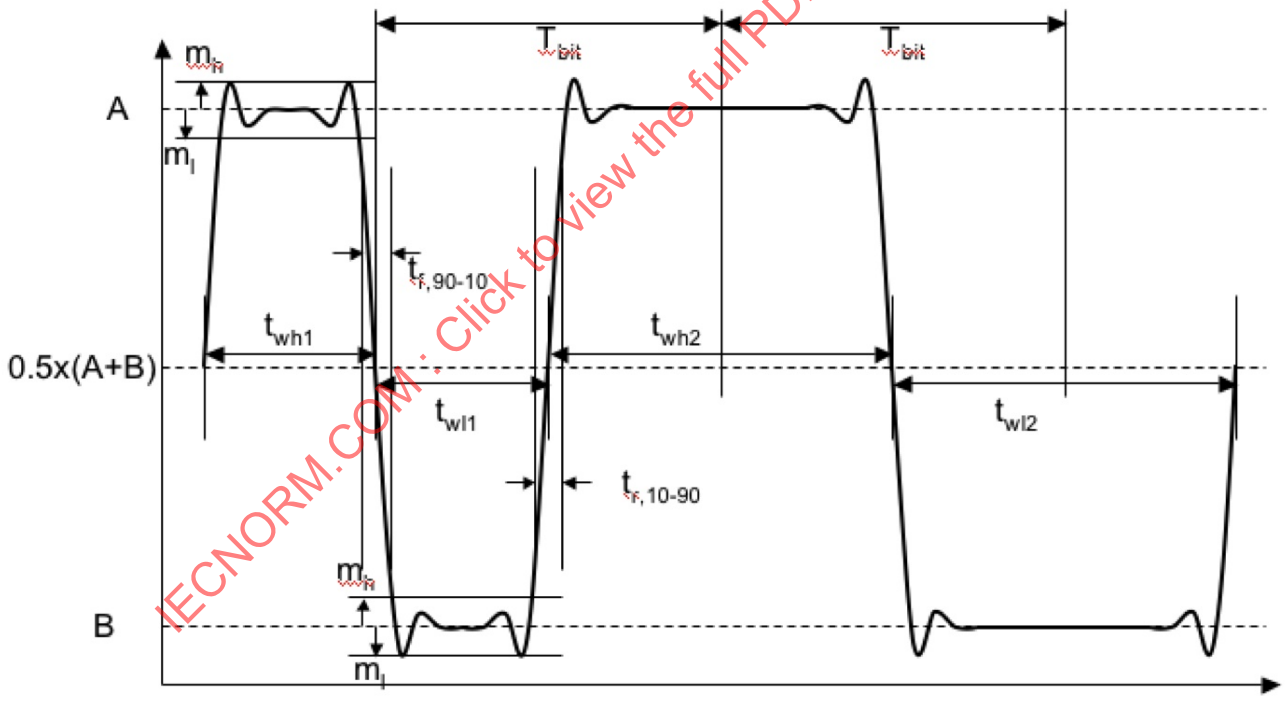


Figure 28 — Manchester waveform

Table 60 — Manchester command modulation parameters

PARAMETER	Minimum	Nominal	Maximum	Units
Tbit		1/data_rate		µs
(A-B)/A	80		90	%
mh/(A-B)	0		5	%
ml/(A-B)	0		5	%
tf,90-10/Tbit	0		33	%
tr,10-90/Tbit	0		33	%
twh1/Tbit	45		55	%
twl1 /Tbit	45		55	%
twh2 /Tbit	90		110	%
twl2 /Tbit	90		110	%

7.5.2.1.6 Interrogator to Tag normal preamble

The normal preamble consists of two parts. Both parts shall be Manchester encoded at the selected data rate indicated in the activation command. The first part is 21 bits of a pseudorandom training sequence that enables receiver dynamic range adjustment, AC coupling training, and timing acquisition. The specific sequence is a 31 bit m-sequence truncated to 21 bits and is defined by the NRZ equivalent below and the Manchester equivalent in Figure 29.

Normal preamble 21 bit training sequence = 1111100 0110111 0101000

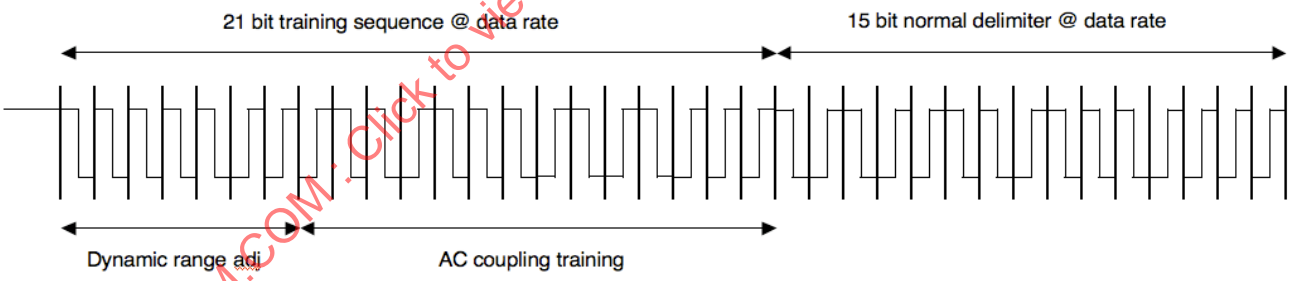


Figure 29 — Forward (R=>T) preamble

The second part is a 15 bit pseudo-random delimiter that uniquely indicates a normal command and the start of the command data. The specific normal delimiter sequence is a 15 bit m-sequence defined by the NRZ equivalent below and is the logical inverse of the activation delimiter.

Normal preamble 15 bit delimiter = 10100 11011 10000

The first bit of the command data shall follow immediately after the last bit of the delimiter at the same data rate as the preamble.

7.5.2.1.7 Transmission order

The transmission order for all R=>T communications shall respect the following conventions:

— Within each message, the most-significant word shall be transmitted first, and

— Within each word, the most-significant bit (MSB) shall be transmitted first.

7.5.2.1.8 Command data bit stuffing

Compliant Interrogators and Tags shall use selective bit stuffing on all command data bits that follow the preamble in both normal and *Activation* commands. Interrogators shall monitor the command data bit stream for the following 14 bit sequence defined in NRZ notation:

0101100 1000111

This specific pattern is the first 14 bits of the activation delimiter. If the pattern is matched exactly in the command data stream, a logical bit 0 shall be bit stuffed into the data stream at the end of this 14 bit sequence, extending the command by one bit for each bit stuffed. This eliminates the possibility of matching of the 15 bit activation delimiter in all Manchester encoded commands. The resulting 15 bit pattern will be as defined in NRZ notation:

0101100 1000111 0

Tags shall monitor the command data bit stream for the same 14 bit pattern with a logical zero in the 15th bit. The Tag shall destuff the logical zero bit from the data bit stream. The output of the destuffing algorithm in Tags will provide the command data format as defined in the commands summary of clause 7.5.4. If the 14 bit pattern is followed by a logical one, an activate delimiter has been received. If the Tag has already received a valid activation command and is processing command data, the command is invalid. If the invalid sequence was received at a data rate of 16 Kbps or higher, the Tag shall then ignore the invalid command. If the command was being received at 8 Kbps, the Tag shall ignore the invalid command and optionally transition to the **activation code check** state to process a possible new *Activation* command.

7.5.2.1.9 Cyclic Redundancy Check (CRC)

CRC calculation shall be the same as defined in clause 6.4.1.5. The order of processing in Interrogators assembling a command is:

1) Calculate the CRC beginning with the first bit of command data following the last bit of the preamble.

2) Process for bit stuffing up to the last bit of the CRC.

The order of processing in Tags handling a received command:

1) Execute the destuffing operation.

2) Calculate the CRC on the output of the destuffing operation.

7.5.2.1.10 Link timing

The Manchester Battery Assisted Passive link timing uses the same parameters as the passive mode in clause 6.4.1.6. However, the parameters T_1, T_2, T_3, and T_4 shall use the definitions in Table 61 below, with the exception for optional BLFs > 640 kHz noted below this table.

Table 61 — Manchester link timing parameters

Parameter	Minimum	Nominal	Maximum	Description
T_1	MIN(2 Tbit,$10T_{pri}$) × (1 – 2|FT|) – 2µs	MAX (2.5 Tbit, $10T_{pri}$)	MAX(3 Tbit,$10T_{pri}$) × (1 + 2|FT|) + 2µs	Time from Interrogator transmission to Tag response (specifically, the time from the end of the last bit of the Interrogator transmission to the first rising edge of the Tag response; in the case of a data 0 as the last bit, the end of the last bit is ½ bit time after the last rising edge), measured at the Tag's antenna terminals.
T_2	0.25Tbit		If T2ext=0, then 4Tbit If T2ext = 1, then max = INACT_T or Global Timeout, as selected by the Tag manufacturer	Interrogator response time required if a Tag is to demodulate the Interrogator signal, measured from the end of the last (dummy) bit of the Tag response to the first falling edge of the Interrogator transmission.
T_3	$0.0T_{pri}$			Time an Interrogator waits, after T_1, before it issues another command
T_4	4xTbit			Minimum time between Interrogator commands

NOTE 1 Tbit denotes the Manchester bit time of the normal commands.

NOTE 2 A Tag may exceed the maximum value for T_1 when responding to commands that write to memory.

NOTE 3 The maximum value for T_2 shall apply only to Tags in the reply or acknowledged states.

NOTE 4 If T2ext = 0, a Tag shall be allowed a tolerance of 4Tbit < T2(max) < 8 Tbit in determining whether T_2 had expired and therefore an "invalid command" was received. This allows the back calculation of elapsed T_2 time for command validation based on a measurement of the longer time from the end of the backscatter to the end of the command preamble where command start time is definitively established. If T2ext = 1, the Tag shall allow valid responses up to the INACT_T or Global Timeout expiration, whichever is selected by the Tag manufacturer.

NOTE 5 FT is the frequency tolerance specified for Manchester Tag BLFs of 640 kHz or less (currently 4%).

Tags and Interrogators that support optional higher frequency BLFs (see Manchester *Query_BAT* command as described in sub-clause 7.5.4.3.1.1) shall have their minimum and maximum T_1 times relaxed to not have to be shorter than that calculated for the maximum required BLF of 640 kHz. This prevents unreasonably small Tag data fetch times and Tag and Interrogator TR turn-around times for Tags and Interrogators supporting these higher BLFs.

7.5.2.1.11 Transmit mask

Interrogators that are claimed to operate according to this International Standard shall meet the local regulations for out-of-channel and out-of-band spurious radio-frequency emissions.

Interrogators that are claimed to operate in the Manchester battery assisted mode, in addition to meeting the local regulations, shall also meet the Manchester mode Transmit Mask specified in this International Standard:

Manchester mode transmit mask: For Interrogator transmissions centred at a frequency f_c, a 3.125/Tbit bandwidth R_{BW} also centred at f_c, an offset frequency f_o = 3.125/Tbit, and a 3.125/Tbit bandwidth S_{BW} centred at ($n \times f_o$) + f_c (integer n), the ratio of the integrated power $P()$ in S_{BW} to that in R_{BW} with the Interrogator transmitting random data shall not exceed the specified values:

$|n| = 1$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -30$ dB
$|n| = 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -60$ dB
$|n| > 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -65$ dB

Where $P()$ denotes the total integrated power in the 3.125/Tbit reference bandwidth. This mask is shown graphically in Figure 7, with dBch defined as dB referenced to the integrated power in the reference channel.

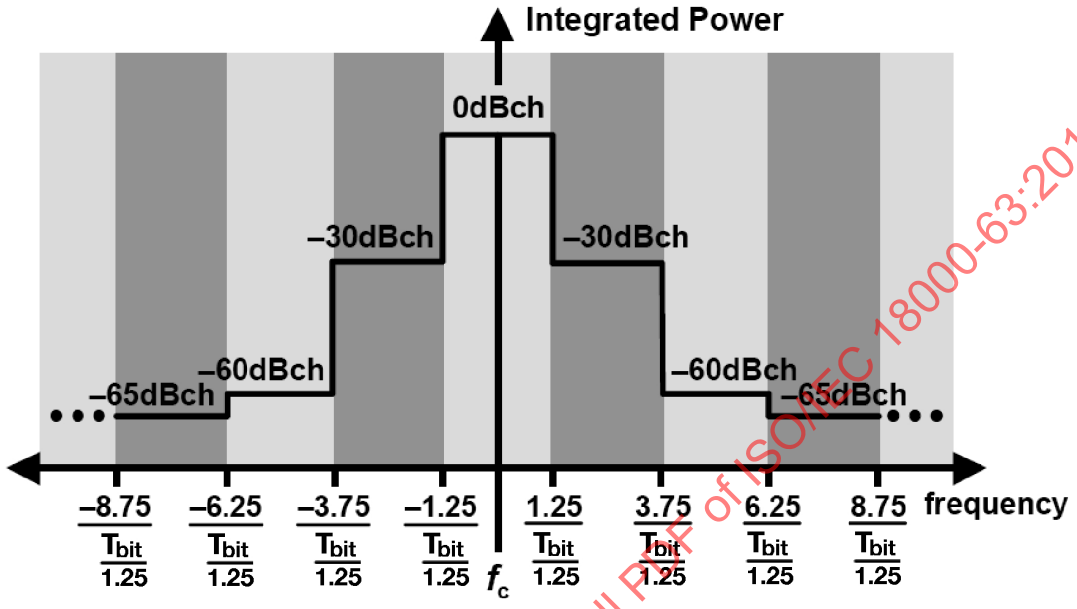


Figure 30 — Transmit mask for Manchester battery assisted mode

7.5.2.2 Tag-to-Interrogator (T=>R)

Tags shall communicate with Interrogators using the modulations described in 6.4.1.3 with extended values of M and with frequency accuracy of +/- 4% for BLF up to and including 640 kHz. For BLF greater than 640 kHz, the frequency accuracy shall be better than +/- 1.5%. The specific data rate is commanded by *Query_BAT* (see 7.5.4.3.1.1) using the values of M and BLF. Tags shall support all possible data rate combinations, whereas Interrogators may select the specific combinations of BLF and M to be implemented.

7.5.3 Manchester Activation

To extend battery life Type C battery assisted Manchester Tags remain in the low power **hibernate** state until receiving an *Activation* command containing a valid Activation Mask that matches all or a selected part of an "Activation Code" (AC) stored on the Tag. The use of two separate Activation Codes allows the ability to have a general purpose primary AC (typically static such as based on the Tag UII) and to have an additional application specific secondary AC (may contain for instance dynamically mapped Tag status information). The fraction of a Tag population chosen for activation may be controlled by means of partial AC matching. Non-selective activation of all battery assisted Tags within the range of the Interrogator is also supported by issuing a "Wildcard Activation".

In order to move from the low power **hibernate** state to the **battery ready** state an *Activation* command shall be sent at a fixed data rate of 8 kbit/s. This command consists of a specific Manchester battery assisted passive preamble that is distinguishable from the normal Manchester command preamble. Following the preamble, the *Activation* command may be a short format or long format command as defined in the following clauses.

Because battery assisted systems will coexist with passive systems, care should be taken to reduce the power drain of the battery assisted systems. Passive RFID Tags as defined in ISO/IEC 18000-6 Type C

receive their operating power from the Interrogator (via transmitted carrier power) which limits operating distances. The application requirements of battery assisted passive (BAP) RFID devices require distances sufficiently large to make the Interrogator an unusable power source. Additionally, BAP devices must co-exist in passive environments and care must be taken to manage power drain from the battery-enabled devices. If a BAP device continually responds to unwanted passive instructions (these being commands for "other" devices) battery power will be drained extremely quickly.

BAP Tags will typically use a very low power mode in the receiver to continuously monitor for an *Activation* command. The *Activation* command uses selection criteria in the Activation Mask to awaken only those Tags for which communication is necessary to preserve battery life. Tags which reside in the RF field for BAP mode communication may be selectively activated, accessed, then placed back into their **hibernate** (or lowest power) state, and the next set of Tags selectively activated. Activated Tags may be individually returned to the **hibernate** state using the *Next* command, or may be commanded as part of a large group of Tags using the *Deactivate_BAT* command.

The *Short Activation* command enables fast singulation cycles in applications that require a minimum of protocol overhead. The *Long Activation* command enables more control over which Tags are activated, particularly in large Tag and Interrogators population environments. Both *Short Activation* and *Long Activation* commands shall use the same preamble, followed by an Activation Control field which determines the specific format of the Activation Code.

The "**hibernate**" state is defined as that state where no Session flag timers are running and the **inventoried** flags are in state A. The "**stateful hibernate**" state is defined as that state where at least one session **inventoried** flag timer is running and the **inventoried** flag is in temporary state B. Normally, the temporary state *B* is interpreted to mean that the Tag was successfully accessed within the timer period using that session/**inventoried** flag, and that the Tag should not respond to *Activation* commands aimed at that **inventoried** flag until the timer expires. However, the Tag will respond to *Activation* commands for that session aimed at temporary state B. The purpose of such activations is normally to reprogram (refresh) the timer.

Tags will usually receive *Activation* commands while in a **hibernate** state. If a Tag that is already activated receives a valid activation preamble, the Tag may optionally ignore the command or it may clear the active **inventoried** flag, **SL** flag, and countdown timer, then transition to the **activation code check** state and continue processing the *Activation* command.

See sub-clause 7.5.3.5 for detailed state machine description of the Manchester Mode.

7.5.3.1 Activation Code

Battery assisted Type C Manchester Tags may support activation based on either a single or two separate Activation Codes for more flexibility at their manufacturer's discretion.

The Activation Code is a 96-bit value specified in part or whole by the Activation Mask during the activation sequence to move the Tag from the **hibernate** state to **battery ready**. The 96-bit Activation Code initially contains all zeroes. Once the AC is programmed and the Tag is placed into the **hibernate** state, the Tag will only respond, moving into the **battery ready** state when a valid activation sequence is received and the Activation Mask contained in the *Activation* command either matches the entire stored Activation Code or a selected sub-portion of the Activation Code. In case of Tags supporting two separate ACs the issued Activation Mask needs to match either the primary or the secondary AC (or both of them) in order to activate the Tag.

The Activation Codes can be programmed with any grouping organization which provides a selected application with desired levels of selectivity in the activation process. The Activation Codes may at the option of the user be the UII (if the UII is 96 bits), the UII plus additional data (if the UII is less than 96 bits), or part of the UII (if the UII is greater than 96 bits). They may also be a custom data string that may or may not include part of the UII.

The Minimum Mask Length (MML) contains seven bits that specify the minimum amount of bits which are required in the Activation Mask to match with the Activation Code during the activation sequence. If the

received value for the Activation Mask Length is less than the value stored in the MML, the incoming *Activation* command shall be ignored. If by application of the Activation Mask Length and Offset address, an overrun occurs (that being the 96th bit of the stored Activation Code) is compared and the received length counter is not exhausted, the remainder of the *Activation* command shall be ignored and the Tag shall wait in the **hibernate** state until a new *Activation* command is received. In case a Tag supports a primary and a secondary AC, a separate MML has to be implemented for each of the two ACs.

Both the MML registers and the Activation Codes shall be readable and writeable using the *OpRegister Read/Write* command in clause 7.5.4.4.1. The first 16 bit word shall contain the 7 bits of MML in the 7 lowest numbered bits of the word, MSB first, with the remaining bits of that word currently RFU. The next six 16 bit words (the Activation Code register) shall contain the AC value, which is always 96 bits. The MSB of the AC (bit index 95 starting from zero) shall be located in lowest addressed bit of the lowest addressed word of the six 16 bit words that make up the AC register. The LSB of the Activation Code (bit index zero) shall be stored in the highest address bit the highest addressed word of the six 16 bit words that make up the AC register. In the *Activation* command, the MSB of the Activation Mask that matches the AC or a section of the AC shall be transmitted MSB first.

Table 62 — Structure of Minimum Mask Length and Activation Code registers (stored in hidden registers described in 7.5.4.4.1)

MML register: first 16-bit word		AC register: next six 16-bit words					
7 bits MML value	9 bits RFU	AC (bits 95-80)	AC (bits 79-64)	AC (bits 63-48)	AC (bits 47-32)	AC (bits 31-16)	AC (bits 15-0)

The initial MML and AC values may be written during Tag manufacturing, or later by the user.

The default value for the Activation Code shall be 0.

7.5.3.2 Activation preamble

The activation preamble provides for Tag receiver dynamic range adjustment and AC-coupling training that is essential to good sensitivity in the Tag receiver. The training period is followed by a unique frame delimiter which, when matched exactly, alerts Tags to wake up parts of its circuitry required to receive and decode the complete *Activation* command and determine if the Tag has been selected for full communication. The frame delimiter is a pseudorandom sequence which cannot occur in error-free normal command data due to the use of selective 14/15 bit stuffing. Both *Short Activation* and *Long Activation* commands shall use the same preamble.

The preamble consists of three parts. All three parts shall be Manchester encoded at 8 kbit/s. The first part is 21 bits of a pseudorandom training sequence that enables receiver dynamic range adjustment, AC coupling training, and initial timing acquisition. The specific sequence is a 31 bit m-sequence truncated to 21 bits and is defined by the NRZ equivalent below and the Manchester equivalent in Figure 31.

Activation preamble 21 bit training sequence = 1111100 0110111 0101000

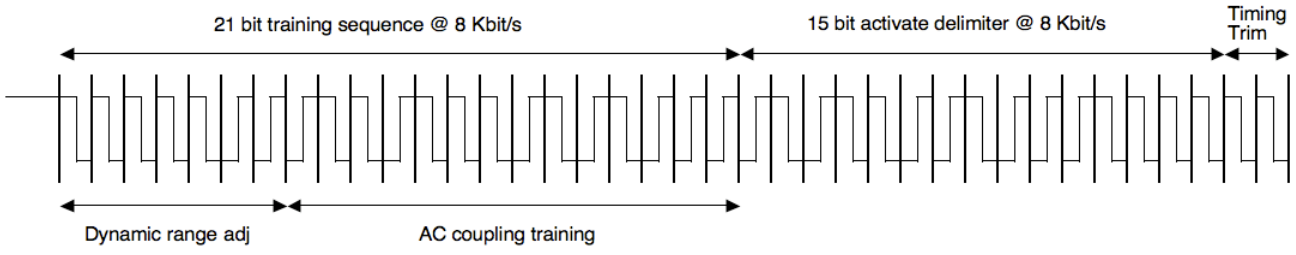


Figure 31 – Activation preamble

The second part is a 15 bit pseudorandom delimiter sequence that uniquely indicates an *Activation* command and the time framing of the command. Correct reception of the delimiter portion moves the Tag from the **hibernate** state into the **activation code check** state (see 7.5.3.5) immediately following the timing trim field. The activation delimiter sequence is a 15 bit m-sequence defined by the NRZ equivalent below and is the logical inverse of the normal delimiter.

Activation preamble 15 bit delimiter = 010110010001111 (followed by two "1" bits for Timing Trim)

The third part is a timing trim field consisting of two Manchester ones and is used to finalize timing synchronization after the frame delimiter is successfully matched and the Tag clock is turned on.

The first bit of the *Activation* command shall follow immediately after the last bit of the timing trim at the fixed data rate of 8 Kbit/s.

7.5.3.3 *Short Activation* command

The *Short Activation* command enables fast singulation cycles in applications that require a minimum of protocol overhead. The *Short Activation* command format is shown in Figure 32. When matched to an internal register, the received activation command moves the Tag from its **hibernate** state to an active state (**battery ready** state in the state machine). The *Short Activation* command is composed of four fields: (in order of reception at the Tag) Activation Control, Mask Length, Offset, and Activation Mask. Each is described in detail below.

Activation Control	Mask Length	Offset	Activation Mask (0-96 bits)
(12 bits)	(7 bits)	(If MaskLength = 0 or MaskLength = 96: 0 bits, Else: 7 bits)	

Figure 32 — Manchester *Short Activation* command format

The Activation Control field shall be used to configure the Tag receiver post activation behaviour. The bits in the Activation Control field are defined in Table 63. The RFU bits shall be set to all zeros for this standard version. If the Tag decodes any non-zero bits in this RFU field it shall ignore the *Activation* command. The Activation Version indicates the short or long format of the *Activation* command being used. The Forward Data Rate indicates the rate at which subsequent commands will be sent. The Sensitivity field shall command Tags to use high or low sensitivity for Tags that are capable of two modes of sensitivity. Low sensitivity Tags are defined as capable of receiving approximately -30 dBm up to +10 dBm, with a recommended upper limit of +15 dBm. High sensitivity Tags are defined as capable of receiving approximately -30 dBm and lower signal levels. These sensitivities refer to the 8 kbps data rate, and other data rates are understood to vary. Tags are not required to implement high sensitivity, and if they do not they shall ignore the command to switch to a high sensitivity mode.

In the Wildcard Activation case of Mask Length = 0 to wake up all Tags (described below), the Offset and Activation Mask are not needed and are not transmitted. If Mask Length is set to 96, then Offset is redundant and is not transmitted.

Table 63 — Manchester Activation Control field description

	RFU	Activation Version	Forward Data Rate	Sensitivity	RFU
# of bits	3	1	3	1	4
Description	RFU	0: short format 1: long format Must be set to 0 for Short Activation	000: RFU 001: 8 kbit/s 010: 16 kbit/s 011:32 kbit/s 100: 64 kbit/s 101: 128 kbit/s 110: RFU 111: RFU	0: low sensitivity (approx -30 dBm to +10 dBm) 1: high sensitivity (approximately -30 dBm and below)	

The Mask Length is a required field that contains seven bits. It may contain values from zero to 2^7-1(= 127). The Mask Length specifies the length of the transmitted Activation Mask from zero bits up to and including 96-bits. AC values greater than 96 are currently unused and reserved for future use (RFU). The length field is used in conjunction with the user defined Minimum Mask Length (MML). The MML controls the minimum controls the minimum length that an *Activation* command can use for the Mask Length. If a value of Mask Length is less than the MML value or greater than 96, the Tag will ignore the rest of the *Activation* command and will stay in the **hibernate** state.

The Offset is a conditionally included field containing 7 bits that indicate the number of bits to offset from the first bit (MSB) of the Activation Code for this comparison. The Offset is not transmitted if the Mask Length is either 0 or 96; the Offset is transmitted for Mask Length between 1 and 95 inclusive.

7.5.3.3.1 *Short Activation* command processing and Wildcard

Activation processing: The Tag follows the below sequence in processing *Short Activation* commands.

A Tag begins processing the *Activation* command by first training, acquiring a slicing reference, and acquiring frame synchronization. The Tag loads the Activation Control field consisting of 3 RFU bits, Activation Version (1 bit of value zero for Short Activation), Forward Data Rate field (3 bits), post activation Sensitivity field (1 bit), and 4 more RFU bits, as they are acquired.

If the Tag then receives a Mask Length value of zero and the Tag is programmed with MML = 0, then a successful Wildcard has occurred (further information below) and the Tag enters the active **battery ready** state and prepares to process inventory commands. All **inventoried** flags are set to *A* and the **SL** flag is deasserted, and all associated timers are cleared. In the Wildcard Activation case the Offset and Activation Mask are not needed and are not transmitted.

In the case of a non-Wildcard activation attempt (received Mask Length > 0), if the Mask Length is less than the MML value (which is set somewhere from 0 to 96), for all Activation Codes implemented by the Tag, the Tag rejects the activation and remains in **hibernate**. If Mask Length is set to 96, then Offset is redundant and is not transmitted. Values of Mask Length which are greater than 96 shall cause the Tag to discontinue activation processing and remain in the **hibernate** state.

If Mask Length is equal or greater than the MML, then the Tag continues processing the Activation command and receives the Offset field (assuming Offset is transmitted due to Mask Length being less than 96). If the combination of the Mask Length and the Offset (zero if not transmitted) result in an overflow (an ending value greater than the size of the 96-bit internal mask register), the Tag shall interpret this as an error, ignore the remainder of the *Activation* command, and remain in the **hibernate** state.

If the sum of Mask Length and Offset value is less than or equal to 96, the Tag proceeds to processing of the Activation Mask, which is transmitted MSB first. The Activation Mask shall be compared bit by bit against the stored Activation Codes or Activation Code segments by using the Mask Length as the number of bits to compare, and the Offset as the number of bits to offset from the first bit (MSB) of the Activation Codes for this comparison. The Tag shall discontinue the activation process if a mismatch in the bit by bit comparison is detected for the primary and the optional secondary Activation Code.

If the received Activation Mask fully matches one of the stored Activation Codes or indicated Activation Code segments, then the activation is successful and the Tag shall complete power-up operations and expect the beginning of an inventory round. All flags **inventoried** flags are set to *A* and the **SL** flag is deasserted, and all associated timers are cleared.

The Tag activation time T_A is the time required to finish the activation code check and fully power-up the Tag so that it is ready to receive a *Select* or *Query* command. The maximum Tag activation time is T_A = **2ms**.

Short Wildcard Activation and authorization:

If the MML register is set to zero, then the Mask Length within the *Activation* command may also be set to zero and be accepted by the Tag as "Wildcard Activation", i.e. a match for all devices in the field. This causes all receiving devices to initiate powering up into the active **battery ready** state. Short Wildcard Activation does not require any flag matching, whereas the later described Long Wildcard Activation may require flag matching. Note that if the MML for one of the supported Activation Codes is programmed to length greater than zero, this provides some security in the sense that the Tag is not authorized to accept Wildcard Activation, and the Interrogator must know at least the number of bits of Activation Code specified in the MML register in order to awaken the Tag.

In the Wildcard Activation case the Offset and Activation Mask are not needed and are not transmitted.

7.5.3.3.2 Short Activation post Activation behaviour

***Select* command behaviour for Short Activation:** The Tag is short activated in Interference Rejection = OFF (Promiscuous) mode (Session Locking not in effect) and all **inventoried** flag timers have been cleared. The Tag shall obey *Select* command reprogramming of the **SL** and **inventoried** flags. The *Select* command also moves any Tags out of the **battery ready** state back to the **battery ready** state. Upon return to **hibernate** the Tag shall set all **inventoried** flags to state *A* and the **SL** flag to the deasserted (**~SL**) state.

Manchester *Query_BAT* command behaviour for Short Activation: The Tag is short activated in Interference Rejection = OFF (Promiscuous) mode (Session Locking not in effect) and all **inventoried** flag timers have been cleared. The Tag shall respond to all *Query_BAT* commands as it normally does according to passive state machine operation (Queries take the Tag immediately into the **arbitrate** state using the session indicated in the *Query_BAT* command). Upon return to **hibernate** the Tag shall set all **inventoried** flags to state *A* and the **SL** flag to the deasserted (**~SL**) state.

7.5.3.4 *Long Activation* command

The *Long Activation* command enables more control over which Tags are activated, particularly in large Tag population environments. When correctly matched to Tag **inventoried** flag values, and to one of the Activation Codes or indicated Activation Code segments, the Tag shall advance from its **hibernate** state to the active state (**battery ready** state in the state machine). This will also occur for the Wildcard Activation case. The *Long Activation* command is composed of the seven fields shown in Figure 33.

Activation Control (12 bits)	Target (16 bits)	Mask Length (7 bits)	Offset (7 bits) (0 bits if Length=0)	Activation Mask (0 - 96 bits) (0 bits if Length=0)	Interrogator Info (17 bits)	CRC (16 bits)

Figure 33 — Long Manchester Activation command format

The Activation Control, Mask Length, Offset address, and Activation Mask fields shall be the same as defined in the Short Activation command.

In the Wildcard Activation case of Mask Length = 0 to wake up all Tags (described below), the Offset and Activation Mask are not needed and are not transmitted. If Mask Length is set to 96, then Offset is redundant and is not transmitted.

The CRC-16 shall be calculated by Interrogators over the first Activation Control bit to the last Interrogator Info bit. The use of the CRC-16 for Tags is optional.

The Target field is expanded as shown below.

Table 64 — Manchester Activation Target field description (Long Activation format only).

	Activation Tag Type Select field	**Session**	**Inventoried flag use**	**Inventoried flag target**	**Stateful hibernation timeout**
# of bits	8	2	1	1	4
Description	As specified in the Activation Tag Type Select field (Table 65)	00:S0 01:S1 10:S2 11:S3	0: Don't care for **inventoried** state 1: Do care for **inventoried** state	0: *A* 1: *B* So targeted if "Do Care" is specified.	0000: 0 s 0001: 0.25 s 0010: 0.5 s 0011: 1 s 0100: 2 s . . . 1111: 4,096 s

NOTE 1 The accuracy of the **hibernate** timer over the nominal temperature range shall be at least +/- 40%, and over the extended temperature range shall be at least +/- 50%.

NOTE 2 If **inventoried** flag use is set to 1: Do Care, then the Tag will upon activation exhibit "Session Locking" and generally reject commands for which the session ID does not match the "activating session" given by this table. For more information, see Annexes E and F. If Interrogator Locking is in effect, that will also set "Session Locking" into effect AFTER activation regardless of the state of **inventoried** flag use. But, even if Interrogator Locking is in effect, the **inventoried** flag use bit will control the Activation, meaning that if Don't Care is selected on **inventoried** flag use, then Tags will activate regardless of Session flag state.

NOTE 3 If **inventoried** flag use = Don't Care, then upon transition to the **battery ready** state the Tag clears all timers and makes the associated **inventoried** flags available to any command that uses them that the Tag receives.

Table 65 — Manchester Activation Tag Type Select Field (NOTES 1-5)

Interpretation (NOTE 1)	**Sensor Alarm**	**Full Function Sensor**	**Simple Sensor**	**RFU**	**RFU**	**RFU**	**Battery Assisted Passive** NOTE 4
1	1	1	1	1	1	1	1
0: Inclusive 1: Exclusive	0: No 1: Yes	0: No 1: Yes	0: No 1: Yes	0	0	0	0: No 1: Yes

NOTE 4 This bit dynamically switches the interpretation of the remainder of the Tag Type Select field between "Inclusive" (the Tag responds if it matches any selection marked "Yes") and "Exclusive" (the Tag responds only if it matches all selected criteria marked "Yes").

NOTE 5 In this table, "Yes" means that category of Tag is considered for inclusion in the activation. In the "Inclusive" case for a criteria listed as Yes the Tag will respond if it meets other activation criteria (such as AC). In the "Exclusive" case the Tag responds only if the Tag meets all criteria marked Yes. Logically the Tag responds to "Inclusive" if a logical OR function of all marked Yes criteria = 1. The Tag responds to "Exclusive" if a logical AND of all marked Yes criteria = 1.

NOTE 6 RFU bits shall be set to zero until defined. If the Interpretation field is "0, Inclusive" the Tag shall respond if it matches at least one selected criteria (the Tag ignores the RFUs because the implied logical OR function is still a one). If the interpretation field is "1, Exclusive", the Tag shall NOT respond if any RFU bits are non-zero (in that case the Tag does not know if it matches the implied AND function).

NOTE 7 This field means the Tag supports a Manchester receiver and a backscatter transmitter.

NOTE 8 The Manchester Activation Tag Type Select Field is identical to the Manchester *Query_BAT* Tag Type Select field.

The **stateful hibernate** timeout field determines the timeout period of the session **inventoried** flag timer specified in the session field.

The Interrogator Info field provides information as to Interrogator identity, whether the Tag is to reply to only the activating Interrogator or not after wake up, and the regulatory region of operation. The regulatory region field is an optional field for Interrogators and Tags and the definition is TBD.

Table 66 — Manchester Activate Interrogator Info Field Description

	Interrogator ID	**Interrogator Lock**	**Regulatory region**
# of bits	8	1	8
Description	Interrogator ID code	0: Tag allows any Interrogator to access 1: Tag only allows this Interrogator to access	Specifies a region in which the Tag operates. Definition TBD.

7.5.3.4.1 Long Activation processing and Wildcard

Long Activation processing: The Tag follows the sequence below in processing *Long Activation* commands.

A Tag begins processing the *Long Activation* command by first training, acquiring a slicing reference, and acquiring frame synchronization. The Tag loads the Activation Control field as it is acquired. The Tag then loads the activation Target field up to the flag use field. If the **inventoried** flag Care / Don't Care states as transmitted in the *Long Activation* command do not match those in the Tag, the Tag discontinues activation processing and remains in **hibernate** or **stateful hibernate**, awaiting the next *Activation* command.

With matching **inventoried** flag states, the Tag loads the rest of the activation Target field, including the Tag Type Select field and its set of criteria that are evaluated inclusively or exclusively as indicated. The Tag may discontinue processing the *Activation* command if it does not match the indicated criteria. If the Tag receives a Mask Length value of zero and the Tag is programmed with MML = 0 for all of its Activation Codes, then a wildcard attempt is in progress (further information below) and the Tag continues activation processing. In the Wildcard Activation case the Offset and Activation Mask are not needed and are not transmitted, so the next field received is the CRC-16. If the CRC-16 is passed (or the Tag does not use it), the Tag performs a final check on the Session flag state and if it still matches (has not timed out), it enters normal active mode and prepares to respond to normal inventory commands. Flag states remain unchanged and any timers in operation continue to run. If the CRC-16 is not passed the Tag remains in the **hibernate** state.

In the case of a non-Wildcard activation attempt with the **inventoried** flag state matching (received Mask Length > 0), if the Mask Length is less than the MML value (which is set somewhere from 0 to 96) for all Activation Codes implemented by the Tag, the Tag rejects the activation and remains in **hibernate**. If Mask Length is set to 96, then Offset is redundant and is not transmitted. Values of Mask Length which are greater than 96 shall cause the Tag to discontinue activation processing and remain in the **hibernate** state.

If Mask Length is equal or greater than the MML, the Tag continues processing the *Activation* command and receives the Offset field (assuming Offset is transmitted due to Mask Length being less than 96). If the combination of the Mask Length and the Offset (zero if not transmitted) results in an overflow (an ending value greater than the size of the 96-bit internal mask register), the Tag shall interpret this as an error, ignore the remainder of the *Activation* command, and remain in the **hibernate** or **stateful hibernate** state.

If the sum of Mask Length and Offset value is less than or equal to 96, the Tag starts processing of the Activation Mask, which is transmitted MSB first. The Activation Mask shall be compared bit by bit against the stored Activation Codes or Activation Code segments by using the Mask Length as the number of bits to compare, and the Offset as the number of bits to offset from the first bit (MSB) of the Activation Code for this comparison. The Tag shall discontinue the activation process and remain in **hibernate** if any mismatch in the bit by bit comparison is detected.

If the received Activation Mask fully matches one of the stored Activation Codes or indicated Activation Code segments, the Tag will continue to processing of the CRC-16. If the CRC-16 is passed (or the Tag does not use it), then the Tag performs a final check on the Session flag states and if it still match (has not timed out), then the activation is successful and the Tag shall complete power-up operations and expect the beginning of an inventory round. If the CRC-16 is failed, then the Tag discontinues activation processing and remains in **hibernate**, awaiting the next *Activation* command.

The Tag activation time T_A is the time required to finish the activation code check and fully power-up the Tag so that it is ready to receive a *Select* or *Query* command. The maximum Tag activation time is T_A = **2ms**.

Long Wildcard activation and authorization:

If the MML register is set to zero, then the Mask Length within the *Activation* command may also be set to zero and be accepted by the Tag as "Wildcard Activation", i.e. a match for all devices in the field for which the specified **inventoried** flag also match. If the Wildcard Activation is to apply to all Tags, then the Inventoried flag usage field is set to the "Don't Care" state. Note that if the MML for one of the supported Activation Code is programmed to length greater than zero, this provides some security in the sense that the Tag is not authorized to accept Wildcard Activation, and the Interrogator must know at least the number of bits of Activation Code specified in the MML register in order to awaken the Tag.

In the Wildcard Activation case the Offset and Activation Mask are not needed and are not transmitted. The CRC-16 (if used by the Tag) is still applicable to the Wildcard and is transmitted.

7.5.3.4.2 Long Activation post Activation behaviour

7.5.3.4.2.1 Long Activation flag timeout definition, behaviour, and usage

Manchester flag timeout definition:

Manchester **SL** and **inventoried** flag persistence shall have a different flag behaviour from passive and BAP PIE specified as follows. The definition of flag persistence shall change from beginning upon Interrogator transmissions dropping below the sensitivity of the Tag or the loss of decoding to beginning at the time the Tag transitions from normal active mode back to **hibernate**. These new flag persistences shall be referred to as “Manchester flag timeout and shall have accuracy as follows:

+/- 40% over the nominal temperature range of -25 to +40 °C.

+/- 50% over the extended temperature range of -40 to +65 °C.

Upon deactivation and return to the **hibernate** state, all flags without active timeouts in operation shall return to state A. Flags with active timeouts associated with other sessions that are in state *B* shall continue their timeouts (i.e. **stateful hibernate**), and upon timeout shall reset to state A. Flags associated with the current session (those specified in the last activation) that are in state *B* shall begin their timer operation and revert to state *A* upon timeout. Flags with active timeouts for which the state is already *A* upon entry to **hibernate** shall remain in *A* and need not continue or begin timer operation. Session flags with active timeouts and not used in a current activation session shall continue to countdown while the activation session is on-going.

When Manchester Tags are operating in BAP PIE mode, they may support the passive PIE compatible flag persistence times as given in Table 55. These are identical to the passive flag persistences of Table 17, except that they have specific maximums for the **S2**, **S3**, and **SL** flags and delay the beginning of flag persistence timeout by INACT_T. Alternatively, Manchester Tags that are operating in BAP PIE mode may support the passive flag persistences of Table 17. When a Manchester Tag is operating in Dead Battery Response mode, it may continue to support the passive compatible persistences of Table 55, or it may revert to the more relaxed persistences of Table 17.

Manchester Tags may but are not required in BAP PIE mode to use environmental validation with regards to INACT_T or Selective Global Timeout refresh. Alternatively, they may use signal strength above a threshold to refresh INACT_T or Selective Global Timeout. If a Manchester Tag prepares to participate in a BAP PIE interrogation round, it shall set its **inventoried** flags to *A* and deassert the select flag. Any Manchester Hibernation timers in operation continue to operate in the background. When the Tag returns to Manchester **hibernate** it shall set its **inventoried** flags according to the state of its **hibernate** timers and deassert the select flag.

When a Manchester Tag is operating in Dead Battery Response mode, it may continue to support the passive compatible persistence maximums of Table 55, or it may revert to the more relaxed persistences of Table 17.

Manchester flag behaviour and usage:

inventoried flag states in state **hibernate** (full or **stateful**) have the typical meaning *A* for “Default / Not Recently Inventoried” (timer not running) and *B* for “Temporary / Recently Inventoried” (timer running). System control in Normal Mode would normally also use *A* for “Not Inventoried” and *B* for “Inventoried”.

Each session/**inventoried** flag shall have its own individual timer for Manchester mode. To program different **inventoried** flags to different timeout values, it is necessary to use multiple *Activation* commands with Session Locking. In Normal Mode with Session Locking the **inventoried** flag of the activating session indicates inventory status in that session, and the other **inventoried** flags indicate their timer status.

With Session Locking not in effect, the Manchester Tag shall upon Activation clear any timers, set all **inventoried** flags to A, and set the **selected** flag to deasserted. Upon return to **hibernate** it shall set all **inventoried** flags to *A* and **selected** flag to deasserted.

With Session Locking in effect, a Tag that returns to **hibernate** via INACT_T or (Selective) Global Timeout shall clear the timer associated with activating session, and set that **inventoried** flag to A. This does not affect any other Hibernation mode timers that may be in operation.

With Session Locking in effect, Tags shall carry their inventory state from **hibernate** to Normal Mode. If returned to **hibernate** via the *Deactivate_BAT* or *Next* commands, they shall set **inventoried** flags to *B* if they have associated timers running and to *A* if no timer is currently running on that **inventoried** flag.

With Session Locking in effect, the Tag shall not respond to mismatched (indicated session does not match activation session) commands. Activation Session, as passed in the Activation command, is a variable that Tags implementing **hibernate** must keep track of in order to determine if there is a session match.

Thus, with Session Locking in effect the Tag is not able to participate in Query rounds with session mismatched Interrogators without first going to **hibernate** and being reactivated by the Interrogator using a different session. Only if Session Locking is not in effect can the Tag go into an inventory round with Interrogators using different sessions during one period of Normal Mode operation.

If a Manchester Tag receives a successful *Long Activation* command that reprograms an active timer, it shall clear the referenced timer, reset it with the newly received time value, and then proceed to the Normal Mode. Typically this operation is performed to “refresh” a timer that is about to expire, and the Tag or Tags would then be deactivated to return to **hibernate** mode, without being reinventoried, with the new timer value and **inventoried** flag in state B. The *Long Activation* command can only program or reprogram a single **inventoried** flag timer per command.

Examples of behaviour of these flags as the Tag crosses from **hibernate** to Normal Mode and back are given in Table 67 in Manchester sub-clause 7.5.3.4.2.1. This table applies to Manchester (Short and Long activation cases) in the row relative to Long Activation. See also Figure 34 for the Tag extended state machine, and sub-clause 7.5.4.3.7.1 for details of the Manchester *Deactivate_BAT* command and its effect on state machine state and flag values.

Examples of behaviour of these flags as the Tag crosses from **hibernate** to Normal Mode and back are given in Table 67 below. This table applies to Manchester (Short and Long activation cases). See also Figure 34 for the Tag extended state machine, and sub-clause 7.5.4.3.7.1 for details of the Manchester *Deactivate_BAT* command and its effect on state machine state and flag values.

Table 67 — Manchester inventory flag behaviour in Hibernation and Normal Modes

	State/command sequence	Next state	S0	S1	S2	S3
1	***Hibernation***		*A*	*A*	*A*	*A*
1.1	After Short Activation OR Long Activation with Don't care for **inventoried** state (no Session Locking)	**battery ready**	*A*	*A*	*A*	*A*
1.1.1	After *Query_BAT* on S1, successful singulation, *Next* -> **hibernate**	**hibernate**	*A*	*A*	*A*	*A*
1.1.2	After *Query_BAT* on S1, successful singulation, *QueryRep*	**battery ready**	*A*	*B*	*A*	*A*
1.1.2.1	After *Deactivate_BAT* (any session, but **SL** and **inventoried** flag	**hibernate**	*A*	*A*	*A*	*A*

	State/command sequence	Next state	S0	S1	S2	S3
	state pointed do match)					
1.1.2.2	After *Deactivate_BAT* (session = S1, **SL** match, but **inventoried** flag of S1 Target = *A* does not match)	**battery ready**	*A*	*A*	*A*	*A*
1.1.2.2	After INACT_T or (Selective) Global Timeout	**hibernate** (including brief stay in **stateful hibernate** to check timers expired)	*A*	*A*	*A*	*A*
1.1.3	After INACT_T or (Selective) Global Timeout (no successful singulation)	**hibernate**	*A*	*A*	*A*	*A*
1.2	After Long Activation with Do care for **inventoried** state S1=A (Session Locking)	**battery ready**	*A*	*A*	*A*	*A*
1.2.1	After *Query_BAT* on S1, successful singulation, *Next* -> **hibernate**	**stateful hibernate**	*A*	*B*	*A*	*A*
1.2.2	After *Query_BAT* on S1, successful singulation, *QueryRep*	**battery ready**	*A*	*B*	*A*	*A*
1.2.2.1	After *Deactivate_BAT*	**stateful hibernate**	*A*	*B*	*A*	*A*
1.2.2.2	After INACT_T or (Selective) Global Timeout	**stateful hibernate**	*A*	*A*	*A*	*A*
1.2.3	After INACT_T or (Selective) Global Timeout (no successful singulation)	**hibernate**	*A*	*A*	*A*	*A*
2	**Stateful hibernation**, S0 & S3 timers running		*B*	*A*	*A*	*B*
2.1	After Short Activation (no Session Locking)	**battery ready**	*A*	*A*	*A*	*A*
2.1.1	After *Query_BAT* on S1, successful singulation, *Next* -> **hibernate**	**hibernate**	*A*	*A*	*A*	*A*
2.1.2	After *Query_BAT* on S1, successful singulation, *QueryRep*	**battery ready**	*A*	*B*	*A*	*A*
2.1.2.1	After *Deactivate_BAT*	**hibernate**	*A*	*A*	*A*	*A*
2.1.2.2	After INACT_T or (Selective) Global Timeout	**hibernate**	*A*	*A*	*A*	*A*
2.1.3	After INACT_T or (Selective) Global Timeout (no successful singulation)	**hibernate**	*A*	*A*	*A*	*A*

	State/command sequence	Next state	S0	S1	S2	S3
2.2	After Long Activation with Don't care for **inventoried** state (no Session Locking)	**battery ready**	*A*	*A*	*A*	*A*
2.2.1	After *Query_BAT* on S1, successful singulation, *Next* -> **hibernate**	**hibernate**	*A*	*A*	*A*	*A*
2.2.2	After *Query_BAT* on S1, successful singulation, *QueryRep*	**battery ready**	*A*	*B*	*A*	*A*
2.2.2.1	After *Deactivate_BAT* (matching session, **SL**, and **inventoried** flag state)	**hibernate**	*A*	*A*	*A*	*A*
2.2.2.2	After *Deactivate_BAT* (matching session, **SL**, but non-matching **inventoried** flag state)	**battery ready**	*A*	*A*	*A*	*A*
2.2.2.3	After INACT_T or (Selective) Global Timeout	**hibernate**	*A*	*A*	*A*	*A*
2.2.3	After INACT_T or (Selective) Global Timeout (no successful singulation)	**hibernate**	*A*	*A*	*A*	*A*
2.3	After Long Activation with Do care for **inventoried** state S1=A (Session Locking in effect)	**battery ready**	*B*	*A*	*A*	*B*
2.3.1	After *Query_BAT* on S1, successful singulation, *Next* -> **hibernate**	**stateful hibernate**	*B*	*B*	*A*	*B*
2.3.2	After *Query_BAT* on S1, successful singulation, *QueryRep*	**battery ready**	*B*	*B*	*A*	*B*
2.3.2.1	After *Deactivate_BAT* (matching session, **SL**, and **inventoried** flag state)	**stateful hibernate**	*B*	*B*	*A*	*B*
2.3.2.2	After INACT_T or (Selective) Global Timeout	**hibernate**	*B*	*A*	*A*	*B*
2.3.3	After INACT_T or (Selective) Global Timeout (no successful singulation)	**stateful hibernate**	*B*	*A*	*A*	*B*
2.4	After Long Activation with Do care for **inventoried** state S3=B (timer refresh) (Session Locking in effect)	**battery ready**	*B*	*A*	*A*	*B*
2.4.1	After *Deactivate_BAT* (matching session, **SL**, and **inventoried** flag state)	**stateful hibernate**	*B*	*A*	*A*	*B*
2.4.1.1	After S0 timer expires	**stateful hibernate**	*A*	*A*	*A*	*B*
2.4.1.2	After S0 and S3 timer expires	**hibernate**	*A*	*A*	*A*	*A*

	State/command sequence	Next state	S0	S1	S2	S3
2.4.2	After *Deactivate_BAT* (matching session, but non-matching **SL**, and matching **inventoried** flag state) NOTE: The **SL** flag is not affected	**battery ready**	*B*	*A*	*A*	*B*
2.4.3	After *Deactivate_BAT* (matching session, non-matching **inventoried** flag state, both **SL** states) NOTE 1 The S3 timer is also cleared NOTE 2 The **SL** flag is not affected	**battery ready**	*B*	*A*	*A*	*A*

7.5.3.4.2.2 Long Activation post Activation command behaviour

***Select* command behaviour without Session Locking and without Interrogator to Tag Locking:** In the case where a Tag is activated with "Don't Care" on the **inventoried** flag use bit of Table 64 (Promiscuous Mode), all **inventoried** flag timers have been cleared and the Tag shall obey *Select* command reprogramming of the **SL** and **inventoried** flags. The *Select* command also moves any Tags out of the **battery ready** state back to the **battery ready** state. Upon return to **hibernate** the Tag shall set all **inventoried** flags to state *A* and the **SL** flag to the deasserted (**~SL**) state.

Manchester *Query_BAT* command behaviour without Session Locking and without Reader to Tag Locking: In the case where a Tag is activated with "Don't Care" on the **inventoried** flag use bit of Table 64 (Promiscuous Mode), all **inventoried** flag timers have been cleared and the Tag shall respond to all *Query* commands as it normally does according to passive state machine operation (Queries take the Tag immediately into the **arbitrate** state using the session indicated in the *Query* command). Upon return to **hibernate** the Tag shall set all **inventoried** flags to state *A* and the **SL** flag to the deasserted (~**SL**) state.

From a functional point of view, it is not necessary that Session Locking be maintained when Interrogator to Tag locking is used. But, for design simplicity it is assumed that if Interrogator to Tag locking is in effect, then Session locking shall also be in effect.

***Select* command behaviour with Session Locking:** In the case where a Tag is activated with "Do Care" on the **inventoried** flag use bit of Table 64 (Interference Rejection Mode ON), the Tag shall obey *Select* command **SL** or **inventoried** flag programming only if the Target of the *Select* command is **SL** or the session that matches that of the Activation Session. If Interrogator Locking is in effect, the Tag shall also verify that the Interrogator ID field is included in the *Select* command and that the ID value matches that of the last *Activation* command before processing the *Select* command. *Select* commands with non-matching Session ID's or Interrogator ID's shall be ignored. Upon return to **hibernate** the Tag shall set the activation **inventoried** flag to *B* and commence the activation programmed timeout, and set the **SL** flag to the deasserted (**~SL**) state. Other session timers will continue their operation as previously programmed (flag state *B* while running, and flag state *A* after timeout).

Manchester *Query_BAT* command behaviour with Session Locking: In the case where a Tag is activated with "Do Care" on the **inventoried** flag use bit of Table 64 (Interference Rejection Mode ON), the Tag has only programmed the timer associated with the Activation Session. The Tag shall respond to *Query_BAT* commands as it normally does according to passive state machine operation (Queries take the Tag immediately into the **arbitrate** state using the **inventoried** flag indicated in the *Query* command) only if the *Query_BAT* command session matches that of the Activation Session, and for Interrogator Locking in effect, only if the Interrogator ID field of the *Query_BAT* command matches that of the last *Activation* command. Non-matching *Query_BAT* commands are ignored. Upon return to **hibernate** the Tag shall set the Activation Session **inventoried** flag to *B* and commence the activation programmed timeout, and set the **SL** flag to the deasserted (**~SL**) state. Other session timers will continue their operation as previously programmed (flag state *B* while running, and flag state *A* after timeout).

7.5.3.5 Manchester extended Tag state machine

Tag manufacturers may define their own state control to minimize power consumption when using *Activation*. A typical state flow for Tags implementing activation mechanism is shown in Figure 34. This suggested state diagram is useful for standardization description and to define Tag behaviour when viewed externally. However, the actual Tag behaviour requirements are for the Tag to respond as if it is executing the states and functions as shown. The Tag activation time T_A is the time required to fully power-up the Tag after the activation code check so that it is ready to receive a *Select* or *Query_BAT* command. The maximum Tag activation time is **T_A = 2 ms**.

The states of this figure are defined as follows:

hibernate state: A low power low data rate state that allows for battery life extension as compared to the Normal Mode. In the **hibernate** state the inventory flag timers are expired. The Tag is listening for valid *Activation* commands while in this state. In this state **inventoried** flags are set to *A* and **SL** is deasserted. Note that Validated INACT_T is required for Manchester, and that it escapes the "Interference Trap" created by an interfering source and will bring a Tag back to **hibernate** in the absence of valid Manchester commands or preambles. The optional Global Timeout may also be used to escape an interference trap created by valid Manchester commands that are not applicable to a particular Tag, such as a Tag whose matching criteria for a "valid" Manchester command is not very selective.

stateful hibernate: This state is similar to **hibernate** with the extension that inventory timers are either in operation or the Tag is very briefly checking that they are in operation. When this check is complete the Tag goes to the **hibernate** state. While in **stateful hibernate** the Tag is still listening for valid *Activation* commands, and (unless there only briefly for timer state check) at least one inventory timer is operating.

After activation and inventory, Tags are typically deactivated by the *Deactivate_BAT* or *Next* commands in order to return to the **hibernate** or **stateful hibernate** state. If the Tag does not for whatever reason receive one of these two commands, it shall via either the INACT_T or Global Timeout timers return to the **hibernate** mode (see 7.5.3.6).

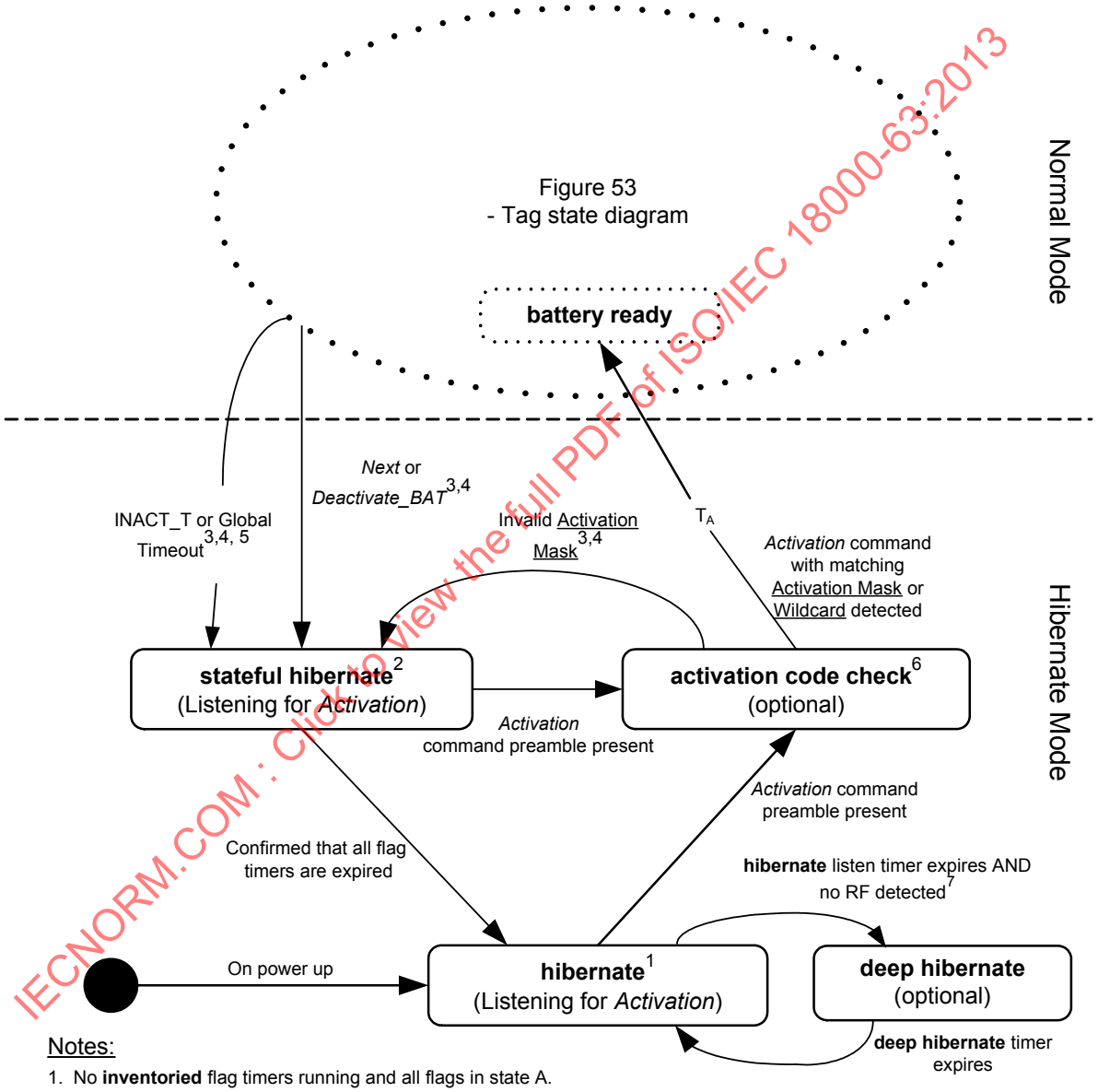
Activation code check state: In this optional state the Tag has received a valid Manchester *Activation* preamble and is proceeding to process the Activation Code. This operation can be conducted also in **hibernate** or **stateful hibernate**, but a separate state is defined to acknowledge that the Tag may be in the process of conduction power-up operations that it is not normally performing in **hibernate**. At the conclusion of successful Activation Code processing the Tag progresses to the **battery ready** state of the Normal Mode. If the AC check fails, the Tag may progress to **stateful hibernate** to continue timer operations or for brief timer state check. Or, if the Tag conducts timer check in the **activation code check** state it may progress straight to **hibernate**.

Deep hibernation state: This is an optional state used for optional duty cycling of the **hibernate** listen action to further extend battery life. The Tag is not listening for *Activation* commands in the **deep hibernate** state, but at manufacturer option it may be listening for RF signal presence to bring it back to **hibernate** earlier than its timer dictates. The duty cycle between **hibernate** and **deep hibernate** may be fixed or user programmable via TCRS. If the Tag supports this functionality, then it keeps internal timers for **hibernate** listen time and **deep hibernate** time. The Tag does not progress from **stateful hibernate** to **deep hibernate** since it has been recently accessed if in **stateful hibernate**. The Tag progresses from **hibernate** to **deep hibernate** when its **listen** Timer has expired and it has not detected either RF or valid *Activation* commands while in **hibernate**. Tags that implement the **deep hibernate** state are required to extend their **hibernate** listen timer to at least 4s when the RF level is above a manufacturer defined threshold.

It is at the Tag manufacturer's option if duty cycling of hibernation is supported, and if so, if it is fixed or programmable. If the **hibernate**-**deep hibernate** duty cycle is fixed and if TCRS is supported, the Tag can report its timing parameters via TCRS. If fixed, the **deep hibernate** period shall not exceed 4s. If programmable, the initial **deep hibernate** period programmed by the manufacturer shall not exceed 4s, though it may be programmed deliberately longer by the user. If programmable, the limits of programmability are as described in sub-clause 7.6.3. on TCRS. The manufacturer may set duty cycle limits smaller than the maximum ranges described, and these limits are also reportable via TCRS.

It is also at the Tag manufacturers' option if either current or recently detected RF or validated *Activation* commands alter the **hibernate** listen time and the **deep hibernate** time. It may be desirable to allow Tag design to go to shorter duty cycles in such situations where *Activation* commands are more likely, and to go to longer duty cycles when such commands are unlikely. If the Tag supports this capability, it is reported and controlled through TCRS.

It is also at the Tag manufacturers' option if duty cycle, either fixed or programmable, may be altered by the Tag in response to a low battery state in order to extend battery life. Parameters for reporting this capability, and allowing user authorization of its use, are provided in TCRS.



Notes:

1. No **inventoried** flag timers running and all flags in state A.
2. At least one **inventoried** flag timer running and **inventoried** flag in temporary state B.
3. **Stateful hibernate** means that at least one timer is running or that the tag is briefly checking if any are running.
4. The tag manufacturer has the option of checking if timers are running in the prior state and going straight to h**ibernate** if none are, or first transitioning to s**tateful hibernate** to check if timers are running, and then going to **hibernate**.
5. The INACT_T and (Selective) Global Timeout clear all timers and the inventory flags in **hibernate** shall be set to A.
6. **Activation code check** is an optional state that may be used if power-up sequencing begins upon successful preamble detection and not upon completed Activation Code validation.
7. The tag may extend its h**ibernate** listen time if RF is detected, according to manufacturer choice. The tag may also extend hibernate listen time and/or shorten s**leep** time according in adaptation to recent activity such as *Activation* commands detected within a Last Activation Timer period (Last_Act_T).

Figure 34 — Extended state machine of BAP Tag implementing Manchester mode

7.5.3.6 Manchester mode Deactivation

After an Interrogator identified a battery assisted Tag and no longer needs to access it, the Interrogator shall use the *Next* or *Deactivate_BAT* command to put the Tag back into the **hibernate** state or the **stateful hibernate** state with a shorter delay. **Hibernate** commands *Next* and *Deactivate_BAT* do not flip **inventoried** state, they set state to *A* or *B* as a function of Activation Session and timer state, that is, upon return to **hibernate** the flag will be in state *A* if Session Locking is not in effect or there is no timer operation, and if Session Locking applies then in state *B* until the expiration of any programmed timer function for that flag.

There are two methods to ensure that a battery assisted Tag falls into the **hibernate** state automatically if the Tag misses the *Deactivate_BAT* or *Next* commands issued by an inventorying Interrogator. These are:

1. A Tag may automatically fall back into the **hibernate** state after a period of inactivity greater than the manufacturer defined Inactivity Threshold INACT_T, which is allowed to be 50 ms or greater. Inactivity means that a Tag does not receive valid selection commands, commands that cause it to participate in an inventory round, commands that control an inventory round the Tag participates in, or commands that directly address the Tag (Tag access commands). The Tag manufacturer may use various tests as to whether a command is valid, as function of Tag state. For example, in the **reply**, **acknowledged**, **open**, and **secured** states the Interrogator uses a Tag generated RN16. The Tag may then require the use of that RN16 as a validator to consider the command valid for the purpose of refreshing INACT_T. See sub-clause 7.3.2.2 for more information on the INACT_T timer function.

2. Tags may implement a Global Timeout in one of two forms, the standard Global Timeout and the Selective Global Timeout. The term (Selective) Global Timeout refers to either of Global Timeout or Selective Global Timeout. The standard Global Timeout causes Tags to fall back into the **hibernate** state a certain time after they were activated. This timeout, if implemented, will force the Tag back to **hibernate** despite the fact that it may be involved in a legitimate inventory round. In order to reduce the odds of an intended inventory round from being interrupted, this Global Timeout if fixed shall be 4 seconds or more (4 s minimum at 40% accuracy over nominal temperature range of -25 to + 40 °C, and 50% accuracy over extended temperature range of -40 to +65 °C). The Selective Global Timeout may refresh its timer upon reception of selected valid Manchester commands, and it may optionally use Session Locking match or Interrogator Locking match. The Selective Global Timeout, if fixed, shall be 2 seconds or more. See sub-clause 7.3.2.3 for more information on the (Selective) Global Timeout function.

Manchester Tags shall implement at least one of INACT_T, or (Selective) Global Timeout. The Tag manufacturer also has the option of allowing these timeouts to be user programmable via the TCRS system of sub-clause 7.6.3. Programmable Tags may allow shorter limits on (Selective) Global Timeout than listed above.

Expiration of either of these timers in a Manchester Tag shall force the Tag back to **hibernate** with the timer from the activating session cleared, the **inventoried** flag state of the activating session set to A, and **selected** flag (**SL**) deasserted.

When a Manchester Tag is operating in BAP PIE mode it shall apply the delay specified in INACT_T to the start of persistence flag timeout as specified in Table 55. It may support the accurately specified maximums of this table, or it may use the more relaxed persistence maximums of Table 17.

7.5.4 Commands summary

7.5.4.1 Commands summary table

Table 68 lists the normal Manchester “battery-assist” commands as compared to ISO/IEC 18000-6 Type C. For Manchester commands that are different from the passive mode PIE commands in Clause 6, the differences are detailed in the clauses below that define the new or modified commands. For commands that are functionally unchanged other than the preamble and bit stuffing of the command data as defined in section 7.5.2 and 7.5.3, Interrogators and Tags shall comply with the requirements in the appropriate sub-clauses in Clause 6.

The *Activation* command is not included in this section as it is defined in clause 7.5.3 with different physical and functional characteristics from normal commands.

Table 68 — Manchester version commands

Command	Binary command code	Command code length	Manchester different from PIE?	Mandatory for Manchester?
QueryRep	00	2	Yes	Yes
ACK	01	2	No	Yes
Query	1000	4	n/a	Not used
QueryAdjust	1001	4	Yes	Yes
Select	1010	4	Yes	Yes
RFU	1011	4	n/a	n/a
NAK	1100 0000	8	Yes	Yes
ReqRN	1100 0001	8	No	Yes
Read	1100 0010	8	No	Yes
Write	1100 0011	8	No	Yes
Kill	1100 0100	8	No	Yes
Lock	1100 0101	8	No	Yes
Access	1100 0110	8	No	No
BlockWrite	1100 0111	8	No	No
BlockErase	1100 1000	8	No	No
BlockPermalock	1100 1001	8	No	No
Deactivate_BAT	1100 1010	8	n/a	Yes
Next	1100 1011	8	n/a	Yes
Query_BAT	1100 1100	8	n/a	Yes
Broadcast ID	1100 1101	8	n/a	No
Multirate_Reset	1100 1110	8	n/a	Yes
BAP PIE Flex Query	1100 1111	8	n/a	n/a
OpRegister Read/Write	1101 0000	8	n/a	Yes
BroadCastSync	1101 0001	8	n/a	No
Reserved for future use	1101 0010 … 1101 1111	8	-	-
Reserved for custom commands (vendor-specific)	1110 0000 0000 0000 … 1110 0000 1111 1111	16	-	-
Reserved for proprietary commands	1110 0001 0000 0000 … 1110 0001 1111 1111	16	-	-
Extended commands (reserved for future use)	1110 0010 0000 0000 … 1110 1111 1111 1111	16	-	-

7.5.4.2 *Select* commands

7.5.4.2.1 Manchester *Select* command

The Manchester version of the *Select* command inserts the 8 bit Short Interrogator ID field just before the CRC16. This field shall be included in the CRC calculation.

Table 69 — Manchester *Select* command

	CMD ID	Target	Action	Mem Bank	Pointer	Length	Mask	Truncate	Short Interrogator ID	CRC 16
# of bits	4	3	3	2	EBV	8	Varia-ble	1	8	16
Description	1010	000: S0 001: S1 010: S2 011: S3 100: **SL** 101: RFU 110: RFU 111: RFU	See Table 21	00: RFU 01: UII 10: TID 11: User	Starting Mask address	Mask Length	Mask Value	0: Disable 1: Enable	Tag checks if interrogator locking is in effect.	

Tags shall not reply to a *Select* command.

7.5.4.3 Inventory commands

7.5.4.3.1 Manchester *Query_Bat*

7.5.4.3.1.1 Manchester *Query_Bat* command

Tags and Interrogators shall implement the Manchester *Query_Bat* command shown in Table 70 and Table 71. *Query_BAT* initiates and specifies an inventory round for battery assisted Tags.

The basic function of the "Tag Type Select Field" field is for selecting the categories of Tags to be included in the interrogation round without requiring a separate *Select* command. These categories include the current Battery Assisted Passive design, Tags featuring various kinds of sensors and sensor conditions (such as a Sensor Alarm that may need expedited handling), and RFU bits for future expansion. A Tag shall ignore the RFU bits within the Tag Type Select regardless of their values and only participate in a singulation round based on the currently defined selection bits.

The Tag shall send the reverse link backscatter signal on subcarrier frequency selected by the BLF field. The maximum required BLF shall be 640 kHz.

The field SS Response enables or disables Simple Sensor response in a Tag implementing Simple Sensor data for a given interrogation round. For Tags that do not support Simple Sensor data, the Tag shall ignore the value of SS Response.

Table 70 — Manchester *Query_BAT* command (see NOTES 1 to 4)

	CMD	**Query Tag Type Select field**	**M**	**TRext**	**Sel**	**Session**	**Target**	**Q**	**SS Resp**	**BLF** See NOTE 2	**T2ext**	**CW power Delta over forward power**	**Short Interrogator ID**	**CRC-5**
# of bits	8	8	4	1	2	2	1	4	1	4	1	2	8	5
Description	1100 1100	NOTE 1	0000: M=1 0001: M=2 0010: M=4 0011: M=8 0100: M=16 0101: M=32 0110: M=64 0111: M=128 1000: M=256 1001 to 1111: RFU	0: No pilot tone 1: Use pilot tone	00: All 01: All 10:~**SL** 11: **SL**	00: S0 01: S1 10: S2 11: S3	0: A 1: B	0-15	0: No 1: Yes	Required values 0000: 25.2632 kHz 0001: 40 kHz 0010: 48 kHz 0011: 64 kHz 0100: 80 kHz 0101: 96kHz 0110: 120 kHz 0111: 160 kHz 1000: 192 kHz 1001: 240 kHz 1010: 320 kHz 1011: 384 kHz 1100: 480 kHz 1101: 640 kHz Optional values: 1110: 960 kHz 1111: 1920 kHz	0: max 4Tbit 1: INACT_T or Global Timeout	11: +30 dB 10: +20 dB 01: +10 dB 00: 0 dB NOTE 3 NOTE 4	Tag checks if Interrogator locking is in effect.	

NOTE 1 See Table 71 for Tag Type Select field options. Tags that match the chosen criteria will enter the Query rounds, while others will remain in the **battery ready** state. If Session Locking is in effect, then there must be a match between the Session specified in the *Query_BAT* command and the activating session for the Tag to respond to this command. See 7.5.3.4.2.2.

NOTE 2 Required BLF accuracy is 4% for BLF <= 640 kHz, and 1.5% for BLF > 640 kHz, over the temperature range that the Tag supports.

NOTE 3 This field indicates optional offset between Interrogator forward link transmit peak power and reverse backscatter supporting carrier power. The field shows that reverse link caririer power may be equal to or greater than forward Interrogator power in steps of 10 dB. Lower Interrogator forward power limits system interference (particularly Interrogator on Interrogator), while higher reverse link supporting carrier power relieves the reverse link limit of RFID systems with highly sensitive BAP Tags. See Annex P for further information.

NOTE 4 Tolerance on the variable Interrogator transmit power levels is not specified, but is recommended as +/- 4 dB.

Table 71 — Manchester *Query_BAT* Tag Type Select field (NOTES 1 to 5)

Interpretation (NOTE 1)	**Sensor Alarm**	**Full Function Sensor**	**Simple Sensor**	**RFU**	**RFU**	**RFU**	**Battery Assisted Passive** NOTE 4
1	1	1	1	1	1	1	1
0: Inclusive 1: Exclusive	0: No 1: Yes	0: No 1: Yes	0: No 1: Yes	0	0	0	0: No 1: Yes

NOTE 5 This bit dynamically switches the interpretation of the remainder of the Tag Type Select field between “Inclusive” (the Tag responds if it matches any selection marked “Yes”) and “Exclusive” (the Tag responds only if it matches all selected criteria marked “Yes”).

NOTE 6 In this table, “Yes” means that category of Tag is considered for inclusion in the Query round. In the “Inclusive” case for a criteria listed as Yes the Tag will respond if it meets other criteria in the *Query* command. In the “Exclusive” case the Tag responds only if the Tag meets all criteria marked Yes as well as other Query criteria. Logically the Tag responds to “Inclusive” if a logical OR function of all marked Yes criteria = 1. The Tag responds to “Exclusive” if a logical AND of all marked Yes criteria = 1.

NOTE 7 RFU bits shall be set to zero until defined. If the Interpretation field is “0, Inclusive” the Tag shall respond if it matches at least one selected criteria (the Tag ignores the RFUs because the implied logical OR function is still a one). If the Interpretation field is “1, Exclusive”, the Tag shall NOT respond if any RFU bits are non-zero (in that case the Tag does not know if it matches the implied AND function).

NOTE 8 This field means the Tag only supports a Manchester receiver and a backscatter transmitter.

NOTE 9 The Manchester *Query_BAT* Tag Type Select field is identical to the Manchester activation Tag Type Select Field.

7.5.4.3.1.2 Tag response to a Manchester *Query_BAT* command

A Tag that generates a slot counter value of 0 replies to Manchester *Query_BAT* command with an RN16. This response is similar to the Query response in Clause 6 except for the modifications to BLF, M, and frequency accuracy.

Table 72 — Tag response to a Manchester *Query_BAT* command

	Response
# of bits	16
Description	RN16

7.5.4.3.2 Manchester *QueryAdjust*

7.5.4.3.2.1 Manchester *QueryAdjust* command

The Manchester version of the *QueryAdjust* command inserts the 8 bit Short Interrogator ID field just before the CRC16. This field shall be included in the CRC calculation. Unlike the PIE *QueryAdjust*, which steps the slot counter parameter Q up and down, the Manchester version can immediately reset the Q to any allowed value.

Table 73 — Manchester *QueryAdjust* command

	CMD ID	Session	Q Value	Short Interrogator ID	CRC-16
# of bits	4	2	4	8	16
Description	1001	00: S0 01: S1 10: S2 11: S3	Q value from 0 to 15	Tag checks if Interrogator locking is in effect.	

7.5.4.3.2.2 Tag response to a Manchester *QueryAdjust* command

A Tag that generated a slot counter value of 0 replies to Manchester *QueryAdjust* command with RN16. This response is similar to the Query response in Clause 6 except for the modifications to BLF, M, and frequency accuracy.

Table 74 — Tag response to a Manchester *QueryAdjust* command

	RN16
# of bits	16
Description	RN16

7.5.4.3.3 Manchester *QueryRep*

7.5.4.3.3.1 Manchester *QueryRep* command

The Manchester version of the *QueryRep* command appends the 8 bit Short Interrogator ID to the end of the command only if Interrogator locking is in effect as indicated in the *Activation* command.

Table 75 — Manchester *QueryRep* command

	CMD ID	Session	Short Interrogator ID
# of bits	2	2	8
Description	00	00: S0 01: S1 10: S2 11: S3	Included only if Interrogator locking is in effect.

7.5.4.3.3.2 Tag response to a Manchester *QueryRep* command

A Tag that generated a slot counter value of 0 replies to Manchester *QueryRep* command with RN16. This response is similar to the Query response in Clause 6 except for the modifications to BLF, M, and frequency accuracy.

Table 76 — Tag response to a Manchester *QueryRep* command

	RN16
# of bits	16
Description	RN16

7.5.4.3.4 Manchester *ACK*

The Manchester *ACK* command has a modified behaviour from the equivalent command in clause 6.4.2.11.2.4. The modification makes Tags with greater sensitivity immune to *ACK* commands intended for other Tags sent by enemy Interrogators (see Annex B and Annex C for details). It works by preventing the change in present state caused by successfully decoding an *ACK* command with invalid RN16 or RN16_handle.

7.5.4.3.4.1 Manchester *ACK* command

The Manchester *ACK* command structure is shown in Table 77.

Table 77 — Manchester *ACK* command

	Command	RN
# of bits	2	16
Description	01	Echoed RN16 or handle

Table 78 — Tag reply to a successful *ACK* command

	Response
# of bits	See C.4.4
description	See C.4.4

7.5.4.3.5 Manchester *NAK*

The Manchester *NAK* command includes two parameters, session, and short Interrogator ID, though the Interrogator ID is only transmitted if Interrogator locking is in effect. These two parameters provide different levels of interference rejection. When Interrogator locking is not in effect, the session parameter is used to avoid accidental return to arbitrate state when the session parameter does not match. A higher degree of protection is achieved by using Interrogator locking. Any Tag that successfully decodes a *NAK* command with the correct parameters shall transition to arbitrate state (except while in **battery ready** or **killed** states). Otherwise it shall ignore the command. When Interrogator locking is not in effect, this command shall be executed if session matching is observed and ignored otherwise. When Interrogator locking is in effect, this command shall be executed only if session and Interrogator ID match, otherwise, it shall be ignored.

7.5.4.3.5.1 Manchester *NAK* command

The Manchester *NAK* command structure is shown in Table 79.

Table 79 — Manchester *NAK* command

	CMD ID	Session	Short Interrogator ID
# of bits	8	2	8
Description	11000000	00: S0 01: S1 10: S2 11: S3	Included only if Interrogator locking is in effect.

Tags shall not reply to a *NAK* command.

7.5.4.3.6 Manchester *Next*

The *Next* command allows an Interrogator to send a single Tag to **hibernate** immediately after singulation or access with a single command/response interchange using the same RN16 from the singulation or access operation. The Tag response to a successful *Next* command gives positive confirmation that the Tag received the command to go to **hibernate**. Alternatively, an Interrogator can send Tags to **hibernate** using the *Deactivate_BAT* command on any and all Tags that match the selected flag conditions without confirmation responses.

7.5.4.3.6.1 Manchester *Next* command

The Manchester *Next* command moves an individual Tag back to **hibernate** using an RN16 or RN16_handle. This command is valid only in the acknowledged, open, and secured states.

Upon deactivation and return to the **hibernate** state, all flags without active timeouts in operation shall return to state A. Flags with active timeouts associated with other sessions that are in state *B* shall continue their timeouts, and upon timeout shall reset to state A. If a timeout was specified in the last activation, then the flag associated with the current Activation Session shall begin its timer operation with Session flag set to state B, and shall revert to state *A* upon timeout. If Session Locking is not in effect then the Tag returns to **hibernate** with all **inventoried** flags set to A. In all cases the Tag returns to **hibernate** with **SL** deasserted.

Tags may have been activated with one sensitivity setting and be taken back to **hibernate** with the same or different sensitivity setting as set by the Hibernate Sensitivity flag in the Manchester *Next* command. Whether a Tag supports high sensitivity is readable by the Interrogator if the Tag supports Tag Capabilities Reporting and Setting (TCRS) as described in 7.6.3.

Table 80 — Manchester *Next* command

	CMD ID	Hibernate Sensitivity NOTE 1	RN
# of bits	8	1	16
Description	1100 1011	0: low sensitivity (approx -30 dBm to +10 dBm) 1: high sensitivity (approximately -30 dBm and below)	RN16 or RN16_handle

NOTE Support for high sensitivity is optional.

7.5.4.3.6.2 Tag response to a Manchester *Next* command

The Tag response to a *Next* shall be as shown in Table 81. Once the Tag sends the *Next* response, it shall transition to the **hibernate** (full or stateful) state.

Table 81 — Tag response to *Next* command

	RN16
# of bits	16
Description	RN16 or RN16_handle

7.5.4.3.7 Manchester *Deactivate_BAT*

7.5.4.3.7.1 Manchester *Deactivate_BAT* command

The *Deactivate_BAT* command is used to conserve power in the Tag by taking groups of Tags no longer required for interrogation and moving them to their lowest power mode **hibernate** state. Tags and Interrogators shall implement the *Deactivate_BAT* command in Table 82. Tags shall execute a *Deactivate_BAT* command from any state (except killed).

In order to reduce the incidence of an Interrogator accidentally sending Tags that are in session with other Interrogators back to the **hibernate** state, the *Deactivate_BAT* command is selective regarding activation session (it obeys Session Locking when in effect) and also specific to that **inventoried** flag state. But, it also possesses Don't Cares also for both the **SL** and Session **inventoried** flag states when more broad application is desired. It also possesses an "Override" field for system resets that will send all Tags that receive the command back to **hibernate** regardless of any flag states.

Tags may have been activated with one sensitivity setting and be taken back to **hibernate** with the same or different sensitivity setting (if supported) as set by the Hibernate Sensitivity flag in the Manchester *Deactivate_BAT* command. Whether a Tag supports high sensitivity is readable by the Interrogator if the Tag supports Tag Capabilities Reporting and Setting (TCRS) as described in 7.6.3.

The Tag shall react to *Deactivate_BAT* as follows:

1. With Session Locking is in effect, then if Activation Session matches command supplied session, and if **SL** state and **inventoried** flag state (target) criteria both match (or if they are Don't Care), then the Tag goes to the **hibernate** or **stateful hibernate** state as a function of timer programming. All flags without active timeouts pending or in operation shall set **inventoried** flags to state A. Flags with active timeouts shall upon return to **hibernate** be set to state *B* with timers beginning (for last Activating Session) or continuing (for other timers that were in effect).

2. With Session Locking in effect and with the Activation Session matching the session specified in the command, but either or both of **SL** and session **inventoried** flag state not matching, the Tag stays in Normal Mode and goes to the **battery ready** state. Assuming well controlled system operation (where **inventoried** flag state *A* means not inventoried and *B* means inventoried), this allows for inventoried Tags to be sent back to **hibernate** while keeping non-inventoried Tags awake for a new Query round.

3. With Session Locking in effect but with the Activation Session not matching the session specified in the command, the Tag ignores the command. This deals with the case where a different Interrogator is properly targeting its own Tags, and a Tag it did not wake up has a timer running on that flag and thus has state *B* on that flag, but should not respond to that different Interrogator.

4. With Session Locking not in effect (and **hibernate** timers thus not used), the Tag shall respond to full match on **SL** and command indicated session **inventoried** flag state (or commanded Don't Care on these states) by returning to **hibernate** with all **inventoried** flags in state *A* and timers cleared.

5. With Session Locking not in effect, the Tag shall respond to mismatch on **SL** and match on command indicated session **inventoried** flag state by going to the **battery ready** state with no change in flag states. With Session Locking not in effect, with any state of **SL** match but mismatch on **inventoried** flag state, the Tag shall go to **battery ready** with **inventoried** flag set to A.

6. If the Override bit is set in the *Deactivate_BAT* and Interrogator Locking is not in effect, then this clears all timers and the Tag shall return to **hibernate** with all flags in state *A* (regardless of Session Locking, **inventoried** match, or **SL** match).

7. If Interrogator Locking is in effect, then Override clears all timers and Select if and only if there is an Interrogator match. Override takes precedence over Session Locking but not over Interrogator Locking.

This action shall elicit no response in any Tag.

Table 82 — Manchester *Deactivate_BAT* command

	CMD ID	Sel	Session	Target	inventoried flag use	Override	Hibernate Sensitivity	Interrogator ID	CRC
# of bits	8	2	2	1	1	1	1	8	16
Description	1100 1010	00: All 01: All 10: ~**SL** 11: **SL**	00:S0 01:S1 10:S2 11:S3	0: A 1: B	0: Don't care for **inventoried** state 1: Do care for **inventoried** state	0:Use **SL** and **inventoried** criteria 1:Go to **hibernate** Always (i.e. all sessions)	0: low sensitivity (approx -30 dBm to +10 dBm) 1: high sensitivity (approximately -30 dBm and below)	Tag checks if Interrogator locking is in effect.	

Table 83 provides examples to take into account the effects of Interrogator Locking.

Table 83 — *Manchester Deactivate_BAT* command procedures

Case	Inter-rogator Locking in effect	Inter-rogator match	Over-ride	Session Locking in effect	Activating session match	Indicated Target matching (or commanded Don't Care)	SL State matching (or commanded Don't Care)	Action	Next state
1	Yes	No	X[2]	X	X	X	X	None	Current state
2	Yes	Yes	Yes	X	X	X	X	Clear Timers, set flags to A	**hibernate**
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Session flag→B	**stateful hibernate**[1]
4	Yes	Yes	No	Yes	Yes	Yes	No	None	**battery ready**
5	Yes	Yes	No	Yes	Yes	No	X	Session flag→A	**battery ready**
6	Yes	Yes	No	Yes	No	X	X	None	Current state
7	No	X	Yes	X	X	X	X	Clear Timers, set flags to A	**hibernate**
8	No	X	No	Yes	Yes	Yes	Yes	Session flag→B	**stateful hibernate**[1]
9	No	X	No	Yes	Yes	Yes	No	None	**battery ready**
10	No	X	No	Yes	Yes	No	X	Session flag→A	**battery ready**
11	No	X	No	Yes	No	X	X	None	Current state
12	No	X	No	No	X	Yes	Yes	Session flag→B	**stateful hibernate**[1]
13	No	X	No	No	X	Yes	No	None	**battery ready**
14	No	X	No	No	X	No	X	Session flag→A	**battery ready**

NOTE 1 If timer was programmed during activation, Session flag→B while timer is operating, after timeout Session flag→A.

NOTE 2 X means that the parameter value does not matter (Don't Care).

7.5.4.3.8 Manchester *Multirate_Reset*

Multirate_Reset is added to provide an override command that can allow Interrogators (mostly portable, but this is not a requirement) to assert control over all awake Manchester Tags no matter what their current state machine state (except killed), **inventoried** flag value, and Forward Data Rate. The *Multirate_Reset* command takes all Tags back to **hibernate** (with specified **hibernate** sensitivity level), so that they may then be reactivated on a single Forward Data Rate. **inventoried** flags are set to *A* and **SL** flag to deasserted, and all associated timers are cleared. This provides a mechanism to start from a known fresh point.

7.5.4.3.8.1 Manchester *Multirate_Reset* command

This command is transmitted sequentially at all Interrogator supported data rates. In between data rates, a time T_4 of CW is allowed for Interrogators to switch data rates, and for Tags to end each command normally. The command code is presented in Table 84. A high level depiction of the command structure is given in Figure 35 (all possible data rates shown, but the Interrogator sends this command only at the data rates supported and in use in a local environment).

Table 84 — Manchester *Multirate_Reset* command

	CMD ID	Hibernate Sensitivity	CRC-16
# of bits	8	1	16
Description	1100 1110	0: low sensitivity (approx -30 dBm to +10 dBm) 1: high sensitivity (approximately -30 dBm and below)	

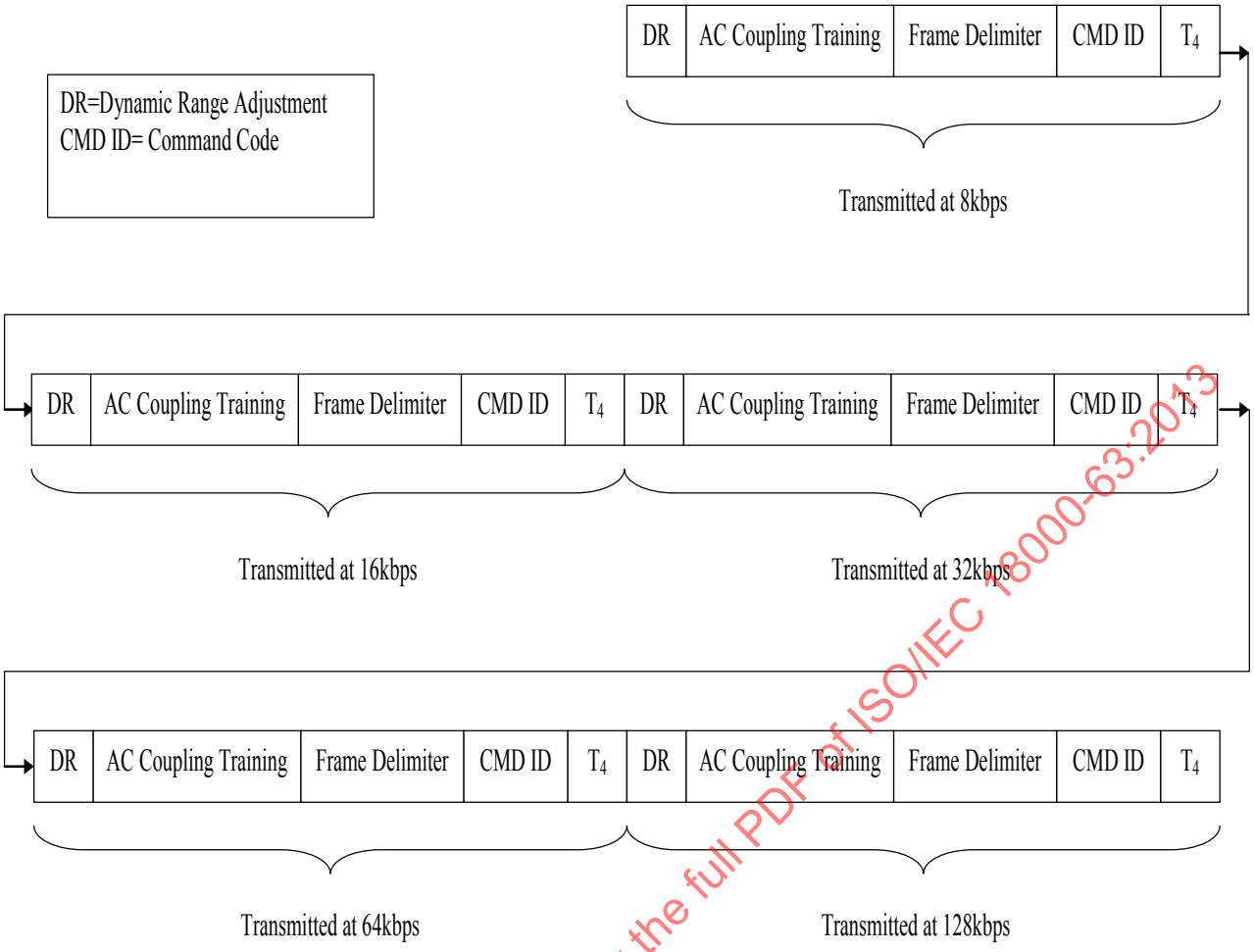


Figure 35 — *Multirate_Reset* command structure

The Tag shall not transmit any response to a *Multirate_Reset* command, but it does take the actions of clearing all **inventoried** flag timers, setting the **inventoried** flag states to A, setting the **SL** to deasserted (**~SL**), and then entering the full **hibernate** state.

7.5.4.4 Access commands

7.5.4.4.1 *OpRegister Read/Write*

Interrogators and Tags shall implement the *OpRegister Read/Write*. Only Tags in the secured state shall execute the *OpRegister Read/Write* command.

7.5.4.4.1.1 *OpRegister Read/Write* command

The *OpRegister Read/Write* command enables reading and writing specific configuration parameters that will be used in low level protocol operations by the Tag circuitry. This enables changing these configuration parameters with immediate application. The registers are not stored in the 4 general memory banks and cannot be read or written with the standard *Read* and *Write* commands, and cannot be matched using the *Select* command.

After issuing an *OpRegister Read/Write* command, an Interrogator shall transmit CW for the lesser of T_{REPLY} or 20 ms, where T_{REPLY} is the time between the Interrogator's *OpRegister Read/Write* command and the Tag's backscattered reply. An Interrogator may observe several possible outcomes from an *OpRegister Read/Write*, depending on the success or failure of the Tag's memory read/write operation.

Upon receiving a valid *OpRegister Read/Write* command with the R/W bit set to 1 indicating a write, a Tag shall write the commanded data into the addressed operational register. The Tag's reply in this write mode shall use the extended preamble, i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query_BAT* that initiated the round.

Upon receiving a valid *OpRegister Read/Write* command with the R/W bit set to 0 indicating a read, the Tag's reply shall use the preamble indicated in the TRext value in the *Query_BAT* that initiated the round.

Table 85 — *OpRegister Read/Write command*

	Command	R/W	Reg ID	WordCount	Data	RN	CRC-16
# of bits	8	1	3	4	Variable	16	16
description	1101 0000	0: Read 1: Write	000: Activation Code (primary) 001: Optional AC (secondary) 010:-111: RFU	Number of words to read/write 0001: MML only 0111: MML+6 AC word	Data to be written	handle	

For an OpRegister Read, if the WordCount is 0, the Tag shall backscatter the entire length of the register identified by Reg ID. For an OpRegister Write, if the WordCount is 0, the Tag shall ignore the command.

For the two AC words, if an invalid count of greater than 7 is received, or if a Reg ID with an RFU value is received, the Tag shall send an error code as defined in Annex K with a value of 00h indicated "other error." The lengths of the other OpRegister parameters are as defined within this standard.

7.5.4.4.1.2 Tag Response to an *OpRegister Read/Write* command

After successfully completing an *OpRegister Read/Write*, the Tag shall backscatter the reply shown in Table 86 within 20 ms of the command. If the Interrogator observes this reply within 20 ms, then the command completed successfully.

If the Tag encounters an error, the Tag shall backscatter an error code during the CW period rather than the reply shown in Table 86 (see Annex I for error-code definitions and for the reply format).

Table 86 — Tag reply to a successful *OpRegister Read/Write* command

	Header	Data	RN	CRC-16
# of bits	1	Read: Variable Write: 0	16	16
description	0: No error	Data if read; none if write	handle	

7.5.4.5 System management commands

7.5.4.5.1 Manchester *Broadcast ID*

7.5.4.5.1.1 *Broadcast ID* command

The *Broadcast ID* command transmits the Interrogator ID and relevant parameters to assist with Interrogator system deployment and management. There are no Tag requirements associated with this command. Specialized tools capable of reading and decoding the Manchester *Broadcast ID* can use this command to perform RF measurements and associate other Interrogator commands containing only short Interrogator ID to a more complete long Interrogator ID. The instances in which the *Broadcast ID* command is used is determined by the implementer.

Table 87 — Manchester *Broadcast ID* command

	CMD ID	ID Length (words)	Long Interrogator ID	Short Interrogator ID	Antenna	Power	Channel	CRC-16
# of bits	8	3	Variable	8	8	8	13	16
Description	1100 1101	Length of long Interrogator ID (16-bit words)	Long Interrogator ID	System assigned short Interrogator ID	Antenna number	2's complement -64 to +63.5 dBm in 0.5 dB step	Channel number in 25 kHz increments; Ch 0 = 830.0 MHz	

7.5.4.5.1.2 Tag Response to a Manchester *Broadcast ID* command

The Tag shall not respond to a *Broadcast ID* command.

7.5.4.6 BAP Manchester Tag PIE support requirements

Manchester Tags and Interrogators shall also support PIE using the definition provided in sub-clause 7.2.

7.6 Extended Protocol Control and Battery Tag Capabilities Reporting and Setting

7.6.1 General

This sub-clause covers Extended Protocol Control (XPC) and BAP Tag capability reporting (Tag to Interrogator) and Tag option setting (generally Interrogator to Tag). Because battery Tags can have a wide range of performance and features, an optional system of reporting of Tag capabilities and options beyond the limitations of XPC is provided. Tag "capabilities" are generally considered to be hardware limits, though it could be possible for them to be upgraded via firmware in a microcontroller based Tag. Within the capabilities of the Tag the Interrogator may control detailed behaviour options with "settings". Very important among settings is duty cycle control that greatly extends Tag battery life. This "Tag Capabilities Reporting and Setting" (TCRS) system is intended to be flexible to allow for the graceful growth of battery RFID system capabilities over time. TCRS may be used for either of BAP PIE or Manchester Tags.

If optional TCRS is supported by the Tag, then Tag capabilities will be documented in a normally locked block or blocks of User memory (the "Capabilities File"), along with default settings that exist prior to Interrogator modification. Note that the Tag must support *BlockPermalock* if the capabilities are to be locked, unless they are permanently "hard locked" by the manufacturer. The current settings will normally be placed in unlocked blocks of User memory (the "Settings File") in order to allow for Interrogator control. Unless the capability block or blocks have been permanently locked by the manufacturer, it can be possible to update capabilities via recommissioning and firmware updating (such as to be compliant to a new standard version). The optional

abilities to rewrite capabilities via recommissioning and to reprogram Tag operating firmware are themselves capabilities that are documented in the Capabilities File.

Because the Interrogator needs efficient access to the capabilities and settings files, and because these files will change over time, they are accessed though a simple directory system that documents their location and size. This system begins with a pointer in TID memory (the TCRS Address and TCRS Map Size) that points to and gives the size of the TCRS Map Table (a simple directory structure). The TCRS Map in turn documents the size and location of the typically locked capabilities and original settings, and typically unlocked settings files that are user adjustable. The Tag may at times of the manufacturers' choosing, such as upon activation or after Tag memory writes, check the current settings and update internal control registers.

To provide for the chain of capabilities from simple passive Tags to the most advanced of battery assisted Tags, a flag system is used to indicate increasing levels of capability. A flag in PC indicates that XPC is supported. A flag in XPC_W1 indicates that TCRS is supported. A word in the TCRS Map Table indicates the version of TCRS support the Tag implements, which defines the number of capability and setting words, and the detailed interpretation of those words. Flags and fields in the capabilities words of the identified TCRS version indicate whether particular optional capabilities are supported, such as duty cycling to extend battery life. All of these capabilities are optional.

7.6.2 Extended Protocol Control definition

The optional Extended Protocol (XPC) Field has been defined to enable the Interrogator to support either sensors or more advanced battery operation or both. The first XPC word XPC_W1 provides additional information about Tag capabilities. To indicate to the Interrogator that XPC_W1 is supported, normal Protocol Control Bit 6 XI flag (the 7th bit of the PC word, which is UII bit 16_h) shall be used to indicate the presence of XPC_W1. The XPC_W1 shall consist of 16 additional protocol control bits as shown in Figure 36 and described in the below text and in Table 88. Still more Tag capabilities may be optionally described in the TCRS structure as defined in this sub-clause.

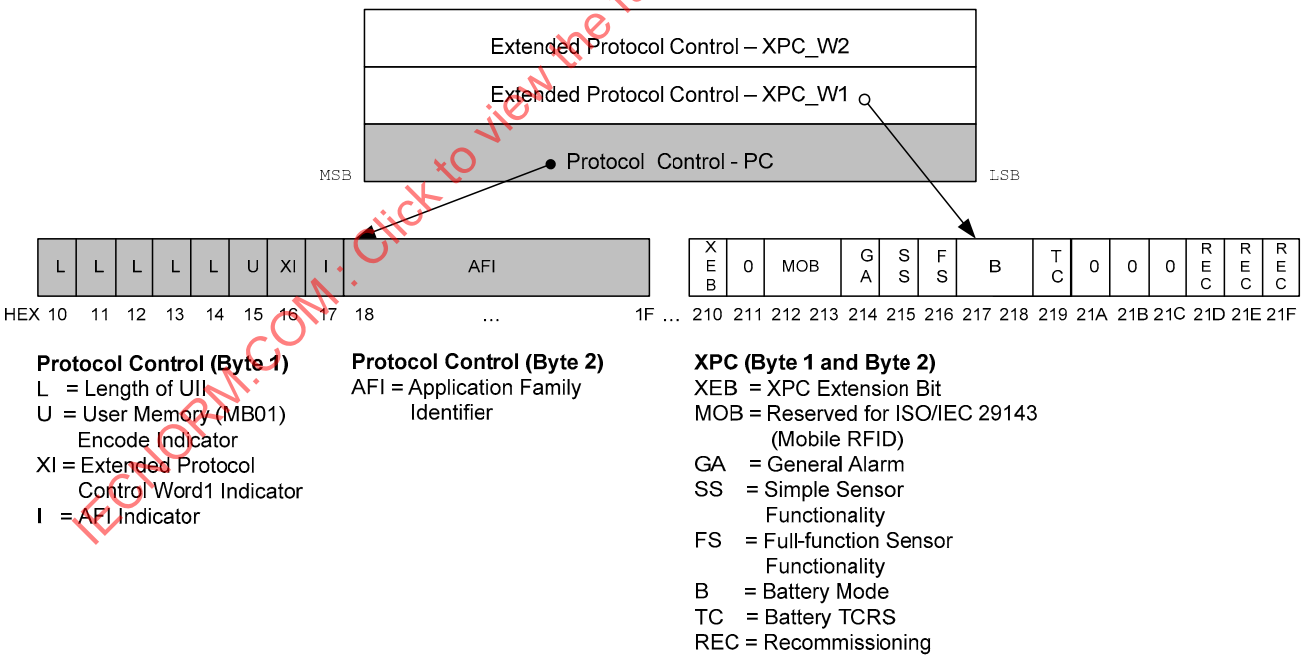


Figure 36 — Extended Protocol Control bit definitions (see NOTES 1 and 2)

NOTE 1 Figure 36 shows how Protocol Control and Extended Protocol Control are linked to each other and how the part of a Tag reply to an *ACK* command dedicated to PC and XPC is structured, assumed that both XPC_W1 and XPC_W2 (currently RFU) are supported. PC and XPC are not stored on subsequent addresses of the UII memory (see Figure 17).

NOTE 2 At the option of the Tag manufacturer, the General Alarm (GA) bit may indicate one or more of sensor alarms, low battery, or any other exception conditions on the tag.

As defined in clause 6.4.2.1, the XPC_W1 begins in Bit 210_h and ends in BIT $21F_h$ in the UII memory. Bit 210h shall be the XPC Extension Bit (XEB). If a Tag does not implement an XPC_W2 then the XEB shall be zero. Bit 214_h shall be the Sensor Alarm (SA) flag, Bit 215_h shall be the Simple Sensor (SS) flag, and Bit 216_h shall be the Full Function Sensor (FS) flag. Bits 217_h and 218_h shall be set according to the supported battery mode. The battery mode bit codes currently defined shall include passive and Battery Assisted Passive, and the two remaining codes are RFU. The Battery TCRS present or absent flag TC at 219_h shall indicate to the Interrogator whether the Tag supports additional information on its capabilities in User memory under the Tag Capabilities Reporting and Setting (TCRS) option.

XPC bits $21D_h$-$21F_h$ are the recommissioning bits as defined in sub-clause 6.4.2.10 governing Killing or Recommissioning a Tag. Battery Tags do not lose power until the battery is depleted, hence there is a need to provide a modified definition of the time when XI is recalculated after certain recommissioning procedures (see 6.4.2.1.2.2). The selected definition of the time when XI is recalculated after certain recommissioning procedures for battery Tags shall be the time following loss of signal relative to the Tag sensitivity level (whatever that may be) or the next time it goes to **hibernate**.

All remaining bits of XPC_W1 as well as all bits of XPC_W2 are currently reserved for future use (RFU) and shall be set to 0 at default.

If TCRS is supported as indicated by the XPC_W1 TC flag, then User memory TCRS files shall be defined according to one of the definitions in sub-clause 7.6.3.

The Sensor Alarm bit shall be set to 1 to indicate the violation of at least one alarm condition in at least one of the attached sensors.

The detailed meaning of combinations of the XPC_W1 flags is described in Table 88 and in its detailed notes.

Table 88 — Meaning of combinations of B field, and SS and FS bits in XPC

B Field Bits 217_h and 218_h	TC Bit Bit 219_h	SS Bit Bit 215_h	FS Bit Bit 216_h	Meaning
00	X/0	0	0	Passive Tag without sensors. TC field is Don't Care when B = 00, and there is no support for TCRS. As a matter of standard practice, with B field = 00, TC flag should be set to 0.
00	X/0	0	1	Passive Tag with at least one Full-function Sensor but not Simple Sensor, no support for TCRS.
00	X/0	1	0	Passive Tag with one pre-configured Simple Sensor, no support for TCRS.
00	X/0	1	1	Passive Tag with at least one Full-function Sensor and one pre-configured Simple Sensor, or Simple Sensor Functionality supported by one Full-function sensor. No support for TCRS.
01	0/1	0	0	BAP Tag without sensors. If TC = 0, no support for TCRS.
01	0/1	0	1	BAP Tag and at least one Full-function Sensor.
01	0/1	1	0	BAP Tag with one pre-configured Simple Sensor.
01	0/1	1	1	BAP Tag, at least one Full-function Sensor available and one pre-configured Simple Sensor available, or Simple Sensor Functionality supported by one Full-function sensor.
10	X	X	X	Reserved for future use with a more advanced battery assisted mode.
11	X	X	X	Reserved for future use with a more advanced battery assisted mode.

7.6.3 Battery Assisted Passive Tag Capability Reporting, Setting, and duty cycle/mode control (optional)

This sub-clause applies to any BAP Tags that support optional TCRS, which may include either of BAP PIE or Manchester Tags.

Battery Tags can be very sophisticated wireless terminals with many options, and the TCRS system provides a convenient and adaptable means for the Tag to report its possibly complex capabilities, and for the Interrogator to command behavioural modes. The sub-option to implement Interrogator control of duty cycle to optimize battery power and Tag latency exists under TCRS.

Reporting and controlling such capabilities is optional as follows:

1. BAP Tags that do not support XPC do not support TCRS. This is because XPC provides the flag that indicates that TCRS is supported.

2. Passive Tags that support XPC (for sensor data) do not support TCRS.

3. Any BAP Tag that supports XPC has the option to support TCRS. Whether a Tag supports TCRS is flagged in XPC_W1, and the Interrogator may access special Battery Capability Words (defined below in this sub-clause) as desired for detailed information.

Among capabilities to report are the possible combinations of BAP PIE, Manchester, and Dead Battery Response. Various capabilities in respect to these modes are reported within the TCRS system. For example, there is a precision oscillator on the Tag if Manchester is supported. This oscillator has accuracy of 4% or better for standard BLFs, and 1.5% or better for optional BLFs. This oscillator could be trimmed for improved performance, or could be crystal controlled for still better performance. Crystal oscillators may allow for exact data rate control (within their specified accuracy), or might be selected for other functions such as time logging (e.g. 32.768 kHz Real Time Clock) which may fit within minimum backscatter link frequency control requirements (4%) but not provide full crystal accuracy for BLF control. Another example of TCRS usage is reporting Tag sensitivity within the wide range of Tag sensitivities that can exist, from approximately -80 dBm (Manchester with narrow filter and RF low noise amplifier) to approximately -10 dBm. The Interrogators may use awareness of Tag sensitivity in adapting their own behaviour in accessing these Tags and controlling interference.

Additionally, Tag behaviour under its capabilities may be adjusted by the Interrogator using settings, such as Tag **hibernate** or Normal Mode duty cycle. Just as duty cycling of BAP PIE Tags may extend battery life, duty cycling of the Manchester **hibernate** mode may also allow significant battery life extension.

To provide for currently defined capabilities and adjustments and for future growth, a normally locked Tag Capabilities File and a normally unlocked Tag Settings File are defined. These files are typically stored in User memory. To allow the locking of the block or blocks of User memory where the Capabilities File is stored, the passive Tag optional *BlockPermalock* command of clause 6.4.2.11.3.9 should be supported (the full set of optional access commands is recommended for BAP Tags). These files are pointed to by a structure very similar to the Sensor Directory System (SDS), called the TCRS Map. The TCRS Map is pointed to by the TCRS Address, which is stored in TID memory. The TCRS Address also includes a word for TCRS Map Size, so that the Interrogator may be immediately aware upon reading the TCRS Address of not only where the TCRS Map is, but how many words the Interrogator needs to read to capture full TCRS Map information in a single read. The TCRS Map provides pointers to the Tag Capabilities and Tag Settings files, and also key information on TCRS Version Number (which may be updated with each standard update), User memory size, and block size (for the *BlockPermalock* command).

This data structure is shown in Figure 37 below. The structure of the 32 bit TCRS Address in TID memory is shown in Table 89. The Lock Status Flag that indicates if the TCRS Map is in a memory block that was locked at the time the TCRS Address and TCRS Map were first programmed.

The first six words in the TCRS Map are not entries, but a header that contains critical information about TCRS and User memory. This includes TCRS Version as defined in Table 90, a 48 bit size indicator of User Memory, and a Block Size for BlockPermalock operations. The use of three memory size words is due to the 4

gigaword limit of a two word size, which may not be adequate in the future in the case of Tags that also have a fast wired interface such as USB (USB memory sticks of greater capacity already exist). The TCRS Version, User memory size, and block size parameters are all stored Most Significant Bit first.

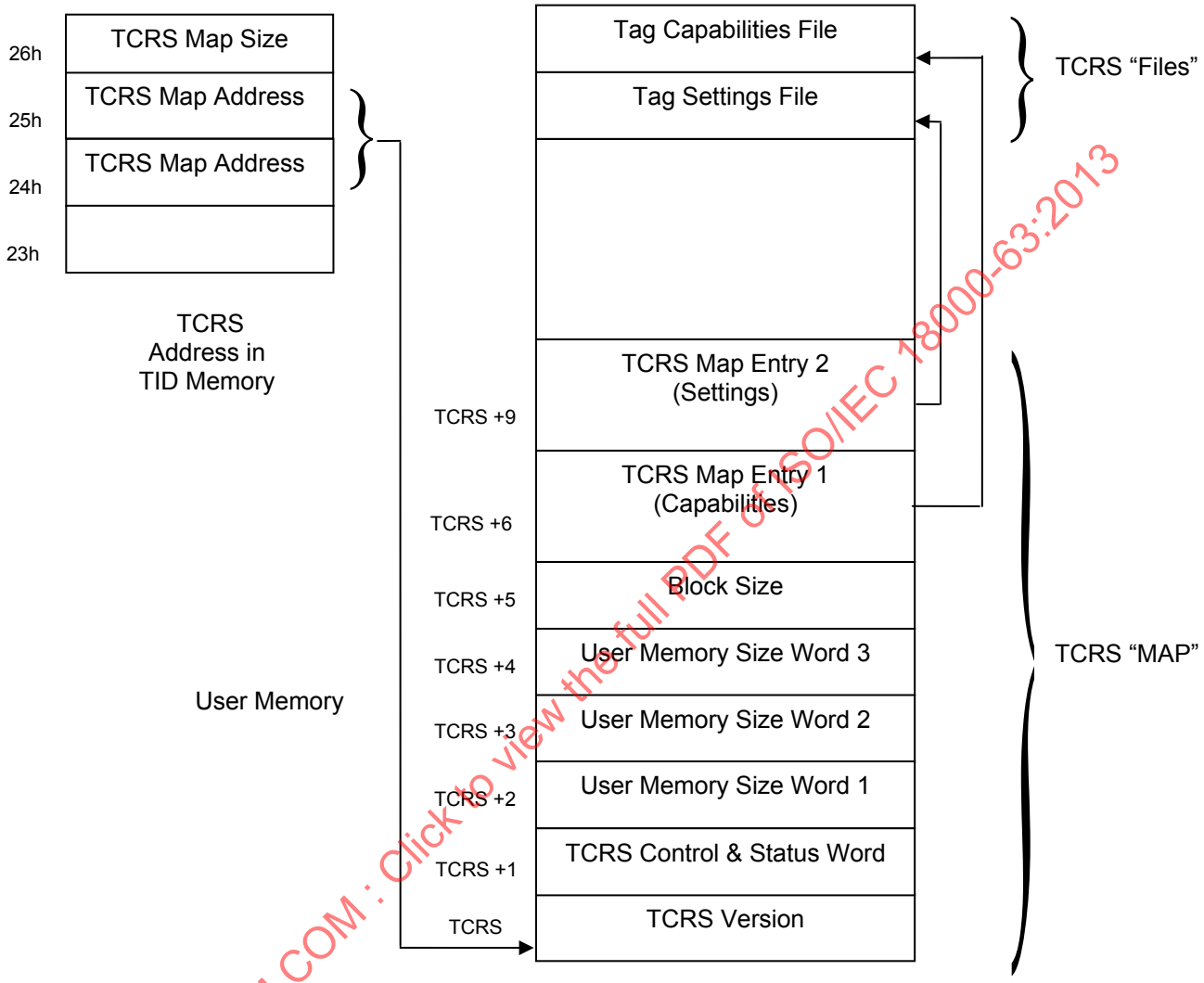


Figure 37 — Relationship of TCRS Address, TCRS Map, and TCRS Capabilities and Setting Files (for Version Code = 1).

Table 89 — Structure of TCRS Address (stored in TID words 24h and 25h)

	RFU	Lock Status of TCRS Map	MB	Word Address
# of bits	5	1	2	24
Description	Reserved for future use	0: Unlocked 1: Locked	Memory bank selector	TCRS starting word address (non-EBV, standard binary address)

The TCRS Version Code system is illustrated in Table 90.

Table 90 — TCRS Version Codes

Version Code (16 bits, shown in base 10)	Interpretation	Comments
0	This Tag does not support TCRS	
1	In accordance with this standard version and no other.	
2 to 32,765	RFU for future ISO standardized versions, allowing for 32,766 future variations.	
32,766 to 49,149	RFU for future non-ISO standardized version	16,284 unique codes reserved for non-ISO standards.
49,150 to 65,535	RFU for manufacturer specific TCRS	16,284 unique codes reserved for custom, non-standardized TCRS.

The structure of the three word entries that make up the TCRS Map following its general information is given in Table 91. This entry structure provides a starting address, a maximum range, and an actually used range. Ranges are interpreted literally, meaning that a range code of 255 actually does mean 255 words and not 256. This is similar to the *BlockWrite* command, which currently has a maximum word count of 255 for an 8 bit WordCount parameter. Thus, a used range of zero in a TCRS Map Entry means the file is pointed to and reserved, but currently empty. This structure also provides a flag to indicate if the pointed entry is in a locked or unlocked block of memory.

Table 91 — Structure of TCRS Map Entries (file pointers for Version Code =1)

	RFU	Lock Status	MB	Word Address	Maximum Range	Used Range
# of bits	5	1	2	24	8	8
Description	Reserved for future use	0: Unlocked 1: Locked	Memory bank selector	File word address (non-EBV, linear address)	Memory word max range (non-EBV)	Memory word used range (non-EBV)

Note that the TCRS Map Entries have a Lock Status flag that indicates if the file pointed to is in a locked block of memory. Currently only User memory can have blocks locked against rewriting by the *BlockPermalock* command, though this is subject to possible change. In the case of the TCRS system, normal operation is to have the Tag Capabilities File containing capabilities and default settings in a locked block of User memory, and to have the Tag Settings File in an unlocked block of User memory. However, manufacturers and users are not required to follow that practice. As a simple form of directory structure, it is appropriate for the TCRS Map to indicate if the pointed files are currently block permalocked or not. Similarly, the TCRS address in TID memory that points to the TCRS Map indicates if the TCRS Map was locked at the time it was first written. If the TCRS Address indicates that the TCRS Map was not originally locked, but the TCRS Map Control and Status Word indicates the TCRS Map is locked, this indicates the TCRS Map was later locked and thus may have been altered from original programming. Leaving the TCRS Map unlocked at original programming may be desired in some cases in order to allow the TCRS Map to be updated without recommissioning the Tag.

The “TCRS Control & Status Word” of Figure 37 above and Table 92 below provides additional information about the TCRS Map itself and other key memory control information. Currently it only provides the lock status of the TCRS Map.

Table 92 — Structure of TCRS Control & Status Word

	RFU	TCRS Lock Status
# of bits	15	1
Description	Reserved for future use	0: TCRS Map is in an unlocked block 1: TCRS Map is in a locked block

The combination of the Version Code and the TRCS Map Entry structure is sufficient to define the file structure of the Capability and Setting files. To complete the definition of TCRS for this standard version requires definition of the details of Battery Capabilities Word(s) (BCWs), Battery Default Setting Word(s) (BDSWs), and Battery Setting Word(s) (BSWs) that are in the Capabilities and Settings files.

The Battery Capabilities Words (BCW’s) are generally hardware supported capabilities, and are thus intended (but not required) to be in a locked block of User memory. The TCRS Entry pointing to the Tag Capabilities File has a flag to indicate if the file is locked or unlocked. However, even locked blocks can be rewritten after recommissioning unless the block was “hard locked” at manufacturing. It is also possible for a microcontroller based Tag to change its capabilities via a firmware upgrade. In that case, so long as the manufacturer has not locked the BCW’s permanently at time of manufacture, then a recommissioning of the Tag along with a firmware upgrade can change the locked BCW’s.

Battery Setting Words allow for Interrogator programming of behavioural features of the Tag. A key kind of setting that Interrogators may reprogram is the listen duty cycle of the Tag in the various modes it supports. Battery Setting Words are thus normally in unlocked User memory, though a user or application could BlockPermalock the BSWs if it is desired to prevent later changes. The TCRS Entry pointing to the Tag Settings File containing the Battery Settings Words has a flag indicating if the file is locked or unlocked.

Battery Default Setting Words (BDSWs) are the initial settings established by the manufacturer or first programmer of the Tag. They may be used to re-establish settings after any inadvertent rewrites of the BSWs, or after recommissioning. Since they must be reliably available, they are logically stored in a locked block of user memory. In this version of this International Standard they are assumed to be part of the Capabilities File, and to follow immediately after the BCW’s defined for this version.

The **Tag Capabilities File** for this standard (Version Code = 1) shall contain as its first word the Battery Capabilities Word 1 as defined in Table 93 followed in ascending order by the other BCW’s. It shall also contain in order the Battery Default Setting Words (BDSWs) defined for this version that are the manufacturer’s original settings with structures as described in the Battery Setting Word tables of this sub-clause (the BDSWs are merely copies of the original BSWs).

The **Tag Settings File** shall contain in ascending order the Battery Setting Words described in this sub-clause. The Tag Settings file is normally stored in an unlocked block of User Memory. Unless the Tag Settings File is locked, duty cycles as well as other settings may normally be programmed as users desire.

Fields that are not used or that do not apply are generally set to zeroes in the Capability and Setting files.

The Version Code = 1 structure of the Tag Capabilities and Settings files, with references to the word definition tables of this version, are shown in Figure 38.

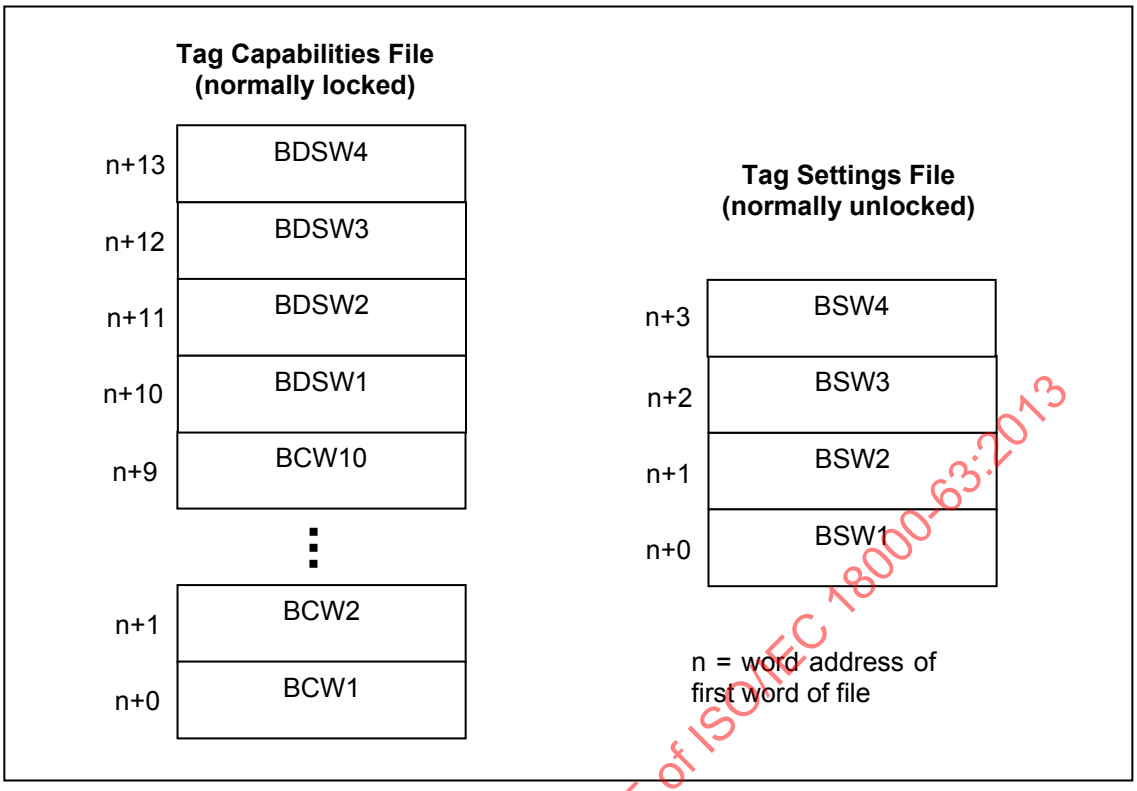


Figure 38 — Structure of Tag Capabilities and Tag Settings Files for Version Code =1.

Battery Capabilities Words (BCW's) for this standard version are given below.

Table 93 — Battery Capabilities Word 1 (BCW1) for the Case of Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Persistence Maximums (PM) flag 1 bit NOTE 1	Bits 0	0: Optional persistence maximums on flags S2, S3, and **SL** are not supported. 1: Optional persistence maximums on flags S2, S3, and **SL** are supported. (See sub-clause 7.3.3)
Low Power Listen (LPL) flag 1 bit When **low power listen** exists for Tags that support duty cycling of BAP PIE, this may mean that the Tag uses the **low power listen** state instead of the **sleep** state.	Bit 1	0: LPL is not supported. 1: LPL is supported.
LPL Parallel (LPLP) flag 1 bit	Bit 2	0: LPL cannot parallel other modes (or does not apply), such as Manchester Hibernation. 1: LPL can parallel other modes.

Bit meaning	Bit assignments	Behaviour
Low Power Listen Sensitivity field (LPLS) 2 bits	Bits 3-4	00: LPL sensitivity > 0 dBm (or does not apply) 01: -10 dBm < Sensitivity < 0 dBm 10: -15 dBm < Sensitivity < -10 dBm 11: Sensitivity < -15 dBm NOTE: Nominal (not guaranteed) sensitivity over band of operation with symbol time ~ 25 µs. This sensitivity is assumed to apply to any **low power listen** modulation forms, and may vary between forms. For example, a LPL capability in Manchester means that the Tag in "**deep hibernate**" is not listening for activation but is listening for RF that can trigger to Tag to return to **hibernate** ahead of its programmed duty cycle.
Dead Battery Response (DBR) Flag 1 bit	Bit 5	0: DBR is not supported. 1: DBR is supported.
Dead Battery Response Sensitivity (DBRS) Field 2 bits	Bits 6-7	00: DBR sensitivity > 0 dBm (or does not apply) 01: -10 dBm < Sensitivity < 0 dBm 10: -15 dBm < Sensitivity < -10 dBm 11: Sensitivity < -15 dBm NOTE: Nominal (not guaranteed) sensitivity over band of operation at Tari = 25 µs
BAP PIE (BPIE) flag 1 bit	Bit 8	0: BAP PIE is not supported. 1: BAP PIE is supported.
BAP PIE Parallel (BPIE_P) flag 1 bit	Bit 9	0: BAP PIE cannot parallel other modes (or does not apply). 1: BAP PIE can parallel other modes while in **listen** (example, Manchester activation).
BAP PIE Low Power Mode (BPIE_LPM) field 2 bits	Bit 10-11	00: Tag does not support a low power mode (waits in **battery ready**) 01: Tag supports **sleep-listen** only 10: Tag supports continuous **low power listen** only 11: Tag supports both **listen** and **low power listen**

Bit meaning	Bit assignments	Behaviour
BAP PIE Duty Cycle Programmability (BPIE_DCP) flag 1 bit	Bit 12	0: Tag listen-sleep(or **low power listen**) cycle is manufacturer fixed (or does not apply) 1: Tag listen-sleep (or **low power listen**) cycle is programmable via Settings
BAP PIE Listen Sensitivity (BPIE_L_S) field 2 bits	Bits 13-14	00: Nominal Tag sens > -10 dBm (or does not apply) 01: -20 dBm < Tag sens < -10 dBm 10: -30 dBm < Tag sens < -20 dBm 11: Nominal Tag sens < -30 dBm NOTE: The sensitivities above refer to nominal listen sensitivity over design band of operation at Tari = 25 µs. Normal sensitivity may vary with data rate. Low Power Listen Sensitivity, if supported for BAP PIE, is as given in bits 3-4 of this word.
RFU 1 bit	Bit 15	

NOTE 1 BAP PIE Tags have the option of whether to support Persistence Maximums, so this flag is helpful to indicate this feature exists. For both passive and BAP PIE S0 has no persistence other than brief signal loss tolerance due to hardware delay for passive and optionally INACT_T for BAP PIE. S1 has a specified maximum for both passive and BAP PIE. See sub-clause 7.3.3 for more information.

Table 94 — Battery Capabilities Word 2 (BCW2) for the Case of Version Code = 1.

Bit meaning	Bit assignments	Behaviour
RFU	Bit 0	
RFU	Bit 1	
RFU	Bit 2	
Manchester (M) flag 1 bit	Bit 3	0: Manchester forward link not supported 1: Manchester is supported
Manchester Hibernate Parallel (MHP) flag 1 bit	Bit 4	0: Manchester Hibernate cannot parallel other modes (or does not apply) 1: Manchester Hibernate can parallel other modes while receiving

Bit meaning	Bit assignments	Behaviour
Manchester Normal Parallel (MNP) flag 1 bit	Bit 5	0: Manchester normal cannot parallel other modes (or does not apply) 1: Manchester normal can parallel other modes while receiving
Manchester Hibernate Sensitivity (MS) field 2 bits	Bits 6-7	00: Nominal Tag sens > -30 dBm (or does not apply) 01: Nominal Tag sens < -30 dBm 10: Nominal Tag sens < -40 dBm 11: Nominal Tag sens < -50 dBm NOTE: The sensitivities above refer to Hibernate sensitivity. Normal sensitivity may vary with data rate.
Manchester Sensitivity Setting (MSS) flag 1 bit Sensitivity setting may be controlled in *Activation* command for Normal Mode and in *Deactivate_BAT* and *Next* for **hibernate**	Bit 8	0: Manchester does not support adjustable sensitivity (or does not apply) 1: Manchester does support adjustable sensitivity
Manchester BLF (MB) flag 1 bit	Bit 9	0: Manchester supports only mandatory BLFs (or does not apply) 1: Manchester also supports currently defined optional BLFs
Hibernate Duty Cycle (Hib_DC) flag 1 bit	Bit 10	0: Tag does not have duty cycling of **hibernate** 1: Tag does support duty cycling of **hibernate**
Hibernate Duty Cycle Programmability (Hib_DC_Prog) flag 1 bit	Bit 11	0: Hibernate Rx duty cycles are manufacturer fixed (or do not apply) 1: Duty cycles are programmable

Bit meaning	Bit assignments	Behaviour
Tag Duty Cycle Control Ability (Tag_DC_ConAbility) flag This flag applies to BAP PIE and Manchester. For BAP PIE it refers to **listen** time for regular communications. For Manchester it refers to **hibernate** listen time. This flag reports that the Tag has the ability to self alter its **listen** and **sleep** (or **deep hibernate**) times for two purposes: 1. Tag may increase listen time or duty cycle to improve response as function of time since last communication. 2. Tag may decrease listen time or duty cycle to preserve battery in the case of low battery life. The definition of low battery is at the manufacturer's choice. Algorithms and methods for increasing duty cycle and/or **listen** time are at the option battery case of the Tag manufacturer. The Tag must be authorized in the Settings file to decrease listen time or duty cycle. This is via the Tag_DC_Auth flag in BSW4. If the Tag does change programmed settings, it will report this via the Tag_DC_Rep flag in BSW4.	Bit 12	0: Tag follows fixed or programmed duty cycling without memory of time since last communication or current battery state (or does not apply) 1: Tag adjusts **listen** and/or **sleep** time to increase chances of correctly receiving new commands or activations if communications have been detected recently, or to preserve battery life in the low battery case
RFU	Bit 13	
RFU	Bit 14	
RFU	Bit 15	

Table 95 — Battery Capabilities Word 3 (BCW3) for the Case of Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Reference Oscillator Presence (ROP) field 2 bits	Bits 0-1	00: Tag does not have an independent oscillator (timing is temporarily derived from RTcal and TRcal symbols or other Interrogator assisted methods) (or does not apply) 01: Tag has an on board oscillator of at least +/-40% accuracy over nominal temp range and +/-50% accuracy over extended temp range (independent of any Real Time Clock) 10: Tag has RO trimmed via Interrogator 11: Tag uses Real Time Clock to derive reference oscillator
Reference Oscillator Accuracy (ROA) field 4 bits This accuracy applies over the Tag's specified temp range and at the point of use for timing and BLF control. For example, a 32.00 KHz RTC with 50 ppm accuracy multiplied up to a 3.84 MHz reference for RO generation will generate 3.84 MHz within 50 ppm and is described under code 1010. But, a 32.768 KHz RTC multiplied to be as close as possible to 3.84 MHz (3.833856 MHz) is 0.16% low, and has accuracy code 0101.) To put these accuracies in practical context, typical specifications of low frequency real time clock crystals are 20, 50, and 100 ppm. These are highly accurate for subcarrier / BLF control purposes, but generally inadequate for carrier generation in an active radio. 2 ppm xtal option is the typical low cost cellular handset type VCTXO, suitable for very narrowband channel control.	Bits 2-5	0000: < 40% over nom. temp range, < 50% over extended temp range (code 0000 only useful for hibernate and persistence timing) (or does not apply) 0001: < 20% nom. / 25% ext. temp range 0010: < 10% nom. / 15% ext. temp range 0011: < 4% (useful for BLFs <= 640 KHz) 0100: <1.5% (useful for BLFs up to 1920 KHz) 0101: < 0.5% 0110: < 1000 ppm 0111: < 500 ppm 1000: < 200 ppm 1001: < 100 ppm 1010: < 50 ppm 1011: < 20 ppm 1100: < 10 ppm 1101: < 5 ppm 1110: < 2 ppm 1111: < 1 ppm

Bit meaning	Bit assignments	Behaviour
Real Time Clock (RTC) field 4 bits	Bits 6-9	0000: None 0001: Free running RC or IC (non-xtal) 0010: RC or IC trimmed by Interrogator 0011: 20 KHz xtal 0100: 32.00 KHz xtal 0101: 32.768 KHz xtal 0110 – 1110: RFU 1111: Other
Real Time Clock Accuracy (RTCA) field 4 bits Applies over temperature range the Tag is specified to operate over	Bits 10-13	0000: < 40% over nom. temp range, < 50% over extended temp range (code 0000 only useful for **hibernate** and persistence timing) (or does not apply) 0001: < 20% nom. / 25% ext. temp range 0010: < 10% nom. / 15% ext. temp range 0011: < 4% (useful for BLFs <= 640 KHz) 0100: <1.5% (useful for BLFs up to 1920 KHz) 0101: < 0.5% 0110: < 1000 ppm 0111: < 500 ppm 1000: < 200 ppm 1001: < 100 ppm 1010: < 50 ppm 1011: < 20 pp 1100: < 10 ppm 1101: < 5 ppm 1110: < 2 ppm 1111: < 1 ppm
RFU 2 bits	Bits 14-15	

Table 96 — Battery Capabilities Word 4 (BCW4) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
BLF Accuracy (BLFA) field 2 bits NOTE 1	Bits 0-1	00: RO accuracy applies 01: RTC accuracy applies 10: Interrogator trimmed accuracy applies 11: Independent standard compliant accuracy applies
Hibernate Accuracy (HA) field 2 bits NOTE 1	Bits 2-3	00: RO accuracy applies 01: RTC accuracy applies 10: Interrogator trimmed accuracy applies 11: Independent standard compliant accuracy applies
Persistence Accuracy (HA) field 2 bits NOTE 1	Bit 4-5	00: RO accuracy applies 01: RTC accuracy applies 10: Interrogator trimmed accuracy applies 11: Independent standard compliant accuracy applies
Electronic Working Temperature Range (EWTR) field 2 bits The Electronic Working Temperature range is the limit of reliable circuit operation independent of the current battery limits as given by the Battery Working Temperature Range in BSW1. The battery related temperature ranges are in the setting words because the battery may be replaceable by a battery with different temperature characteristics than the original battery.	Bits 6-7	00: Nominal temp range -25 to +40 C 01: Extended temp range -40 to +65 C 10: RFU 11: RFU
Battery Replace (BR) Field 2 bits	Bits 8-9	00: Battery is not user replaceable 01: Battery is user replaceable 10: Batter is not service centre replaceable 11: Battery is service centre replaceable
Sensor Command Support (SCS) flag 1 bit	Bit 10	0: Tag does not support optional sensor access command 1: Tag does support optional sensor access command

Bit meaning	Bit assignments	Behaviour
Flex_Query (FQ) flag 1 bit	Bit 11	0: Tag does not support PIE *Flex_Query* command 1: Tag does support PIE *Flex_Query* command
PIE Access Commands flag 1 bit The optional PIE and Manchester access commands include *Access*, *BlockWrite*, *BlockErase*, and *BlockPermalock*. This cluster is strongly recommended for battery Tags.	Bit 12	0: Tag does not support clause 6.4.2.11.3 optional access commands 1: Tag does support all optional access commands
Capabilities File Update (CFW) field 2 bits	Bits 13-14	00: Capabilities file is not locked and Tag design allows updating (or does not apply) 01: Capabilities file is not locked, but should not be updated 10: Capabilities file is BlockPermalocked and Tag design allows updating via recommissioning 11: Capabilities file BlockPermalocked and should not be updated via recommissioning, or is hard locked and cannot be altered by any means
RFU 1 bit	Bit 15	

NOTE 2 The documenting of Reference Oscillator and Real Time Clock presence, methods, and accuracy allows the major timing requirements of bit link frequency, persistence (if timed), and hibernation to have their accuracy precisely specified by referencing its source. These accuracies apply over the temperature range the Tag is specified to operate over.

Table 97 — Battery Capabilities Word 5 (BCW5) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Firmware Reprogram (FR) field 2 bits	Bits 0-1	00: Tag cannot be reprogrammed 01: Tag is firmware based and can be reprogrammed via hardware port 10: Tag is firmware based and can be reprogrammed via RF link 11: Tag can only be reprogrammed via service centre
Serial Port (SP) field 3 bits	Bits 2-4	000: Tag does not have optional serial port 001: Tag has USB serial port 010 – 110: RFU 111: Tag has manufacturer specific serial port
On/Off Control (OOC) field 3 bits	Bits 5-7	000: Tag has no On-Off control 001: Tag has one time mechanical Off to On switch 010: Tag has reusable mechanical On-Off switch 011: Tag has specialized On-Off control (example, magnetic switch) 100-111 RFU
Battery Assisted Passive Frequency Range (BAPFR) field 4 bits	Bits 8-11	0000: Not reported 0001: 600 – 1200 MHz 0010: 700 – 1100 MHz 0011: 800 – 1000 MHz 0100: 860 – 960 MHz 0101: 865 – 868 MHz 0110: 952-954 MHz 0111: 902-928 MHz 1000: 915-925 MHz 1001-1111: RFU Closest best estimate of intended coverage for Battery Assisted Passive behaviour As a general guide, for fixed frequency Tags the 10 dB bandwidth of the smaller of transmit and receive bandwidths For adjustable frequency Tags, the intended adjustable range

Bit meaning	Bit assignments	Behaviour
Frequency Trim (FT) flag 1 bit	Bit 12	0: Frequency of operation is not Interrogator trimmable 1: Frequency of operation is Interrogator trimmable
Transmit Power Control (TPC) field 2 bits	Bits 13-14	00: Tag transmit power is based on reader carrier power 01: Tag may be reader commanded to adjust backscatter 10: Tag may self adjust backscatter based on receive signal strength 11: RFU
RFU 1 bit	Bit 15	

Table 98 — Battery Capabilities Word 6 (BCW6) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Tag INACT_T Status field (INACT_T_S field) 2 bits	Bits 0-1	00: Tag does not support INACT_T 01: Tag supports fixed INACT_T (value given just below) 10: Tag supports programmable INACT_T (per Settings) 11: RFU
INACT_T State Dependence (INACT_T_SD) flag 1 bit	Bit 2	0: INACT_T is not state dependent (or does not apply) 1: INACT_T is state dependent State dependence means that INACT_T refresh uses different validity criteria for commands based on Tag state. An example is using RN16 verification for appropriate states.

Bit meaning	Bit assignments	Behaviour
Fixed INACT_T Value (INACT_T_V) field 3 bits	Bits 3-5	000: 50 ms (or does not apply) 001: 100 ms 010: 250 ms 011: 500 ms 100: 1 sec 101: 2 sec 110: 4 sec 111: 8 sec or more NOTE: Fixed INACT_T need not be these exact values. If not, this field shall be the closest representative of actual INACT_T.
Global Timeout Status (GTS) field 3 bits	Bits 6-8	000: Tag does not support any Global Timeout 001: Tag supports fixed Global Timeout (if so, then value below in Fixed Global Timeout field) 010: Tag supports programmable Global Timeout (per Settings) 011: Tag supports fixed Selective Global Timeout (if so, value below) 100: Tag supports programmable Selective Global Timeout (per settings) 101-111: RFU Selective Global Timeout is refreshed based on receiving valid commands.
Selective Global Timeout State Dependence (SGT_SD) flag 1 bit	Bit 9	0: SGT is state dependent (if supported) (or does not apply) 1: SGT is not state dependent State dependence means that the Selective Global Timeout refresh uses different validity criteria for commands based on Tag state. An example is verifying session if Session Locking is in effect.

Bit meaning	Bit assignments	Behaviour
Fixed Global Timeout or Selective Global Timeout value field 3 bits	Bits 10-12	000: 50 ms (or does not apply) 001: 100 ms 010: 250 ms 011: 500 ms 100: 1 sec 101: 2 sec 110: 4 sec 111: 8 sec or greater NOTE: Fixed (Selective) Global Timeout need not be these exact values. If not, this field shall be the closest representative of that implemented.
BAP Programmable Duty Cycling Resolutions (Pro_DC_Res) flag 1 bit The codes to specify programmable duty cycle times to high resolution down to 500 µs are supported, but the Tag circuit design may not support all of those. Minimums supported are specified in BCW8.	Bit 13	0: Low resolutions (250 ms and up, similar to programmable Hibernation persistence and Hibernation duty cycles) (or does not apply) 1: High resolution (may be as short as 500 µs, but if supported allows one or more times less than 250 ms)
RFU 2 bits	Bits 14-15	

Table 99 — Battery Capabilities Word 7 (BCW7) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Fixed BAP PIE Listen Time 4 bits	Bits 0-3	0000: 500 µs (or listen time does not apply, for example for a BAP TAG that supports only **low power listen**) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s or greater 1111: Continuous These times may be the closest approximation to that actually used.
Fixed BAP PIE Sleep Or Low Power Listen Time 4 bits	Bits 4-7	0000: 500 µs (or does not apply, for example a BAP Tag that supports **listen** but not **sleep** or **low power listen**) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: 16 s or greater These times may be the closest approximation to that actually used.
RFU	Bits 8-15	

Table 100 — Battery Capabilities Word 8 (BCW8) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Fixed Duty Cycle Manchester Hibernate Listen Time 4 bits	Bits 0-3	0000: 500 µs (or does not apply, for example for a Tag that does not support Manchester) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s or greater 1111: Continuous These times may be the closest approximation to that actually used.
Fixed Duty Cycle Manchester Deep Hibernate (not listening) Time 4 bits	Bits 4-7	0000: 500 µs (or does not apply, for example for a Tag that does not support Manchester, or for a Manchester Tag that does not duty cycle its Hibernation) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s

Bit meaning	Bit assignments	Behaviour
		1111: 16 s These times may be the closest approximation to that actually used
Programmable Duty Cycle BAP PIE Shortest Listen and Sleep / Low Power Listen Time Supported (BPIE_Shortest_Listen) field 4 bits The BPIE_LPM field in BCW1 indicates if duty cycled BAP PIE is supported. The BPIE_DCP flag in BCW1 indicates if such duty cycling is programmable. The Pro_DC_Res flag of BCW6 indicates if high resolution (< 250 ms) duty cycling is supported. The best (minimum) resolution of programmable duty cycle BAP PIE listen and low power listen / sleep times are indicated by the BPIE_Shortest_Listen field here in BCW8. If the user attempts to program a time shorter than specified here, then the Tag shall set itself to the shortest time it does support.	Bits 8-11	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001-1111: Not used
Programmable Duty Cycle BAP PIE Longest Listen Time Supported (BPIE_Longest_Listen) field 4 bits This field indicates the longest programmable listen time supported by the Tag. If an attempt is made to program a longer time, the Tag shall set itself to the longest time it does support.	Bits 12-15	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous

Table 101 — Battery Capabilities Word 9 (BCW9) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Programmable Duty Cycle BAP PIE Longest Sleep or Low Power Listen Time Supported (BPIE_Longest_Sleep) field 4 bits If a user attempts to set a longer sleep time, the Tag will set itself to the longest it does support.	Bits 0-3	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous
Programmable Duty Cycle Manchester Shortest Listen Time Supported (Man_Shortest_Listen) field 4 bits	Bits 4-7	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001-1111: Not used
RFU	Bits 8-15	

Table 102 — Battery Capabilities Word 10 (BCW10) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Programmable Duty Cycle Manchester Shortest Listen and Deep Hibernate Time Supported (Man_Shortest_Deep) field 4 bits	Bits 0-3	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001-1111: Not used
Programmable Duty Cycle Manchester Longest Listen Supported (Man_Longest_Listen) field 4 bits	Bits 4-7	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous

Bit meaning	Bit assignments	Behaviour
Programmable Duty Cycle Manchester Longest Deep Hibernate Supported (Man_Longest_Deep) field 4 bits If a user attempts to set a longer sleep time, the Tag will set itself to the longest it does support.	Bits 8-11	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous
RFU 3 bits	Bits 12-15	

The Battery Settings Words (BSWs) for this version are next shown. These structures with initial settings are duplicated in the Battery Default Settings Words (BDSWs), which are copies of the initial BSWs stored at the end of the Capabilities File.

Table 103 — Battery Settings Word 1 (BSW1) with TCRS Version Code = 1.

Bit meaning	Bit assignments	Behaviour
Battery Working Temperature Range (BWTR) Field 2 bits	Bits 0-1	00: RFU for TBD restricted temp range 01: Nominal temp range -25 to +40 C 10: Extended temp range -40 to +65 C 11: Unknown
Storage Temperature Range (StoTR) field Storage Temperature Range is the range over which the product may be stored for a period of time equal to its typical service life without the battery losing more than 30% of its capacity or typical lifetime.	Bits 2-3	00: TBD 01: TBD 10: TBD 11: Unknown
Safety Temperature Range (SafTR) field 2 bits Safety Temperature Range is the range over which the product will not exhibit battery or other breakdown that could be hazardous.	Bits 4-5	00: TBD 01: TBD 10: TBD 11: Unknown
Battery Chemistry (BatChem) field 4 bits Battery chemistry partially informs the user as to whether the product has any special disposal issues. The Special Battery Disposal Field provides additional information and warnings.	Bits 6-9	0000: Carbon Zinc primary 0001: Lithium Manganese Dioxide primary 0010: Zinc Manganese Dioxide primary 0011: Lithium Thionyl Chloride primary 0100: Zinc Air primary 0101: Lithium Fluoride primary 0110: NiCad rechargeable 0111: NiMH rechargeable 1000: Lithium ion rechargeable 1001-1101 RFU 1110: Unknown 1111: Any other known battery chemistry
Battery Life (BatLife) field 3 bits Battery life is an approximate estimate (not a guarantee) of typical service life for primary batteries, or charge life for rechargeable, under typical operating conditions. This estimate is understood to vary under different use cases.	Bits 10-12	000: <1 month 001: 1-2 months 010: 2-4 months 011: 4-12 months 100: 1- 2 years 101: 2 – 5 years 110: > 5 years 111: Unknown
Battery Primary / Secondary (BP/S) Flag 1 bit	Bit 13	0: Battery is primary only. 1: Battery is rechargeable.
RFU	Bits 14-15	

Table 104 — Battery Settings Word 2 (BSW2) with TCRS Version Code = 1.

Bit meaning	Bit assignment	Behaviour
Special Battery Disposal (SBD) Field	Bits 0-3	0000: No special disposal requirements 0001 - 1110: RFU special disposal requirements do apply 1111: Unknown
Battery State	Bits 4-6	000: Battery monitoring is not supported 001: Battery < 10% life 010: 10% < Battery Life < 25% 011: 25% < Battery Life < 50% 100: 50% < Battery Life < 75% 101: 75% < Battery Life < 100% 110: RFU 111: RFU
Tag INACT_T Control field 3 bits Note: A BAP Tag that is programmed to INACT_T greater than 500 ms shall apply a maximum of 500 ms delay to the start of its inventory persistence timers when operating in BAP PIE mode.	Bits 7-9	000: 50 ms 001: 100 ms 010: 250 ms 011: 500 ms 100: 1 s 101: 2 s 110: 4 s 111: 8 s The INACT_T Status field (BCW6) indicates if this field may be used. If not programmable, these bits are Don't Care and this timeout (if used) is manufacturer fixed. The accuracy of this field shall be at least +/- 40% over the nominal temperature and at least +/- 50% over the extended temperature range (if supported).
(Selective) Global Timeout Control Field 3 bits The Tag may support a Global Timeout, and if so it shall be either of the Selective Global Timeout (refreshes on commands that have Tag manufacturer defined validity checks) or the regular Global Timeout (does not refresh). BAP Tags support at least one of INACT_T and Global Timeout. Tags that support INACT_T and a	Bits 10-12	000: 100 ms 001: 200 ms 010: 500 ms 011: 1 s 100: 4 s 101: 16 s 110: 32 s 111: Disabled (does not timeout) The Global Timeout Status field (BCW6) indicates if this field may be used. If not

Bit meaning	Bit assignment	Behaviour
programmable Global Timeout may have their Global Timeout disabled. Tags that support only a programmable Global Timeout shall not accept the “Disable” programming (their maximum Global Timeout is 32 seconds).		programmable, these bits are Don’t Care and this timeout (if used) is manufacturer fixed. The accuracy of this field shall be at least +/- 40% over the nominal temperature and at least +/- 50% over the extended temperature range (if supported).
RFU 3 bits	Bits 13-15	

Table 105 — Battery Settings Word 3 (BSW3) with TCRS Version Code = 1.

Bit meaning	Bit Assignment	Behaviour
Programmable Duty Cycle BAP PIE Listen Time (Pro_DC_PIE_LT) field 4 bits The BPIE_LPM field in BCW1 indicates if BAP PIE is continuous or duty cycled. The BPIE_DCP flag in BCW1 indicates if the duty cycle of BAP PIE is fixed or programmable. The Pro_DC_Res flag in BCW6 indicates the supported general resolution of programmable duty cycling (High < 250 ms, Low >= 250 ms). The BPIE_Shortest_Listen field in BCW8 indicates the shortest time < 250 ms that the Tag supports. If a user attempts to program a time shorter than supported, the Tag will set itself to the shortest time it does support. The BPIE_Longest_Listen field in BCW8 indicates the longest listen time that the Tag supports. If a user attempts to program a time longer than supported, the Tag will set itself to the longest listen time it does support.	Bits 0-3	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous The accuracy of this field shall be at least +/- 40% over the nominal temperature range and at least +/- 50% over the extended temperature range (if supported).

Bit meaning	Bit Assignment	Behaviour
Programmable Duty Cycle BAP PIE Sleep Or Low Power Listen Time (Pro_DC_PIE_S) 4 bits The shortest **sleep** time is equal to the shortest allowed **listen** time. The BPIE_Longest_Sleep field in BCW9 specifies the longest sleep time the Tag supports (to maintain latency). If the user attempts to program a longer **sleep** time than supported, the Tag will set itself to the longest it does support.	Bits 4-7	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: 16 s The accuracy of this field shall be at least +/- 40% over the nominal temperature range and at least +/- 50% over the extended temperature range (if supported).
RFU	Bits 8-15	

Table 106 — Battery Settings Word 4 (BSW4) with TCRS Version Code = 1.

Bit meaning	Bit Assignment	Behaviour
Manchester Hibernate Listen field (Man_Hib_Listen field) The Hib_DC flag in BCW2 indicates if the Hibernate modes can be duty cycled. The Hib_DC_Prog flag in BCW2 indicates if the duty cycles of the supported Hibernation modes are fixed or programmable. The Pro_DC_Res flag (common to supported battery modes) in BCW6 indicates the supported general resolution of programmable duty cycling (High < 250 ms, Low >= 250 ms). The Man_Shortest_Listen field in BCW10 indicates the shortest time < 250 ms that the Tag supports. If a user attempts to program a time shorter than supported, the Tag will set itself to the shortest time it does support. The Man_Longest_Listen field in BCW10 indicates the longest time that the Tag supports for **hibernate** listen. If a user attempts to program a listen time longer than supported, the Tag will set itself to the longest time it does support. If the Tag itself is authorized to alter the duty cycle as specified by the Tag_DC_ConAbility flag of BCW1 and has done so as reported in the Tag_DC_Rep flag of this word, then the setting it uses for **listen** will be reported in this field.	Bits 0-3	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous The accuracy of this field shall be at least +/- 40% over the nominal temperature range and at least +/- 50% over the extended temperature range (if supported).
Programmable Duty Cycle Manchester Deep Hibernate (Man_Hib_Deep) time field 4 bits The minimum **deep hibernation** time is equal to minimum **listen** time. The Man_Longest_Deep field in BCW10 indicates the longest time that the Tag supports for **hibernate** listen. If a user attempts to program a time shorter or longer than supported, the Tag will set itself to the shortest or longest time it does support. If the Tag itself is authorized to alter the duty cycle as specified by the Tag_DC_ConAbility flag of BCW1 and has done so as reported in the Tag_DC_Rep flag of this word, then the setting it uses for **deep hibernate** will be reported in this field.	Bits 4-7	0000: 500 µs (or does not apply) 0001: 1 ms 0010: 2 ms 0011: 4 ms 0100: 8 ms 0101: 16 ms 0110: 32 ms 0111: 64 ms 1000: 128 ms 1001: 250 ms 1010: 500 ms 1011: 1 s 1100: 2 s 1101: 4 s 1110: 8 s 1111: Continuous

Bit meaning	Bit Assignment	Behaviour
		The accuracy of this field shall be at least +/- 40% over the nominal temperature range and at least +/- 50% over the extended temperature range (if supported).
Tag Duty Cycle Authorization flag (Tag_DC_Auth) flag 1 bit Interrogator writes this bit to allow Tag the option of altering duty cycle in the case of a low battery. The definition of "low battery" is determined by the manufacturer. If the Tag has the capability of adapting duty cycle for either improved response or to preserve battery life, this is reported in the Tag_DC_ConAbility flag in BCW2.	Bit 8	0: Tag is not allowed to alter duty cycle to save battery life (or does not apply) 1: Tag is allowed to alter duty cycle to preserve battery life
Tag Duty Cycle Report (Tag_DC_Rep) flag 1 bit Tag sets this bit if it has altered duty cycles, and also reports exact settings used in the Man_Hib_Listen and Man_Hib_Deep fields as appropriate.	Bit 9	0: Tag has not altered duty cycles (or does not apply) 1: Tag has altered duty cycles
RFU 6 bits	Bits 10-15	

The duty cycle controls as given above shall have minimum accuracy as follows:

+/- 40% over the nominal temperature range of -25 to +40 °C.

+/- 50% over the extended temperature range of -40 to +65 °C.

8 Sensor support

8.1 Applicability

In case an Interrogator or Tag supports any command, response or feature of Clause 8 then this Interrogator or Tag shall support all mandatory commands, responses or features and it may support all optional commands, responses or features of Clause 8.

In case an Interrogator or Tag does not support any command, response or feature of Clause 8 then Clause 8 does not apply for this device.

8.2 Overview

This clause describes an optional extension to Types C and D that adds sensor support. However, references to RFID air interface commands, variable names, and state names in this clause refer to Type C unless otherwise noted. An RFID Tag having sensor support must have a Real Time Clock (RTC) to support sensor operations. The RTC and its use are defined in 8.3.

Two classes of sensor are supported:

— **Simple Sensor:** A Simple Sensor is programmed at source and is not required to be user programmed. A Simple Sensor by its nature delivers its payload as a Simple Sensor Data (SSD) block, appended to the object-related UII if requested by the Interrogator (see 8.5) as a Tag is inventoried, and therefore does not require a separate dialogue to collect the sensor data.
The three fundamental characteristics of a Simple Sensor are:
  — the sensor data is appended to the UII during Tag arbitration if requested by the Interrogator,
  — the sensor does not need to be user programmed and
  — the sensor computes pass/fail based on its characteristics, however it may also provide more details (e.g. an 8 bit sensor value)
The SSD block includes information about:
  — the type of sensor, limited to a few basic environmental features
  — measurement spans
  — thresholds
  — alarm status, indicating pass / fail conditions

  The SSD block is read by any user and provides limited configuration capabilities of the Simple Sensor by authorized (password controlled) user only. See Annex N and Annex O for details.

  Simple Sensors are defined in 8.5.

— **Full Function Sensor:** Full Function Sensors provide greater flexibility than Simple Sensors, by:
  — supporting a greater variety of sensor types and measurement spans,
  — enabling thresholds to be set within a wider range
  — capturing and processing different types of data

  The Full Function Sensor senses and records a set of measurements, stores the measurements and delivers them to an Interrogator when instructed to do so. Access to the sensor data requires the Tag to first be singulated and then a dialogue conducted between the Tag and an Interrogator. A Full Function Sensor may if required be programmed by the user either once or on multiple occasions.

  Full Function Sensors are defined in 8.6.

Tags may be equipped with one or more sensors. If a Type C Tag has sensor support then bit 16_h (XI) of the UII memory shall be asserted and XPC_W1 shall be supported, see also 7.6.2.

If it has the capability, an RFID Tag may support a combination of Full Function Sensors and one Simple Sensors. Each sensor shall be fully compliant with the class of sensor.

8.3 Real Time Clock (RTC)

8.3.1 General

RFID Tags having sensor support shall have a 32-bit RTC with an LSB of 1 second and the RTC shall be used as the source of UTC timestamps for sensor related data. Timestamps shall be based on the UTC time epoch beginning at 1970-01-01 00:00:00. At configuration of a sensor, the RTC shall be set to the current 32-bit UTC time precise to 1 second.

8.3.2 Setting the RTC

The RTC shall be accessible via the RTC Address. The RTC Address shall be stored in the TID memory (memory bank 10_2) at memory word 28_h MSB first (see Figure 39). The RTC Address shall comprise 6 bits reserved for future use (RFU) followed by 2 bits identifying the memory bank (MB) where the RTC is stored, and a 24-bit address in non EBV format specifying the starting word of the RTC.

Table 107 — Structure of RTC Address

	RFU	MB	Word Address
# of bits	6	2	24
description	Reserved for future use	Memory bank selector	RTC starting word address

The RTC may be set using the *Write* or *BlockWrite* commands. Write operations may be restricted by the use of write locking with access password, write permalocking, or other means implemented by the RFID Tag. It shall not be permitted to write a UTC time of zero into the RTC nor set the RTC when an alarm condition is present other than the low battery alarm.

The current 32-bit UTC timestamp shall be transmitted by the Interrogator to set the RTC during sensor configuration and at other times to maintain time synchronisation.

The 32-bit UTC timestamp may be taken from the Interrogator if it supports UTC time as specified in this clause. Timestamps that are not in seconds and of a 32-bit format are not suitable. Timestamps that are based on local time are not acceptable and shall not be used.

If the Interrogator is not capable of providing a timestamp in the specified format, then this should be provided using some accurate network-based UTC time. Some services compliant with IEEE 1588 can deliver a 64-bit timestamp, where the 32 MSB bits identify time to a second. Services compliant with the internet Simple Network Time Protocol, as defined in IETF RFC 1769, might also be able to provide the time as a 32-bit value that can be directly used.

The current 32-bit UTC timestamp within the RTC may be obtained using the *Read* command.

8.3.3 *BroadcastSync* command (optional, for Type C)

Type C Interrogators and Tags may implement the *BroadcastSync* command shown in Table 108. *BroadcastSync* allows an Interrogator to provide current UTC time information to a population of Tags to allow for synchronisation of the RTC time within a Tag to that of the clock time within the Interrogator. Time shall be a 32-bit integer value representing UTC time with an LSB of one second. It shall not be permitted to write a UTC time of zero into the RTC nor set the RTC when an alarm condition is present other than the low battery alarm.

The *BroadcastSync* command includes a CRC-16 that is calculated over the first command-code bit to the last UTC Time bit. If a Tag receives a *BroadcastSync* with a valid CRC-16 it may update its internal time to that of the UTC Time. If the RTC provides sufficient accuracy then it may decide to ignore the *BroadcastSync* command. The *BroadcastSync* command may update the RTC regardless if the RTC is located within a memory region that is locked or permalocked. An Interrogator may issue a *BroadcastSync* command at any time although it is recommended to use the command prior to the start of an inventory session. There shall be no reply from a Tag in response to a *BroadcastSync* command and the Tag shall not make any state transitions as a result of a *BroadcastSync* command.

Table 108 — *BroadcastSync* command (optional)

	Command	**UTC Time**	**CRC-16**
# of bits	8	32	16
description	1101 0001	Current Time	

8.3.4 Time synchronisation

The RTC mechanism on an RFID Tag might not be able to maintain an accurate record of time. This is partly a factor of the cost of providing additional accuracy and the impact on temperature variation to which the RTC is exposed. If the RFID Tag implements sensor event records, then it shall also implement a time synchronisation record block to enable an application to reconstruct a more accurate time line. The information about this record is given in TID memory words $2D_h$, $2E_h$, and $2F_h$.

The number of 32-bit timestamps that are required to be stored on the RFID Tag may be kept to a minimum by using the ordinal number of the sensor sample to reduce the use of memory and air interface transmission.

The time synchronisation record block enables a UTC timestamp to be recorded against the most recent sample count value. Each recorded time synchronisation record shall comprise, in sequence:

- The prevailing time as recorded by the RFID Tag. This may be a 32-bit RTC value maintained since the time of configuration (Record Type is set to 0 in Figure 39), or the 16-bit sample count value (Record Type is set to 1 in Figure 39). The 32-bit structure is recommended where time accuracy is important or where there is a large sample interval.

- The 32-bit UTC timestamp.

The number of time synchronisation records contained within the time synchronisation record block shall be specified in the Number of Records in word $2F_h$ in TID (see Figure 39). Every time the Tag decides to write a time synchronisation record, the Number of Records field is updated accordingly. The timestamp accuracy at the application level shall be achieved provided that a sufficient time synchronisation update rate is maintained. The time synchronisation update rate should define a maximum time interval between updates necessary to maintain timestamp accuracy as well as a minimum time interval between updates necessary to generate a time synchronisation record. When the allocated memory for the time synchronisation block becomes full no further records are possible, but no error shall be indicated.

8.3.4.1 Reconciling UTC with the 32-bit RTC time

As both the RTC and UTC timestamps have the same common datum, the UTC timestamp at configuration, it is possible to determine the extent of the difference in time at each point when time synchronisation took place. The extent of any difference might vary over the sample period and because of different causes, the RTC could be 'fast' or 'slow' without any pattern. Any time correction can only be applied more accurately close to the time of synchronisation, and be less certain at a point midway between two time synchronisation events.

8.3.4.2 Reconciling UTC with the 16-bit sample count

In this simpler approach, the best level of synchronisation that can be achieved is to write the UTC timestamp against the most recent sample count value. This means that time synchronisation is also a factor of the size of the sample interval. The larger the sample interval, the more likely that the UTC timestamp will represent a time that is between the most recent sample count value and the next sample count value. This provides a simple indication:

- that the RTC and UTC are within the same time period as determined by the sample interval, or

- if the UTC indicates an earlier time than the sample count, then the RTC is 'fast', or

- if the UTC indicates a later time than sample count + 1, then the RTC is 'slow'.

It is only when the synchronisation is at these outer limits that any inaccuracy can be corrected.

8.4 *HandleSensor* command (optional, for Type C)

To provide the means to support a broad range of different sensors types and hence different command sets, the *HandleSensor* command provides a transport mechanism to carry commands for intelligent sensors as a payload. The related response to the Interrogator shall carry the response of the sensor after receiving and processing the specific sensor command.

Tags shall interpret the payload of the *HandleSensor* command and execute the according sensor access command locally or forward it to its sensors for processing if required and transmit the response to the command within a well defined response time.

The *HandleSensor* command includes a 7 bit PortNr field which can be used to forward the included payload to a specific sensor without further inspection of the encapsulated sensor command. PortNr logically addresses a specific sensor attached to a Tag. Logical sensor addresses are assigned in ascending order starting at 0_2, e.g. 0_2, 1_2, 10_2, and so on.

The structure of *HandleSensor* shall be as shown in Table 109.

Tags execute *HandleSensor* only from the open or secured states. The CRC-16 is calculated over the first command code bit to the last handle bit.

An Interrogator may observe several possible outcomes from a *HandleSensor* command:

The *HandleSensor* command succeeds: After completing the *HandleSensor* command a Tag shall backscatter the reply shown in Table 110 comprising a header (a 0-bit), the optional sensor response to the encapsulated sensor command (applies only if the bit Response Expected has been set to value 1_2 in the *HandleSensor* command), the Tag's handle, and a CRC-16 calculated over the 0-bit to the last handle bit. The reply to *HandleSensor* shall begin within 20 ms.

The Tag encounters an error: The Tag shall backscatter an error code (see Annex I for error-code definitions and for the reply format).

Table 109 — Structure of *HandleSensor* command

	Command	PortNr	Payload Size	Payload	Response Expected	Response Length	RN	CRC-16
# of bits	8	7	Variable	Variable	1	Variable	16	16
description	11011001	Logical sensor address	Length of the Payload in bits (EBV-8)	Sensor command	1 = true 0 = false	Expected length of the sensor response in bits (EBV-8)	handle	

Table 110 — Tag reply to successful *HandleSensor* command

	Header	Response (optional)	RN	CRC-16
# of bits	1	Variable	16	16
Description	0	Sensor response	handle	

8.5 Simple Sensor

Although called "Simple Sensors", the devices are required to support features common to any type of sensor device. The Simple Sensor has to monitor the environmental characteristic for which it is designed, take samples at defined intervals, compare and process against criteria, and report its status.

The Simple Sensor transmits its data as a Simple Sensor Data (SSD) block which is appended to the Tag UII if requested by the Interrogator. The *Flex_Query* command in clause 7.4.1 and *Query_BAT* command in 7.5.4.3.1 contain the field SSD Resp to authorize Tags to append the SSD block after an *ACK* command. The SSD block shall consist of 32 bits or 48 bits depending on the type of Simple Sensor.

The specification of Simple Sensors also includes the provisions of the Sensor Directory System (SDS) as specified in sub-clause 8.6.

8.5.1 Type C and Simple Sensor

Simple sensors shall be implemented either as a memory mapped Simple Sensor or as a ported Simple Sensor. A memory mapped Simple Sensor is accessed via memory read and write operations to the RFID Tag. A ported Simple Sensor is accessed via dedicated sensor commands to the RFID Tag encapsulated in the *HandleSensor* transport command defined in 8.4.

The Simple Sensor Data (SSD) shall be accessible via the SSD Address. The SSD Address shall be stored in the TID memory (memory bank 10_2) at memory word 26_h MSB first (see Figure 39). The MSB of the SSD Address shall indicate the access method for the Simple Sensor. If the MSB contains a logical 0, then the access method is memory mapped and the remainder of the SSD Address is comprised of 3 bits reserved for future use (RFU), followed by 2 bits identifying the SSD size, followed by 2 bits identifying the memory bank (MB) where the SSD is stored, and a 24-bit address in non EBV format specifying the starting word of the SSD block. If the MSB contains a logical 1, then the access method is ported and the remainder of the SSD Address is comprised of 7 bits for a port number and 24 bits reserved for future use.

Table 111 — Structure of SSD Address for Memory Mapped Simple Sensor

	Access Method	RFU	SSD Size	MB	Word Address
# of bits	1	3	2	2	24
description	‘0’ Memory Mapped	Reserved for future use	00 = 32 bits 01 = 48 bits 10 = RFU 11 = RFU	Memory bank selector	SSD starting word address

Table 112 — Structure of SSD Address for Ported Simple Sensor

	Access Method	PortNr	RFU
# of bits	1	7	24
description	‘1’ Ported	Port Number	Reserved for future use

NOTE 1 The SSD Address is not necessarily required to be factory programmed and can be changed if the Tag has more than one sensor and it is decided to support another sensor from a certain time onwards.

NOTE 2 For Simple Sensors the SSD Address as well as the SSD itself may be write locked to the Interrogator, though alarms remain Tag/sensor writable.

NOTE 3 Tags supporting a Simple Sensor may indicate the necessity to request SSD to the Interrogator by using the Sensor Alarm flag (214h) of the XPC, see 7.6.2.

NOTE 4 Products developed before this document is officially published may transmit SSD at default without the requirement of explicitly requesting the Simple Sensor Data block.

NOTE 5 For Tags having more than one Simple Sensor, there is a mechanism provided to select the Active Simple Sensor, see 8.6.1.3.

Tags that support a Simple Sensor shall use the extended protocol control described in 7.6.2 to make this detail visible to the Interrogator. Within XPC_W1 the Simple Sensor (SS) bit 215_h shall be set to 1 to indicate the presence of a Simple Sensor, 0 else.

The SSD block comprises the sensor type, measure span, ACC, sampling regime, high in-range limit, low in-range limit, monitor delay, high out-of-range alarm delay, low out-of-range alarm delay, and alarms, all as defined in Annex N.

The contents of the Simple Sensor Data Block, see Annex N, shall be transmitted by the Tag subsequent to the UII after receiving an *ACK* in the course of an inventory round in which transmission of Simple Sensor Data is enabled by properly setting the SSD Resp field in *Flex_Query* or *Query_BAT*. SSD shall not be transmitted as default. The SSD block transmission as requested by the Interrogator by asserting the appropriate flag in the inventory command (see 8.5) is defined in Table 113.

The reply to the *ACK* command in case Simple Sensor Data transmission is requested is defined in Table 113. In general, Simple Sensor Data is appearing like an extension of the XPC data, which is the reason why it is never mentioned as explicit data in other clauses, however, only XPC is mentioned.

If Simple Sensor Data is transmitted as part of the reply to the *ACK* command, a dynamic PacketPC shall be used instead of the StoredPC available at UII memory location 10_h – 14_h, where the 5 most significant bits of the PacketPC included in the Tag response shall be used to indicate the length of the UII+XPC, but not extended by the SSD as shown in Table 113. In this context length always refers to 16-bit words.

Table 113 — Tag reply to a successful *ACK* command if SSD is requested

	Response	SSD	PacketCRC
# of bits	21 to 480 or 464 (see NOTE 1)	32 or 48	16
Description	PC word, XPC words, UII	Simple Sensor Data (according to Annex N)	CRC-16

NOTE 6 If a Tag does not support XPC_W2 then the maximum of UII + SSD is 480 bits. If a Tag supports XPC_W2 then the maximum length of UII + SSD to be backscattered is reduced by one word to accommodate the optional XPC_W2, so is 464 bits.

PacketCRC is a dynamic CRC to be calculated over all bits of the response to an *ACK* command received prior to the CRC-16.

8.5.1.1 Memory mapped Simple Sensor

A memory mapped Simple Sensor is accessed by an Interrogator using memory read and write operations to the RFID Tag.

The Simple Sensor may be configured using the *Write* or *BlockWrite* commands and writing the entire Simple Sensor Data block as defined in Annex N into the RFID Tag memory. Write operations to the Simple Sensor Data block may be restricted by the use of write locking with access password, write permalocking, or other means implemented by the RFID Tag. The RFID Tag may be implemented such that some parameter fields (e.g. sensor type, span, accuracy) are not writeable by an Interrogator and these bits shall be ignored by the RFID Tag when processing the *Write* or *BlockWrite* commands. The RFID Tag shall not permit setting alarm conditions when processing the *Write* or *BlockWrite* commands.

The Simple Sensor Data block may be obtained using the *Read* command.

Configuring the memory mapped Simple Sensor shall reset all the alarms within the Simple Sensor Data block, stop the RTC, and initialize the RTC to zero. The memory mapped Simple Sensor is disarmed when the RTC has a zero value. Writing the current time to the RTC shall then start the RTC running and arm the memory mapped Simple Sensor. Once armed, the sensor shall be monitored as configured and check for the event conditions required to declare an alarm condition. If an alarm condition other than low battery is declared, then the RTC shall be stopped and its current time retained as a timestamp for when the event occurred.

8.5.1.2 Ported Simple Sensor

A ported Simple Sensor is accessed by an Interrogator using dedicated sensor commands to the RFID Tag. The sensor commands are delivered to the Tag as the payload encapsulated within the *HandleSensor* command defined in 8.4. The ported Simple Sensor command set is defined in Annex O.

A number of records are defined for inclusion on a mandatory or optional basis to support the processing, encoding and subsequent diagnostics of a ported Simple Sensor. The ported Simple Sensor record structures are defined in Annex O.

8.6 Sensor Directory System and Full Function Sensors

The Sensor Directory System (SDS) is a simple directory system to assist in efficient accessing of sensors and sensor data. It applies to all sensor types, including Simple Sensors of both the memory mapped and ported sub-types, and to Full Function Sensors.

Full functions sensors may optionally be implemented on an 18000-6 Type C Tag. The Full Function Sensor characteristics and capabilities are given in the IEEE 1451.7 standard. Within XPC_W1 the Full Function Sensor (FS) bit 216_h shall be set to 1 to indicate the presence of a Full Function Sensor. These sensors are ported through a logical port. Sensor memory management is abstracted through the use of sensor records that are matched to dedicated sensor commands. These commands are transported to the sensor through the *HandleSensor* command defined in 8.4. The Full Function Sensor set-up (available sensors, ports, types, etc) of a Tag is specified in the Sensor Directory System (SDS) in regular Tag memory.

8.6.1 Sensor Access – General Approach

Sensor access is handled via a fixed address for an initial hook to sensor information and mapped memory positions via SDS. All the information needed to access a particular sensor is given its SDS Entry.

8.6.1.1 SDS Address

Tags equipped with one or more Full Function Sensors shall provide a 32-bit SDS Address pointing to the starting word address of a Sensor Directory System. The SDS Address shall be stored in the TID memory (memory bank 10_2) at memory word 22_h MSB first.

It shall comprise of 6 bits reserved for future use (RFU), followed by 2 bits identifying the memory bank (MB) where the SDS is stored, and a 24-bit field specifying the starting word address of the SDS in linear (non EBV) format.

Table 114 — Structure of SDS Address

	RFU	MB	Word Address
# of bits	6	2	24
description	Reserved for future use	Memory bank selector	SDS starting word address

The default value for the SDS Address shall be 0 for no sensor. Tags with one or more Full Function Sensors shall have a SDS Address ≠ 0.

Figure 39 shows a detailed layout of a Tag's TID memory if sensor access via SDS, Simple Sensor, and Tag Capabilities Reporting and Setting are supported.

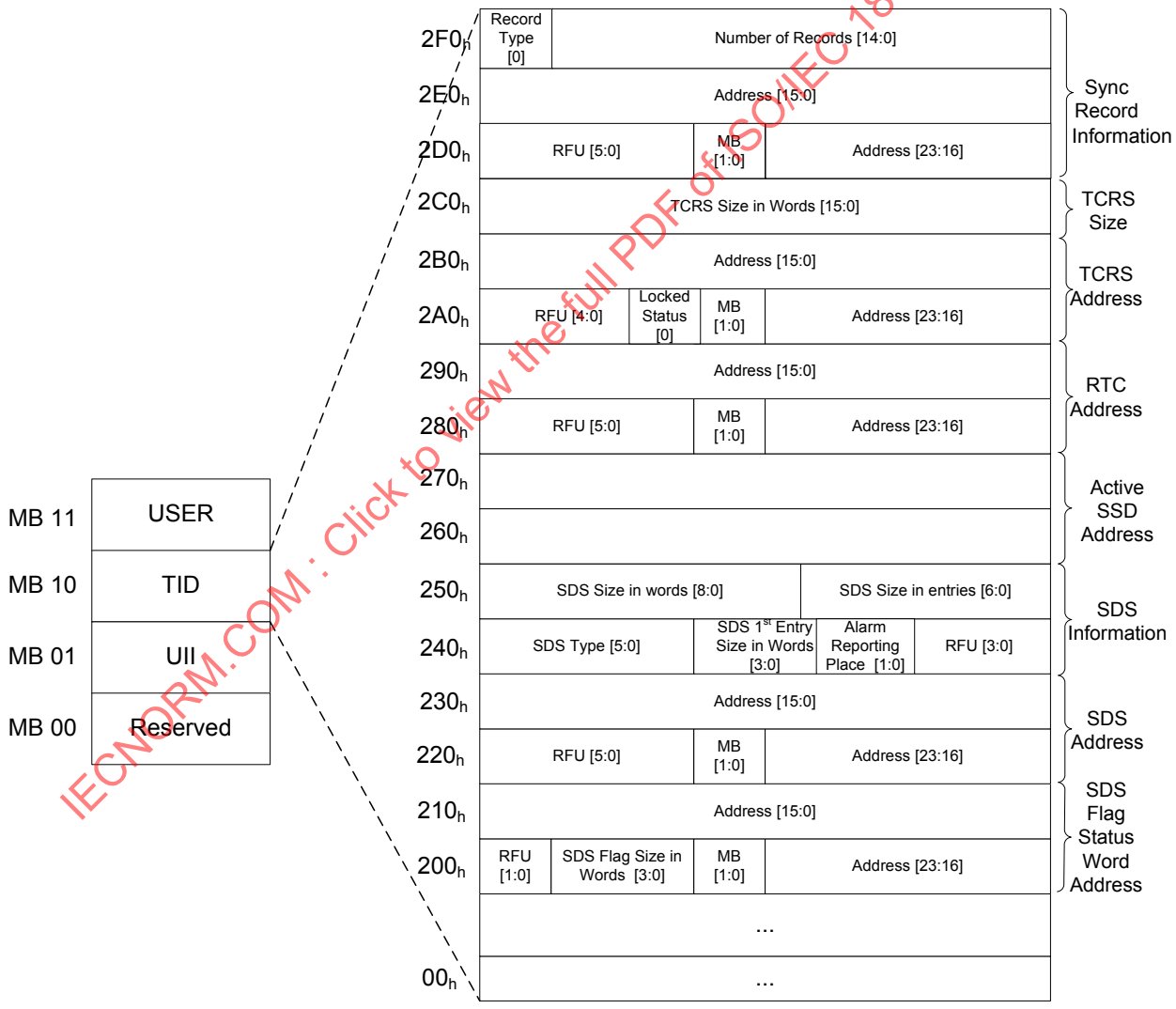


Figure 39 — Structure of TID memory

8.6.1.2 Sensor Directory System (SDS) pointers

The Sensor Directory System (SDS) is pointed to by two pointers located in TID, namely, the SDS Flag Status Word Address, and the SDS address, additionally, the SDS Information words are provided to access the SDS more efficiently. The structure of those pointers and the SDS Information words is given in Figure 39.

The SDS Flag Status Word Address is provided so that sensor alarms may be given in a separate memory location from the SDS Entries, in this way, the SDS entries can be completely hardcoded. The 24-bit address is in non EBV format. The SDS Flag Status Word is followed by one or more SDS Flag words.

Table 115 — SDS Flag Status Word structure

Word	Structure			
1	RFU [5:0]	Alarm Flag Latency [3:0]	BlockPermalock Status [1:0]	Lock Status [3:0]

The interpretation of the Alarm Flag Latency is given in Table 116.

Table 116 — Alarm Flag Latency field interpretation

Code	Meaning
0000	Sensor Alarms not supported
0001	Interrupt driven (near zero latency)
0010	Duty cycle driven
0011	Activation driven
0100	Poll driven
0101	Duty cycle, activation, and poll driven
0110	Sensor sample rate driven
0111 - 1111	RFU

The BlockPermalock field description is given in Table 117. This refers to the BlockPermalock status of the SDS Flag Status word and the SDS Flag words.

Table 117 — BlockPermalock Field interpretation

Code	Meaning
00	Not reported
01	Not BlockPermalocked
10	BlockPermalocked
11	RFU

The Lock Status field description is given in Table 118. This refers to the Lock status if the SDS Flag Status word and the SDS Flag words.

Table 118 — Lock Status Field interpretation

Code	Meaning
0000	Temp writable from **open** and **secured** states (Lock action 00)
0001	Permanently writable from **open** and **secured** states (Lock action 01)
0010	Temp writable from **secured** but not from **open** (Lock action 10)
0011	Temp not writable from any state (Lock action 11)
0100-1110	RFU
1111	Lock status is not reported

The structure of the SDS Flag word(s) is given in Figure 40.

Word	Structure															
1	SDS Entry 16 Alarm	.	.	.	.	.	.	.	.	.	.	SDS Entry 4 Alarm	SDS Entry 3 Alarm	SDS Entry 2 Alarm	SDS Entry 1 Alarm

Figure 40 — SDS Flag word(s) structure

More SDS Flag words are added as needed, and the number of words is given in the SDS Flag address word.

The SDS Information field makes the access to the SDS simpler and more efficient. It provides the SDS Type to enable future definitions; the only SDS Type defined in this standard is 000000_2 all other values are RFU. The First Entry Size in words is provided so that an Interrogator can read the SDS entries one by one, or with the aid of the SDS Size in words, read the whole SDS. The Alarm Reporting Place field interpretation is given in the Table 119.

Table 119 — Alarm Reporting Place field interpretation

Code	Meaning
00	Alarms Not reported
01	Alarms Reported in SDS Flag word only
10	Alarms Reported in SDS Entry only
11	Alarms Reported in SDS Flag word and SDS Entry

8.6.1.3 SDS Entry Record

The SDS Entry Record Structure is shown in Figure 41.

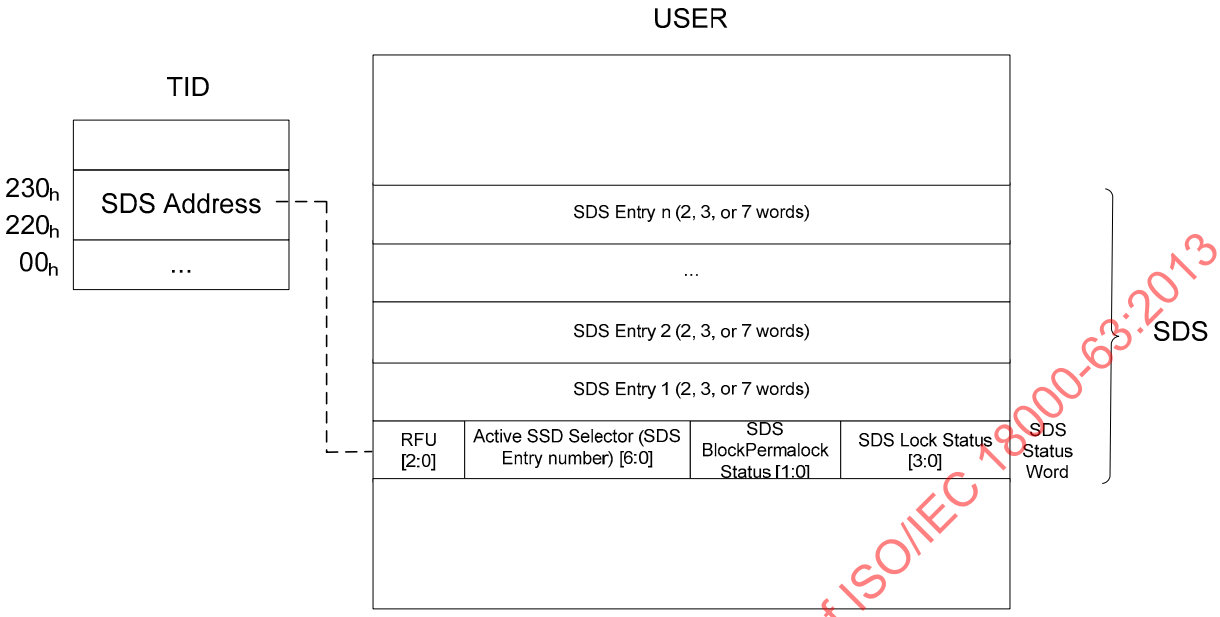


Figure 41 — SDS

The first word of the SDS is a status word. The structure of the SDS Status word is given in Figure 41. Within the SDS Status Word of Fig. 87, the SDS BlockPermalock Status field bits are as given in Table 117, and the SDS Lock Status field bits are as given in Table 118. For a Tag that has multiple Simple Sensor capabilities, writing the SDS entry into the “Active SSD Selector” field selects the SSD block that is backscattered after an *ACK* command (if so authorized by the Interrogator). This also routes the SSD block to the SSD address (if memory mapped) or updates the port number of the Simple Sensor (if ported). Since an SDS entry may correspond to a Simple Sensor or a Full Function Sensor, the Tag must backscatter an error code when the “Active SSD Selector” field is written with a Full Function Sensor Entry number. Additionally, the Tag must update the SSD size field accordingly.

Each sensor (Simple of Full Function) must have an entry in the SDS (with the exception of Tags with only one Simple Sensor since the information needed to access it is given in the TID pointers see Figure 39).

8.6.1.3.1 SDS Entry for a Memory Mapped Simple Sensor

The structure for this kind of entry is shown in Table 120.

Table 120 — SDS Entry for Memory Mapped Simple Sensors (3 words always)

Words	Structure				
1 (Lowest Memory Location)	Sensor Access Method [1:0]	Standard Identifier [5:0]	Sensor Type [3:0] (Annex N)	Sensor Alarm [0]	Next Entry Size in Words [2:0]
2	RFU [5:0]	MB [1:0]	SSD Address [23:16]		
3 (Highest Memory Location)	SSD Address [15:0]				

The interpretation of the Sensor Access Method is given in Table 121.

Table 121 – Sensor Access Method Interpretation

Code	Meaning
00	Simple Sensor Memory Mapped
01	Simple Sensor Ported
10	RFU
11	Full Function Sensor Ported

The interpretation of the Standard Identifier field is given in Table 122.

Table 122 — Standard Identifier Interpretation

Code	Meaning
000000	RFU
000001	IEEE 1451.7
000010	ISO/IEC 18000 6 Annex N
000011	ISO/IEC 18000 6 Annex O
000100~111111	RFU

Depending on the choice of the manufacturer, the Sensor Alarm field may or may not contain the alarm information (see Table 119). The 24-bit SSD address is not in EBV format.

8.6.1.3.2 SDS Entry for a Ported Simple Sensor

The structure for this kind of entry is shown in Table 123.

Table 123 — SDS Entry for Ported Simple Sensors (2 words always)

Words	Structure				
1 (Lowest Memory Location)	Sensor Access Type [1:0]	Standard Identifier [5:0]	Sensor Type [3:0] Annex N	Sensor Alarm [0]	Next Entry Size in Words [2:0]
2 (Highest Memory Location)	Port Number [6:0]		RFU [8:0]		

Port Number logically addresses a specific sensor attached to a Tag. Logical sensor addresses are assigned in ascending order beginning with 0000000. The number of sensors attached to a Tag can be determined from the number of SDS Entries field in the SDS information words.

8.6.1.3.3 SDS Entry for a Ported Full Function Sensor

The structure for this kind of entry is shown in Table 124.

Table 124 — SDS Entry for Full Function Sensors (3 or 7 words)

Words	Structure					
1	Sensor Access Type [1:0]	Standard Identifier [5:0]	Sensor Type [6:0] (See NOTE 1)			Sensor Alarm [0]
2	Next Entry Size in Words [2:0]	Port Number [6:0]			TEDS Type [2:0] (see NOTE 1)	AI Security Function Code [2:0] (see NOTE 1)
3	AI Security Indicator [0] (see NOTE 1)	Sensor Security Indicator [1:0] (see NOTE 1)	Sensor Security Function Code [2:0] (see NOTE 1)	Authentication Encryption Function Code [2:0] (see NOTE 1)	5 bits Units Extension [4:0] (see NOTE 1)	RFU [1:0]
4 (optional)	Sensor ID [63:48] (see NOTE 1)					
5 (optional)	Sensor ID [47:32] (see NOTE 1)					
6 (optional)	Sensor ID [31:16] (see NOTE 1)					
7 (optional)	Sensor ID [15:0] (see NOTE 1)					

NOTE These fields come from the IEEE 1451.7 standard.

Annex A
(normative)

Extensible bit vectors (EBV)

An *extensible bit vector* (EBV) is a data structure with an extensible data range.

An EBV is an array of *blocks*. Each block contains a single extension bit followed by a specific number of data bits. If B represents the total number of bits in one block, then a block contains B – 1 data bits. Although a general EBV may contain blocks of varying lengths, Tags and Interrogators manufactured according to this International Standard shall use blocks of length 8 bits (EBV-8).

The data value represented by an EBV is simply the bit string formed by the data bits as read from left-to-right, ignoring the extension bits.

Tags and Interrogators shall use the EBV-8 word format specified in Table A.1.

Table A.1 — EBV-8 word format

	0	0	0000000				
	1	0	0000001				
$2^7 - 1$	127	0	1111111				
2^7	128	1	0000001	0	0000000		
$2^{14} - 1$	16383	1	1111111	0	1111111		
2^{14}	16384	1	0000001	1	0000000	0	0000000

Because each block has 7 data bits available, the EBV-8 can represent numeric values between 0 and 127 with a single block. To represent the value 128, the extension bit is set to 1 in the first block, and a second block is appended to the EBV-8. In this manner, an EBV-8 can represent arbitrarily large values.

This International Standard uses EBV-8 values to represent memory addresses and Mask Lengths.

Annex B
(normative)

State-transition tables

B.1 Contents

State-transition Tables in sections B.2.1 to B.4.10 shall define a Tag's response to Interrogator commands for the different Tag types as follows:

1. Passive PIE: see B.2

2. BAP PIE: see B.3

3. BAP Manchester: see B.4

The term "handle" used in the state-transition tables is defined in 6.4.2.4.5; error codes are defined in Annex I; "slot" is the slot-counter output shown in Figure 19 and detailed in Annex J; "–" in the "Action" column means that a Tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

Within the tables, passive states and passive states shared by BAP are in lower case, whereas specifically BAP states have capitalized first letters.

B.2 State transition tables for passive

B.2.1 Present state: Ready

Table B.1 — Ready state-transition table

Command	Condition	Action	Next State
Query [a]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**ready**
QueryRep	all	–	**ready**
QueryAdjust	all	–	**ready**
ACK	all	–	**ready**
NAK	all	–	**ready**
Req_RN	all	–	**ready**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**ready**
Write	all	–	**ready**
Kill	all	–	**ready**
Lock	all	–	**ready**
Access	all	–	**ready**

Command	Condition	Action	Next State
BlockWrite	all	–	**ready**
BlockErase	all	–	**ready**
BlockPermalock	all	–	**ready**
Invalid [b]	all	–	**ready**

[a] *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

[b] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

B.2.2 Present state: Arbitrate

Table B.2 — Arbitrate state-transition table

Command	Condition	Action	Next State
Query [a,b]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**ready**
QueryRep	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
	slot<>0 after decrementing slot counter	–	**arbitrate**
QueryAdjust [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	all	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
Erase	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
Invalid [c]	all	–	**arbitrate**

[a] *Query* starts a new round and may change the session.

[b] *Query* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.2.3 Present state: Reply

Table B.3 — Reply state-transition table

Command	Condition	Action	Next State
Query [a,b]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
QueryRep	all	–	**arbitrate**
QueryAdjust [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	valid RN16	see Table 14	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	§	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
Invalid [c]	all	–	**reply**

[a] *Query* starts a new round and may change the session.
[b] *Query* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.
[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.2.4 Present state: Acknowledged

Table B.4 — Acknowledged state-transition table

Command	Condition	Action	Next State
Query [a,b]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid RN16	see Table 14	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
Invalid [c]	all	–	**acknowledged**

[a] *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

[b] As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.2.5 Present state: Open

Table B.5 — Open state-transition table

Command	Condition	Action	Next State
Query [a,b]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid handle	see Table 14	**open**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	valid handle & valid memory access	backscatter data and handle	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Write	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**open**
	invalid handle	–	**open**
Lock	all	–	**open**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**open**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockErase	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockPermalock	all	–	**open**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**open**

Command	Condition	Action	Next State
	Otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.
b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.
c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.
d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

B.2.6 Present state: Secured

Table B.6 — Secured state-transition table

Command	Condition	Action	Next State
Query [a,b]	slot=0; matching **inventoried** [b] & **SL** flags	backscatter new RN16; transition **inventoried** [b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried** [b] & **SL** flags	transition **inventoried** [b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	Otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	All	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	All	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid handle	see Table 14	**secured**
	invalid handle	–	**arbitrate**
NAK	All	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	valid handle & valid memory access	backscatter data and handle	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Write	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**secured**
	invalid handle	–	**secured**
Lock	valid handle & valid lock payload	backscatter handle when done	**secured**
	valid handle & invalid lock payload	backscatter error code	**secured**
	invalid handle	–	**secured**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**secured**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**

Command	Condition	Action	Next State
BlockErase	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockPermalock	valid handle, valid payload, & Read/Lock = 0	backscatter permalock bits and handle	**secured**
	valid handle, invalid payload, & Read/Lock = 0	backscatter error code	**secured**
	valid handle, valid payload, & Read/Lock = 1	backscatter handle when done	**secured**
	valid handle, invalid payload, & Read/Lock = 1	backscatter error code	**secured**
	invalid handle	–	**secured**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**secured**
	Otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.
b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.
c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.
d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively, or any other command either not recognized or not executable by the Tag.

B.2.7 Present state: Killed

Table B.7 — Killed state-transition table

Command	Condition	Action	Next State
Query	all	–	**killed**
QueryRep	all	–	**killed**
QueryAdjust	all	–	**killed**
ACK	all	–	**killed**
NAK	all	–	**killed**
Req_RN	all	–	**killed**
Select	all	–	**killed**
Read	all	–	**killed**
Write	all	–	**killed**
Kill	all	–	**killed**
Lock	all	–	**killed**
Access	all	–	**killed**
BlockWrite	all	–	**killed**
BlockErase	all	–	**killed**
BlockPermalock	all	–	**killed**
Invalid [a]	all	–	**killed**

a "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

B.3 State transition tables for BAP PIE

B.3.1 Present state: sleep

Table B.8 — sleep state-transition table

Command	Condition	Action	Next State
N/A	**Sleep** timer expires	–	**listen** or **stateful listen**
N/A	RF detected by optional **sleep** mode detector	–	**battery ready** or **stateful battery ready**

B.3.2 Present state: low power listen

Table B.9 — low power listen state-transition table

Command	Condition	Action	Next State
N/A	RF detected by **low power listen** mode detector	–	**battery ready** or **stateful battery ready**

B.3.3 Present state: listen or stateful listen

Table B.10 — listen or stateful listen state-transition table

Command	Condition	Action	Next State
N/A	RF detected within time LT	–	**battery ready** or **stateful battery ready**
N/A	No RF detected within time LT	–	**stateful sleep** or **stateful low power listen**

B.3.4 Present state: stateful sleep or stateful low power listen

Table B.11 — stateful sleep or stateful low power listen state-transition table

Command	Condition	Action	Next State
N/A	**sleep** timer expires	–	**listen** or **stateful listen**
N/A	Persistence timers all expire	–	**sleep** or **low power listen**
N/A	RF detected by optional **sleep** mode detector or **low power listen** mode detector	–	**battery ready** or **stateful battery ready**

B.3.5 Present state: battery ready

Table B.12 — battery ready state-transition table

Command	Condition	Action	Next State
Query [a], *Flex_Query* [d]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**battery ready**
QueryRep	all	–	**battery ready**
QueryAdjust	all	–	**battery ready**
ACK	all	–	**battery ready**
NAK	all	–	**battery ready**
Req_RN	all	–	**battery ready**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Read	all	–	**battery ready**
Write	all	–	**battery ready**
Kill	all	–	**battery ready**
Lock	all	–	**battery ready**
Access	all	–	**battery ready**
BlockWrite	all	–	**battery ready**
BlockErase	all	–	**battery ready**
BlockPermalock	all	–	**battery ready**
HandleSensor	all	–	**battery ready**
BroadcastSync	all	–	**battery ready**
INACT_T [c] or (Selective) Global Timeout	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
INACT_T [c] or (Selective) Global Timeout	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
Invalid [b]	all	–	**battery ready**

[a] *Query* starts a new round and may change the session. *Query* also instructs a tag to load a new random value into Its Slot Counter.

[b] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the tag.

[c] Details about these timers can be found in sub-clauses 7.3.2.2 and 7.3.2.3.

[d] Assumes Tag Type Select field matches criteria, otherwise this command is ignored.

B.3.6 Present state: Arbitrate

Table B.13 — Arbitrate state-transition table

Command	Condition	Action	Next State
Query [a,b], *Flex_Query* [d]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**battery ready**
QueryRep	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
	slot<>0 after decrementing slot counter	–	**arbitrate**
QueryAdjust [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	all	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**

Command	Condition	Action	Next State
Select	All	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Read	All	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
Erase	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
HandleSensor	all	–	**arbitrate**
BroadcastSync	all	–	**arbitrate**
INACT_T or (Selective) Global Timeout	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
INACT_T or (Selective) Global Timeout	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
Invalid [c]	all	–	**arbitrate**

a *Query* starts a new round and may change the session.
b *Query* and *QueryAdjust* instruct a tag to load a new random value into its slot counter.
c "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the tag.
d Assumes tag type select field matches criteria, otherwise this command is ignored.

B.3.7 Present state: Reply

Table B.14 — Reply state-transition table

Command	Condition	Action	Next State
Query[a,b], *Flex_Query*[d]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**battery ready**
QueryRep	All	–	**arbitrate**
QueryAdjust [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	valid RN16	see Table 14	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	All	–	**arbitrate**
Select	All	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Read	All	–	**arbitrate**
Write	All	–	**arbitrate**
Kill	All	–	**arbitrate**

Command	Condition	Action	Next State
Lock	All	–	**arbitrate**
Access	All	–	**arbitrate**
BlockWrite	All	–	**arbitrate**
BlockErase	All	–	**arbitrate**
BlockPermalock	All	–	**arbitrate**
HandleSensor	All	–	**arbitrate**
BroadcastSync	All	–	**reply**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
Invalid [c]	All	–	**reply**

[a] *Query* starts a new round and may change the session.
[b] *Query* and *QueryAdjust* instruct a tag to load a new random value into its slot counter.
[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the tag.
[d] Assumes tag type select field matches criteria, otherwise this command is ignored.

B.3.8 Present state: Acknowledged

Table B.15 — Acknowledged state-transition table

Command	Condition	Action	Next State
Query[a], *Flex_Query*[d]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**Battery ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**Battery ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**Battery ready**
ACK	valid RN16	see Table 14	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Read	all	–	**arbitrate**

Command	Condition	Action	Next State
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
HandleSensor	all	–	**arbitrate**
BroadcastSync	all	–	**acknowledged**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
Invalid [c]	all	–	**acknowledged**

[a] *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

[b] As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

[d] Assumes Tag Type Select field matches criteria, otherwise this command is ignored.

B.3.9 Present state: Open

Table B.16 — Open state-transition table

Command	Condition	Action	Next State
Query[a], *Flex_Query*[e]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**battery ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
ACK	valid handle	see Table 14	**open**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Read	valid handle & valid memory access	backscatter data and handle	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Write	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**

Command	Condition	Action	Next State
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**open**
	invalid handle	–	**open**
Lock	all	–	**open**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**open**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockErase	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockPermalock	all	–	**open**
HandleSensor	valid handle & valid command payload	backscatter header = 0, response code and handle when internal processing done	**open**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**open**
	invalid handle	–	**open**
BroadcastSync	all	–	**open**
INACT_T or (Selective) Global Timeout	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
INACT_T or (Selective) Global Timeout	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**open**
	Otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.

d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

e Assumes Tag Type Select field matches criteria, otherwise this command is ignored.

B.3.10 Present state: Secured

Table B.17 — Secured state-transition table

Command	Condition	Action	Next State
Query[a], *Flex_Query*[e]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid handle	see Table 14	**secured**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
Select	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	valid handle & valid memory access	backscatter data and handle	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Write	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**secured**
	invalid handle	–	**secured**
Lock	valid handle & valid lock payload	backscatter handle when done	**secured**
	valid handle & invalid lock payload	backscatter error code	**secured**
	invalid handle	–	**secured**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**secured**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockErase	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockPermalock	valid handle, valid payload, & Read/Lock = 0	backscatter permalock bits and handle	**secured**
	valid handle, invalid payload, & Read/Lock = 0	backscatter error code	**secured**
	valid handle, valid payload, & Read/Lock = 1	backscatter handle when done	**secured**
	valid handle, invalid payload, & Read/Lock = 1	backscatter error code	**secured**
	invalid handle	–	**secured**

Command	Condition	Action	Next State
HandleSensor	valid handle & valid command payload	backscatter header = 0, response code and handle when internal processing done	**secured**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**secured**
	invalid handle	–	**secured**
BroadcastSync	all	–	**secured**
INACT_T or (Selective) Global Timeout	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
INACT_T or (Selective) Global Timeout	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**secured**
	Otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.

d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

e Assumes Tag Type Select field matches criteria, otherwise this command is ignored.

B.3.11 Present state: Killed

Table B.18 — Killed state-transition table

Command	Condition	Action	Next State
Query, Flex_Query	all	–	**killed**
QueryRep	all	–	**killed**
QueryAdjust	all	–	**killed**
ACK	all	–	**killed**
NAK	all	–	**killed**
Req_RN	all	–	**killed**
Select	all	–	**killed**
Read	all	–	**killed**
Write	all	–	**killed**
Kill	all	–	**killed**
Lock	all	–	**killed**
Access	all	–	**killed**
BlockWrite	all	–	**killed**
BlockErase	all	–	**killed**
BlockPermalock	all	–	**killed**
HandleSensor	all	–	**killed**
BroadcastSync	all	–	**killed**
INACT_T or (Selective) Global Timeout	BAP Tag supports Battery Saver Mode	–	**killed**
INACT_T or (Selective) Global Timeout	BAP Tag does not support Battery Saver Mode	–	**killed**
Invalid [a]	all	–	**killed**

[a] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

B.4 State transition tables for BAP Manchester

B.4.1 Present state: Hibernate

Table B.19 — Hibernate state-transition table

Command	Condition	Action	Next State
Activation command preamble	*all*	–	**activation code check**

B.4.2 Present state: Activation code check

Table B.20 — Activation code check state-transition table

Command	Condition	Action	Next State
Short Activation command	Invalid Activation Mask	–	**stateful hibernate**
	Valid Activation Mask or Wildcard (Wildcard authorized)	Clear all timers, set **inventoried** flags to A, deassert **SL**	**battery ready**
Long Activation command	Invalid Activation Mask	–	**stateful hibernate**
	Valid Activation Mask or Wildcard (Wildcard authorized), matching flag criteria, Session Locking OFF	Clear all timers, set **inventoried** flags to A, deassert **SL**	**battery ready**
	Valid Activation Mask or Wildcard (Wildcard authorized), matching flag criteria, Session Locking ON or Interrogator locking in effect	Program activating Session timeout timer	**battery ready**
	Valid Activation Mask or Wildcard (Wildcard authorized), mismatching flag criteria	–	**stateful hibernate**
N/A	Flag timer expires	Set that flag to *A* as soon as current operations allow	**activation code check**

B.4.3 Present state: Stateful Hibernate

Table B.21 — Stateful Hibernate state-transition table

Command/Event	Condition	Action	Next State
Check flag timers	all timers expired	–	**hibernate**
	at least one flag timer active	–	**stateful hibernate**
Activation command preamble	all	–	**activation code check**
N/A	flag timer expires	Set that flag to *A* as soon as current operations allow	**stateful hibernate**

B.4.4 Present state: Battery Ready

Table B.22 — Battery Ready state-transition table

Command/Event	Condition	Action	Next State
Activation command preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT[a] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	Otherwise	–	**battery ready**
Query_BAT[a] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	Otherwise	–	**battery ready**
QueryRep	all	–	**battery ready**
QueryAdjust	all	–	**battery ready**
ACK	all	–	**battery ready**
NAK	all	–	**battery ready**
Req_RN	all	–	**battery ready**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**battery ready**
Read	all	–	**battery ready**
Write	all	–	**battery ready**
Kill	all	–	**battery ready**
Lock	all	–	**battery ready**

Command/Event	Condition	Action	Next State
Access	all	–	**battery ready**
BlockWrite	all	–	**battery ready**
BlockErase	all	–	**battery ready**
BlockPermalock	all	–	**battery ready**
Broadcast ID	all	–	**battery ready**
Next	all	–	**battery ready**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or **SL** flags, Override=0	–	**battery ready**
	Mismatching Interrogator ID (if Interrogator locking is in effect), mismatching activating session (if session locking is in effect), Override=0	–	**battery ready**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**battery ready**
(Selective) Global Timeout or INACT_T	all	Set active session flag to *A* and clear active session flag timeout timer, **SL→~SL**	**stateful hibernate**
HandleSensor	all	–	**battery ready**
BroadcastSync	all	–	**battery ready**
OpRegister Read/Write	all	–	**battery ready**
Invalid [b]	all	–	**battery ready**

[a] *Query_BAT* starts a new round and may change the session. *Query_BAT* also instructs a Tag to load a new random value into its slot counter.

[b] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

B.4.5 Present state: Arbitrate

Table B.23 — Arbitrate state-transition table

Command	Condition	Action	Next State
Activation command preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT[a,b] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	Otherwise	–	**battery ready**
Query_BAT[a,b] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	Otherwise	–	**arbitrate**
QueryRep	Matching Interrogator ID (if Interrogator locking is in effect), slot=0 after decrementing slot counter	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0 after decrementing slot counter	–	**arbitrate**
QueryAdjust[b]	Matching Interrogator ID (if Interrogator locking is in effect), slot=0	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0	–	**arbitrate**
	Otherwise	–	**arbitrate**
ACK	all	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**arbitrate**
Read	all	–	**arbitrate**

Command	Condition	Action	Next State
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
Erase	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
Broadcast ID	all	–	**arbitrate**
Next	all	–	**arbitrate**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or **SL** flags, Override=0	–	**battery ready**
	otherwise	–	**arbitrate**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**arbitrate**
(Selective) Global Timeout or INACT_T	all	Set active session flag to *A* and clear active session flag timeout timer, **SL→~SL**	**stateful hibernate**
HandleSensor	all	–	**arbitrate**
BroadcastSync	all	–	**arbitrate**
OpRegister Read/Write	all	–	**arbitrate**
Invalid [c]	all	–	**arbitrate**

[a] *Query_BAT* starts a new round and may change the session.

[b] *Query_BAT* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than *Query_BAT or Deactivate_BAT*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.4.6 Present state: Reply

Table B.24 — Reply state-transition table

Command	Condition	Action	Next State
Activation command preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT [a,b] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	otherwise	–	**battery ready**
Query_BAT [a,b] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried** & **SL** flags and mini-select	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried** & **SL** flags and mini-select	–	**arbitrate**
	Otherwise	–	**reply**
QueryRep	Matching Interrogator ID (if Interrogator locking is in effect)	–	**arbitrate**
	otherwise	–	**reply**
QueryAdjust [b]	Matching Interrogator ID (if Interrogator locking is in effect), slot=0	backscatter new RN16	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0	–	**arbitrate**
	Otherwise	–	**reply**
ACK	valid RN16	see Table 14, Backscatter SSD if authorized (see 8.5.1)	**acknowledged**
	invalid RN16	–	**reply**
NAK	Matching Interrogator ID (if Interrogator locking is in effect)	–	**arbitrate**
	Otherwise	–	**reply**
Req_RN	all	–	**arbitrate**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**reply**
Read	all	–	**arbitrate**

Command	Condition	Action	Next State
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
Broadcast ID	all	–	**reply**
Next	all	–	**reply**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or **SL** flags, Override=0	–	**battery ready**
	otherwise	–	**reply**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**reply**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
HandleSensor	all	–	**arbitrate**
BroadcastSync	all	–	**reply**
OpRegister Read/Write	all	–	**arbitrate**
Invalid [c]	all	–	**reply**

[a] *Query_BAT* starts a new round and may change the session.

[b] *Query_BAT* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than *Query_BAT or Deactivate_BAT*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.4.7 Present state: Acknowledged

Table B.25 — Acknowledged state-transition table

Command	Condition	Action	Next State
Activation command preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT[a] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags and mini-select	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**battery ready**
Query_BAT[a] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**arbitrate**
	otherwise	–	**acknowledged**
QueryRep	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**acknowledged**
QueryAdjust	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**acknowledged**
ACK	valid RN16	see Table 14, Backscatter SSD if authorized (see 8.5.1)	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	Matching Interrogator ID (if Interrogator locking is in effect)	–	**arbitrate**
	otherwise	–	**acknowledged**
Req_RN	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**

Command	Condition	Action	Next State
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**acknowledged**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
BlockPermalock	all	–	**arbitrate**
Broadcast ID	all	–	**acknowledged**
Next	valid RN16	Backscatter RN16	**stateful hibernate**
	invalid RN16	–	**acknowledged**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or **SL** flags, Override=0	–	**battery ready**
	otherwise	–	**acknowledged**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**acknowledged**
T_2 timeout	See Figure 16 and Table 13	–	**arbitrate**
HandleSensor	all	–	**arbitrate**
BroadcastSync	all	–	**acknowledged**
OpRegister Read/Write	all	–	**arbitrate**
Invalid [c]	all	–	**acknowledged**

[a] *Query_BAT* starts a new round and may change the session. *Query_BAT* also instructs a Tag to load a new random value into its slot counter.

[b] As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

[c] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than *Query_BAT or Deactivate_BAT*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.4.8 Present state: Open

Table B.26 — Open state-transition table

Command	Condition	Action	Next State
Activation Command Preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT[a] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags and mini-select	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**battery ready**
Query_BAT[a] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**arbitrate**
	otherwise	–	**open**
QueryRep	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**open**
QueryAdjust	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**open**
ACK	valid handle	see Table 14, Backscatter SSD if authorized	**open**
	invalid handle	–	**open**
NAK	Matching Interrogator ID (if Interrogator locking is in effect)	–	**arbitrate**
	otherwise	–	**open**
Req_RN	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**open**
Read	valid handle & valid memory access	backscatter data and handle	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Write	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**open**
	invalid handle	–	**open**
Lock	all	–	**open**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**open**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**

Command	Condition	Action	Next State
BlockErase	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockPermalock	all	–	**open**
Broadcast ID	all	–	**open**
Next	valid RN16_handle	Backscatter RN16_handle	**stateful hibernate**
	invalid RN16_handle	–	**open**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or **SL** flags, Override=0	–	**battery ready**
	otherwise	–	**open**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**open**
(Selective) Global Timeout or INACT_T	all	Set active session flag to *A* and clear active session flag timeout timer, **SL**→**~SL**	**stateful hibernate**
HandleSensor	valid handle & valid command payload	backscatter header = 0, response code and handle when internal processing done	**open**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**open**
	invalid handle	–	**open**
BroadcastSync	all	–	**open**
OpRegister Read/Write	all	–	**open**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**open**
	Otherwise valid commands, except *Req_RN* or *Query_BAT*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query_BAT* starts a new round and may change the session. *Query_BAT* also instructs a Tag to load a new random value into its slot counter.
b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.
c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.
d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than *Query_BAT* or *Deactivate_BAT*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

B.4.9 Present state: Secured

Table B.27 — Secured state-transition table

Command	Condition	Action	Next State
Activation command preamble	all	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
Query_BAT[a] (Short Activation or Long Activation with Session Locking OFF)	slot=0; matching **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags and mini-select	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**battery ready**
Query_BAT[a] (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), slot=0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**reply**
	Matching Interrogator ID (if Interrogator locking is in effect), slot<>0; matching activating session **inventoried**[b] & **SL** flags and mini-select	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A*	**arbitrate**
	otherwise	–	**secured**
QueryRep	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**secured**
QueryAdjust	Matching Interrogator ID (if Interrogator locking is in effect)	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
	otherwise	–	**secured**
ACK	valid handle	see Table 14, Backscatter SSD if authorized (see 8.5.1)	**secured**
	invalid handle	–	**secured**
NAK	Matching Interrogator ID (if Interrogator locking is in effect)	–	**arbitrate**
	otherwise	–	**secured**
Req_RN	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
Select (Short Activation or Long Activation with Session Locking OFF)	all	assert or de-assert **SL**, or set **inventoried** to *A* or B	**battery ready**
Select (Long Activation with Session Locking ON)	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session, or Target is **SL**	assert or de-assert **SL**, or set activating session **inventoried** to *A* or *B*	**battery ready**
	Otherwise	–	**secured**
Read	valid handle & valid memory access	backscatter data and handle	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Write	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**

Command	Condition	Action	Next State
Kill (see also Figure 23)	valid handle & valid nonzero kill password & Recom = 0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom <> 0	backscatter handle when done	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**secured**
	invalid handle	–	**secured**
Lock	valid handle & valid lock payload	backscatter handle when done	**secured**
	valid handle & invalid lock payload	backscatter error code	**secured**
	invalid handle	–	**secured**
Access (see also Figure 25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**secured**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockErase	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockPermalock	valid handle, valid payload, & Read/Lock = 0	backscatter permalock bits and handle	**secured**
	valid handle, invalid payload, & Read/Lock = 0	backscatter error code	**secured**
	valid handle, valid payload, & Read/Lock = 1	backscatter handle when done	**secured**
	valid handle, invalid payload, & Read/Lock = 1	backscatter error code	**secured**
	invalid handle	–	**secured**
Broadcast ID	all	–	**secured**
Next	valid RN16_handle	Backscatter RN16_handle	**stateful hibernate**
	invalid RN16_handle	–	**secured**
Deactivate_BAT	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried** or SL flags, Override=0	–	**battery ready**
	otherwise	–	**secured**
Multirate_Reset	all	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**secured**
(Selective) Global Timeout or INACT_T	all	Set active session flag to *A* and clear active session flag timeout timer, **SL→~SL**	**stateful hibernate**
HandleSensor	valid handle & valid command payload	backscatter header = 0, response code and handle when internal processing done	**secured**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**secured**
	invalid handle	–	**secured**
BroadcastSync	all	–	**secured**

Command	Condition	Action	Next State
OpRegister Read/Write	valid handle, read, valid reg ID and wordcount	backscatter read bit, handle, and data	**secured**
	valid handle, write, valid reg ID and wordcount	backscatter write bit, handle when done	**secured**
	valid handle, invalid reg ID or wordcount	backscatter read/write bit, error code	**secured**
	invalid handle	–	**secured**
Invalid [d]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**secured**
	Otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**

a *Query_BAT* starts a new round and may change the session. *Query_BAT* also instructs a Tag to load a new random value into its slot counter.
b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.
c As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom = 0.
d "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than *Query_BAT* or *Deactivate_BAT*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

B.4.10 Present state: Killed

Table B.28 — Killed state-transition table

Command	Condition	Action	Next State
Activation Command Preamble	**all**	–	**Killed**
Query_BAT	all	–	**Killed**
QueryRep	all	–	**Killed**
QueryAdjust	all	–	**Killed**
ACK	all	–	**Killed**
NAK	all	–	**Killed**
Req_RN	all	–	**killed**
Select	all	–	**killed**
Read	all	–	**killed**
Write	all	–	**killed**
Kill	all	–	**killed**
Lock	all	–	**killed**
Access	all	–	**killed**
BlockWrite	all	–	**killed**
BlockErase	all	–	**killed**
BlockPermalock	all	–	**killed**
Broadcast ID	all	–	**killed**
Next	all	–	**killed**
Deactivate_BAT	all	–	**killed**
Flag timer expires	all	Set that flag to *A* as soon as current operations allow	**killed**
(Selective) Global Timeout or INACT_T	all	–	**killed**
Multirate_Reset	all	–	**killed**
HandleSensor	all	–	**killed**
BroadcastSync	all	–	**killed**
OpRegister Read/Write	all	–	**killed**
Invalid [a]	all	–	**killed**

[a] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

Annex C
(normative)

Command-response tables

C.1 Contents

Command-response tables in sections C.2.1 to C.4.31 shall define a Tag's response to Interrogator commands for the different Tag types as follows:

1. Passive PIE: see C.2

2. BAP PIE: see C.3

3. BAP Manchester: see C.4

The term "handle" used in the state-transition tables is defined in 6.4.2.4.5; error codes are defined in Annex I; "slot" is the slot-counter output shown in Figure 19 and detailed in Annex J; "–" in the "Response" column means that a Tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

Within the tables, passive states and those shared with passive are in lower case, whereas specifically BAP states are indicated by a capitalized first letter.

C.2 Command response tables for passive

C.2.1 Command response: Power-up

Table C.1 — Power-up command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	power-up	–	**ready**
killed	all	–	**killed**

C.2.2 Command response: *Query*

Table C.2 — *Query*[a] command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**ready**

Starting State	Condition	Response	Next State
acknowledged, open, secured	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	all other conditions	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
	Otherwise	–	**ready**
killed	All	–	**killed**

[a] *Query* (in any state other than killed) starts a new round and may change the session; *Query* also instructs a Tag to load a new random value into its slot counter.

[b] As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

C.2.3 Command response: *QueryRep*

Table C.3 — *QueryRep* command-response table [a]

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	slot<>0 after decrementing slot counter	–	**arbitrate**
	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
reply	all	–	**arbitrate**
acknowledged, open, secured	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
killed	all	–	**killed**

[a] See C.2.18 for the Tag response to a *QueryRep* whose session parameter does not match that of the current inventory round.

C.2.4 Command response: *QueryAdjust*

Table C.4 — *QueryAdjust* [a] command-response table[b]

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply	slot<>0	–	**arbitrate**
	Slot=0	backscatter new RN16	**reply**
acknowledged, open, secured	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
killed	all	–	**killed**

[a] *QueryAdjust*, in the arbitrate or reply states, instructs a Tag to load a new random value into its slot counter.

[b] See C.2.18 for the Tag response to a *QueryAdjust* whose session parameter does not match that of the current inventory round.

C.2.5 Command response: *ACK*

Table C.5 — *ACK* command-response table

Starting State	Condition	Response	Next State
Ready	all	–	**ready**
Arbitrate	all	–	**arbitrate**
Reply	valid RN16;	backscatter {PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {000002, truncated UII, StoredCRC}	**acknowledged**
	invalid RN16	–	**arbitrate**
acknowledged	valid RN16;	backscatter {PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {000002, truncated UII, StoredCRC}	**acknowledged**
	invalid RN16	–	**arbitrate**
Open	valid handle;	backscatter { PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {000002, truncated UII, StoredCRC}	**open**
	invalid handle	–	**arbitrate**
Secured	valid handle;	backscatter { PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {000002, truncated UII, StoredCRC}	**secured**
	invalid handle	–	**arbitrate**
Killed	all	–	**killed**

C.2.6 Command response: *NAK*

Table C.6 — *NAK* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged, open, secured	all	–	**arbitrate**
killed	all	–	**killed**

C.2.7 Command response: *Req_RN*

Table C.7 — *Req_RN* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply	all	–	**arbitrate**
acknowledged	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
open	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
secured	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.8 Command response: *Select*

Table C.8 — *Select* command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	All	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**ready**
killed	All	–	**killed**

C.2.9 Command response: *Read*

Table C.9 — *Read* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter data and handle	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter data and handle	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.10 Command response: *Write*

Table C.10 — *Write* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.11 Command response: *Kill*

Table C.11 — *Kill*[1] command-response table

Starting State	Condition	Response	Next State
ready	All	–	**ready**
arbitrate, reply, acknowledged	All	–	**arbitrate**
open	valid handle & kill password=0	backscatter error code	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password & Recom=0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom<>0	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & kill password=0	backscatter error code	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password & Recom=0	backscatter handle when done	**killed**
	valid handle & valid nonzero kill password & Recom<>0	backscatter handle when done	**open**
	invalid handle	–	**secured**
killed	All	–	**killed**

NOTE 1 See also Figure 23

NOTE 2 As described in 6.4.2.11.3.4, if a Tag does not implement recommissioning then the Tag treats nonzero Recom bits as though Recom=0.

C.2.12 Command response: *Lock*

Table C.12 — *Lock* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	all	–	**open**
secured	valid handle & invalid lock payload	backscatter error code	**secured**
	valid handle & valid lock payload	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.13 Command response: *Access*

Table C.13 — *Access*[1] command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid access password	–	**arbitrate**
	valid handle & valid access password	backscatter handle	**secured**
	invalid handle	–	**open**
secured	valid handle & invalid access password	–	**arbitrate**
	valid handle & valid access password	backscatter handle	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

NOTE See also Figure 25

C.2.14 Command response: *BlockWrite*

Table C.14 — *BlockWrite* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.15 Command response: *BlockErase*

Table C.15 — *BlockErase* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.16 Command response: *BlockPermalock*

Table C.16 — *BlockPermalock* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	all	–	**open**
secured	valid handle, valid payload, & Read/Lock = 0	backscatter permalock bits and handle	**secured**
	valid handle, invalid payload, & Read/Lock = 0	backscatter error code	**secured**
	valid handle, valid payload, & Read/Lock = 1	backscatter handle when done	**secured**
	valid handle, invalid payload, & Read/Lock = 1	backscatter error code	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.2.17 Command response: T_2 timeout

Table C.17 — T_2 timeout command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	all	–	**arbitrate**
Reply, acknowledged	See Figure 16 and Table 13	–	**arbitrate**
open	all	–	**open**
secured	all	–	**secured**
killed	all	–	**killed**

C.2.18 Command response: Invalid command

Table C.18 — Invalid command table

Starting State	Condition	Response	Next State
ready[a]	all	–	**ready**
arbitrate[b]	all	–	**arbitrate**
reply[b]	all	–	**reply**
acknowledged[b]	all	–	**acknowledged**
open[b]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**open**
	otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**
secured[b]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**secured**
	otherwise valid commands, except *Req_RN* or *Query*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**
killed[a]	all	–	**killed**

[a] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

[b] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

C.3 Command response tables for BAP PIE

Same as in C.2, with the **ready** state replaced by **battery ready**, and the addition of *Flex_Query*, *BroadcastSync*, INACT_T, and (Selective) Global Timeout.

C.3.1 Command response: Flex_Query (optional for BAP PIE)

Table C.19 — *Flex_Query*[a,c] command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, reply	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**battery ready**
acknowledged, open, secured	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	all other conditions	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
	otherwise	–	**battery ready**
killed	all	–	**killed**

a *Flex_Query* (in any state other than killed) starts a new round and may change the session; *Flex_Query* also instructs a Tag to load a new random value into its slot counter.

b As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

c Tag checks the Tag Type Select field, if it matches the conditions in that field, Tag executes this command, otherwise, it ignores it.

C.3.2 Command response: INACT_T or Selective Global Timeout

Table C.20 — INACT_T[a] or Selective Global Timeout[b] command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, open, secured	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
battery ready, arbitrate, open, secured	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
killed	all	–	**killed**

a See clause 7.3.2.2 for details on the refresh of the INACT_T timer

b See clause 7.3.2.3 for details on the refresh of the Selective Global Timeout timer

C.3.3 Command response: Global Timeout

Table C.21 —Global Timeout command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, reply, acknowledged, open, secured	BAP Tag supports Battery Saver Mode	–	**stateful sleep** or **stateful low power listen**
battery ready, arbitrate, reply, acknowledged, open, secured	BAP Tag does not support Battery Saver Mode	–	**battery ready** or **stateful battery ready**
killed	all	–	**killed**

C.3.4 Command response: *HandleSensor*

Table C.22 — *HandleSensor* command-response table

Starting State	Condition	Response	Next State
battery ready	All	–	**battery ready**
arbitrate, reply, acknowledged	All	–	**arbitrate**
open, secured	valid handle & valid command payload	backscatter header = 0, response code[a] and handle when internal processing done	**open, secured**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**open, secured**
	invalid handle	–	**open, secured**
killed	All	–	**killed**

[a] Sensor specific output of *HandleSensor* command payload processing.

C.3.5 Command response: *BroadcastSync*

Table C.23 — *BroadcastSync* command-response table

Starting State	Condition	Response	Next State
hibernate,	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready, arbitrate, reply, acknowledged, open, secured	all	–	**battery ready, arbitrate, reply, acknowledged, open, secured**
killed	All	–	**killed**

C.4 Command Response Tables for Manchester

C.4.1 Command response: Power-up

Table C.24 — Power-up command-response table

Starting State	Condition	Response	Next State
OFF	power-up	–	**hibernate**
killed	all	–	**killed**

C.4.2 Command response: *QueryRep*

Table C.25 — *QueryRep* command-response table [a]

Starting State	Condition	Response	Next State
battery ready	All	–	**battery ready**
arbitrate	slot<>0 after decrementing slot counter	–	**arbitrate**
	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
reply	All	–	**arbitrate**
acknowledged, open, secured	All	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
killed	All	–	**killed**

[a] In Manchester with Long Activation and Interrogator locking in effect, the Tag checks first the Interrogator ID, if it matches, it proceeds with the command execution, otherwise, it ignores the command.

C.4.3 Command response: *QueryAdjust*

Table C.26 — *QueryAdjust* [a] command-response table[b]

Starting State	Condition	Response	Next State
battery ready	all	–	**battery ready**
arbitrate, reply	Slot<>0	–	**arbitrate**
	Slot=0	backscatter new RN16	**reply**
acknowledged, open, secured	all	transition **inventoried** from *A*→*B* or *B*→*A*	**battery ready**
killed	all	–	**killed**

[a] *QueryAdjust*, in the arbitrate or reply states, instructs a Tag to load a new random value into its slot counter.

[b] In Manchester with Long Activation and Interrogator locking in effect, the Tag checks first the Interrogator ID, if it matches, it proceeds with the command execution, otherwise, it ignores the command.

C.4.4 Command response: *ACK*

Table C.27 — *ACK* command-response table[a]

Starting State	Condition	Response	Next State
battery ready	All	–	**battery ready**
arbitrate	All	–	**arbitrate**
reply	valid RN16; no Simple Sensor functionality	backscatter {PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**acknowledged**
	valid RN16; Simple Sensor functionality (SS Resp =1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, SSD, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**acknowledged**
	valid RN16; Mobile RFID(XPC_W 1 212h=1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, MIIM content name, PacketCRC} or {000002, truncated UII, StoredCRC}	**acknowledged**
	invalid RN16	–	**reply**
acknowledged	valid RN16; no Simple Sensor functionality	backscatter {PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**acknowledged**
	valid RN16; Simple Sensor functionality (SS Resp =1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, SSD, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**acknowledged**
	valid RN16; Mobile RFID(XPC_W 1 212h=1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, MIIM content name, PacketCRC} or {000002, truncated UII, StoredCRC}	**acknowledged**
	invalid RN16	–	**arbitrate**
open	valid handle; no Simple Sensor functionality	backscatter { PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**open**
	valid handle; Simple Sensor functionality (SS Resp =1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, SSD, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**open**
	invalid handle	–	**open**
secured	valid handle; no Simple Sensor functionality	backscatter { PC, XPC_W1(if XI=1), XPC_W2(if XEB=1), UII, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**secured**
	valid handle; Simple Sensor functionality (SS Resp =1)	backscatter {PC, XPC_W1, XPC_W2(if XEB=1), UII, SSD, PacketCRC} or {00000_2, truncated UII, StoredCRC}	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

a See *Query_BAT* command definitions for more details when there is a SSD associatted with the response.

C.4.5 Command response: *NAK*

Table C.28 — *NAK*[b] command-response table[a]

Starting State	Condition	Response	Next State
battery ready	all	–	**battery ready**
arbitrate, reply, acknowledged, open, secured	all	–	**arbitrate**
killed	all	–	**killed**

[a] In Manchester with Long Activation and Interrogator locking in effect, the Tag checks first the Interrogator ID, if it matches, it proceeds with the command execution, otherwise, it ignores the command.
[b] Notice this command contains session as a parameter.

C.4.6 Command response: *Req_RN*

See C.2.7.

C.4.7 Command response: *Select*

Table C.29 — *Select* command-response table[a,b]

Starting State	Condition	Response	Next State
hibernate	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready, arbitrate, reply, acknowledged, open, secured	all	assert or de-assert **SL**, or set **inventoried** to *A* or *B*	**battery ready**
killed	all	–	**killed**

[a] If session locking is in effect, the Tag only accepts the command if the session matches the activating session.
[b] If Interrogator locking in effect, the Tag first checks the Interrogator ID, if it matches, it proceeds with the command execution; if Interrogator locking is in effect but it does not match, it ignores the command; if Short Activation, it proceeds with the command execution.

C.4.8 Command response: *Read*

See C.2.9.

C.4.9 Command response: *Write*

See C.2.10.

C.4.10 Command response: *Kill*

See C.2.11.

C.4.11 Command response: *Lock*

See C.2.12.

C.4.12 Command response: *Access*

See C.2.13.

C.4.13 Command response: *BlockWrite*

See C.2.14.

C.4.14 Command response: *BlockErase*

See C.2.15.

C.4.15 Command response: *BlockPermalock*

See C.2.16.

C.4.16 Command response: T_2 timeout

See C.2.17.

C.4.17 Command response: *Long Activation*

Table C.30 — *Long Activation* command-response table

Starting State	Condition	Response	Next State
hibernate, stateful hibernate	Valid activation preamble detected	–	**activation code check**
	Otherwise	–	**hibernate, stateful hibernate**[a]
activation code check	Wildcard or matching Activation Mask detected	If Session Locking On, program specified session timeout timer. If Session Locking Off, clear all **inventoried** flag timeout timers	**battery ready**
	Otherwise	–	**hibernate, stateful hibernate**[a]
battery ready, arbitrate, reply, acknowledged, open, secured	Commanded to receive data rate mode of 16 Kbps or higher	Ignore command	**battery ready, arbitrate, reply, acknowledged, open, secured**
	Commanded to receive data rate mode of 8 Kbps	Manufacturer option to ignore or “re-activate”. If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
killed	Valid activation preamble detected	–	**killed**

[a] Return to **stateful hibernate** if at least one **inventoried** flag timer is still running.

C.4.18 Command response: *Short Activation*

Table C.31 — *Short Activation* command-response table

Starting State	Condition	Response	Next State
hibernate, stateful hibernate	valid activation preamble detected	–	**activation code check**
	otherwise	–	**hibernate, stateful hibernate**[a]
activation code check	Wildcard or matching Activation Mask detected	Clear all timers, set **inventoried** flags to A, deassert **SL**	**battery ready**
	otherwise	–	**hibernate, stateful hibernate**[a]
battery ready, arbitrate, reply, acknowledged, open, secured	Commanded to receive data rate mode of 16 Kbps or higher	Ignore command	**battery ready, arbitrate, reply, acknowledged, open, secured**
	Commanded to receive data rate mode of 8 Kbps	Manufacturer option to ignore or "re-activate". If re-activation supported, then set **inventoried** flag to *A* for the active battery ready session, deassert **SL**, and go to **hibernate** mode/**activation code check** state	**activation code check**
killed	valid activation preamble detected	–	**killed**

[a] Return to **stateful hibernate** if at least one **inventoried** flag timer is still running.

C.4.19 Command response: *Query_BAT*

Table C.32 — *Query_BAT*[a,d] command-response table

Starting State	Condition	Response	Next State
hibernate	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready, arbitrate, reply[c,d]	Tag operating in battery assisted mode; slot=0; matching **Tag type** & **inventoried**[b] & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **Tag type & inventoried**[b] & **SL** flags	–	**arbitrate**
	otherwise	–	**battery ready**
acknowledged, open, secured[c,d]	Tag operating in battery assisted mode; slot=0; matching **Tag type & inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	Tag operating in battery assisted mode; slot<>0; matching **Tag type & inventoried**[b] & **SL** flags	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	Tag operating in battery assisted mode; all other conditions	transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**battery ready**
killed	all	–	**killed**

[a] *Query_BAT* (in any state other than killed) starts a new round and may change the session (if session locking is off); *Query_BAT* also instructs a Tag to load a new random value into its slot counter.

[b] As described in 6.4.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

[c] If session locking is in effect, the Tag only accepts the command if the session matches the activating session.

[d] If Interrogator locking in effect, the Tag first checks the Interrogator ID, if it matches, it proceeds with the command execution; if Interrogator locking is in effect but it does not match, it ignores the command; if Short Activation, it proceeds with the command execution.

[e] Tag checks the Tag Type Select field, if it matches the conditions in that field, Tag executes this command, otherwise, it ignores it.

C.4.20 Command response: *Next*

Table C.33 —*Next* command-response table

Starting State	Condition	Response	Next State
hibernate,	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready	all	–	**battery ready**
arbitrate, reply	all	-	**arbitrate, reply**
acknowledged, open, secured	valid handle, session **hibernate** timer running	backscatter RN16, handle or RN16, set sensitivity as indicated if that sensitivity is supported	**stateful hibernate**
	valid handle, no session **hibernate** timer	backscatter RN16, handle or RN16, set sensitivity as indicated if that sensitivity is supported	**hibernate**
	otherwise	–	**acknowledged, open, secured**
killed	all	–	**killed**

[a] Tags may switch to passive mode in the course of an inventory round if battery is drained below critical threshold; In passive mode *Next* is no longer supported.

C.4.21 Command response: *Deactivate_BAT*

Table C.34 — *Deactivate_BAT*[a] command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, reply, acknowledged, open, secured	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** & **SL** flags, Override=0	–	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), Override=1	Set **inventoried** flags to A, deassert **SL**, and clear timers	**stateful hibernate**
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), matching **inventoried** but mismatching **SL**, Override=0	–	**battery ready**

Starting State	Condition	Response	Next State
	Matching Interrogator ID (if Interrogator locking is in effect), matching activating session (if session locking is in effect), mismatching **inventoried**, **SL** does not matter, Override=0	Set **inventoried** flag to A	**battery ready**
	Mismatching Interrogator ID (if Interrogator locking is in effect) or mismatching activating session (if session locking is in effect), Override=0	–	**battery ready, arbitrate, reply, acknowledged, open, secured**
killed	all	–	**killed**
[a] Don't care' in **inventoried** flag use field is interpreted as any flag state matches.			

C.4.22 Command response: *Broadcast ID*

Table C.35 — *Broadcast ID* command-response table

Starting State	Condition	Response	Next State
hibernate,	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready, arbitrate, reply, acknowledged, open, secured	all	–	**battery ready, arbitrate, reply, acknowledged, open, secured**
killed	all	–	**killed**

C.4.23 Command response: *Multirate_Reset*

Table C.36 — *Multirate_Reset* command-response table

Starting State	Condition	Response	Next State
hibernate,	All	–	**hibernate**
stateful hibernate	All	–	**stateful hibernate**
battery ready ,arbitrate, reply, acknowledged, open, secured	All	Clear all timers, set **inventoried** flags to A, deassert **SL**. This applies to all Normal Mode (not Hibernation, see Figure 80) states, including intermediate steps within the *Kill* or *Access* sequences prior to the Tag completing the sequence (receiving both halves of the appropriate password).	**hibernate**
killed	All	–	**killed**

C.4.24 Command response: *HandleSensor*

Table C.37 — *HandleSensor* command-response table

Starting State	Condition	Response	Next State
hibernate, stateful hibernate	All	–	**hibernate, stateful hibernate**[b]
activation code check	All	–	**activation code check**
battery ready	All	–	**battery ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open, secured	valid handle & valid command payload	backscatter header = 0, response code[a] and handle when internal processing done	**open, secured**
	valid handle & invalid command payload	backscatter header = 1, error code (see Annex I) and handle	**open, secured**
	invalid handle	–	**open, secured**
killed	all	–	**killed**

[a] Sensor specific output of HandleSensor command payload processing.

[b] Return to **stateful hibernate** if at least one **inventoried** flag timer is still running.

C.4.25 Command response: *BroadcastSync*

See C.3.5.

C.4.26 Command response: Session Flag timer timeout

Table C.38 — Session Flag timer timeout command-response table

Starting State	Condition	Response	Next State
stateful hibernate	all	Set that flag to A	**stateful hibernate**
activation code check	all	Set that flag to *A* as soon as current operations allow	**activation code check**
battery ready	all	Set that flag to *A* as soon as current operations allow	**battery ready**
arbitrate	all	Set that flag to *A* as soon as current operations allow	**arbitrate**
reply	all	Set that flag to *A* as soon as current operations allow	**reply**
acknowledged	all	Set that flag to *A* as soon as current operations allow	**acknowledged**
open	all	Set that flag to *A* as soon as current operations allow	**open**
secured	all	Set that flag to *A* as soon as current operations allow	**secured**
killed	all	–	**killed**

C.4.27 Command response: INACT_T or Selective Global Timeout

Table C.39 — INACT_T[a] or Selective Global Timeout command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, open, secured	all	Set active **inventoried** flag to *A* and clear active session flag timeout timer, **SL→~SL**	**stateful hibernate**
killed	all	–	**killed**

[a] See clause 7.3.2.2 for details on the refresh of the INACT_T timer

C.4.28 Command response: Global Timeout

Table C.40 —Global Timeout command-response table

Starting State	Condition	Response	Next State
battery ready, arbitrate, reply, acknowledged, open, secured	all	Set active **inventoried** flag to *A* and clear active session flag timeout timer, **SL→~SL**	**stateful hibernate**
killed	all	–	**killed**

C.4.29 Command response: T_A

Table C.41 — T_A command-response table

Starting State	Condition	Response	Next State
activation code check	all	–	**battery ready**
killed	all	–	**killed**

C.4.30 Command response: OpRegister Read/Write

Table C.42 — *OpRegister Read/Write* command-response table

Starting State	Condition	Response	Next State
hibernate	all	–	**hibernate**
stateful hibernate	all	–	**stateful hibernate**
battery ready	all	–	**battery ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	all	–	**open**
secured	valid handle, read, valid reg ID and wordcount	Successful response is to backscatter header (value = “0”) bit, data being read, RN handle, and CRC-16 as per Table 270.	**secured**
	valid handle, write, valid reg ID and wordcount	Successful response is to backscatter header (value = “0”) bit, RN handle, and CRC-16 as per Table 270.	**secured**
	valid handle, invalid reg ID, wordcount, or any other failure	Failure response is to backscatter header (value = “1”) bit, error code as per Annex K, RN handle, and CRC-16.	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.4.31 Command response: Invalid command

Table C.43 — Invalid command table

Starting State	Condition	Response	Next State
battery ready[a]	all	–	**battery ready**
arbitrate[b]	all	–	**arbitrate**
reply[b]	all	–	**reply**
acknowledged[b]	all	–	**acknowledged**
open[b]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**open**
	otherwise valid commands, except *Req_RN*, *Query_BAT* interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**
secured[b]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**secured**
	otherwise valid commands, except *Req_RN, Query*, *Query_BAT* interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 23 and Figure 25).	–	**arbitrate**
killed[a]	all	–	**killed**

[a] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

[b] "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query_BAT with session locking off*) with a session parameter not matching that of the inventory round currently in progress, an otherwise valid command interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively; or any other command either not recognized or not executable by the Tag.

Annex D
(informative)

Example slot-count *(Q)* selection algorithm

D.1 Example algorithm an Interrogator might use to choose *Q*

Figure D.1 shows an algorithm an Interrogator might use for setting the slot-count parameter *Q* in a *Query* command. Q_{fp} is a floating-point representation of *Q*; an Interrogator rounds Q_{fp} to an integer value and substitutes this integer value for *Q* in the *Query*. Typical values for *C* are $0.1 < C < 0.5$. An Interrogator typically uses small values of *C* when *Q* is large, and larger values of *C* when *Q* is small.

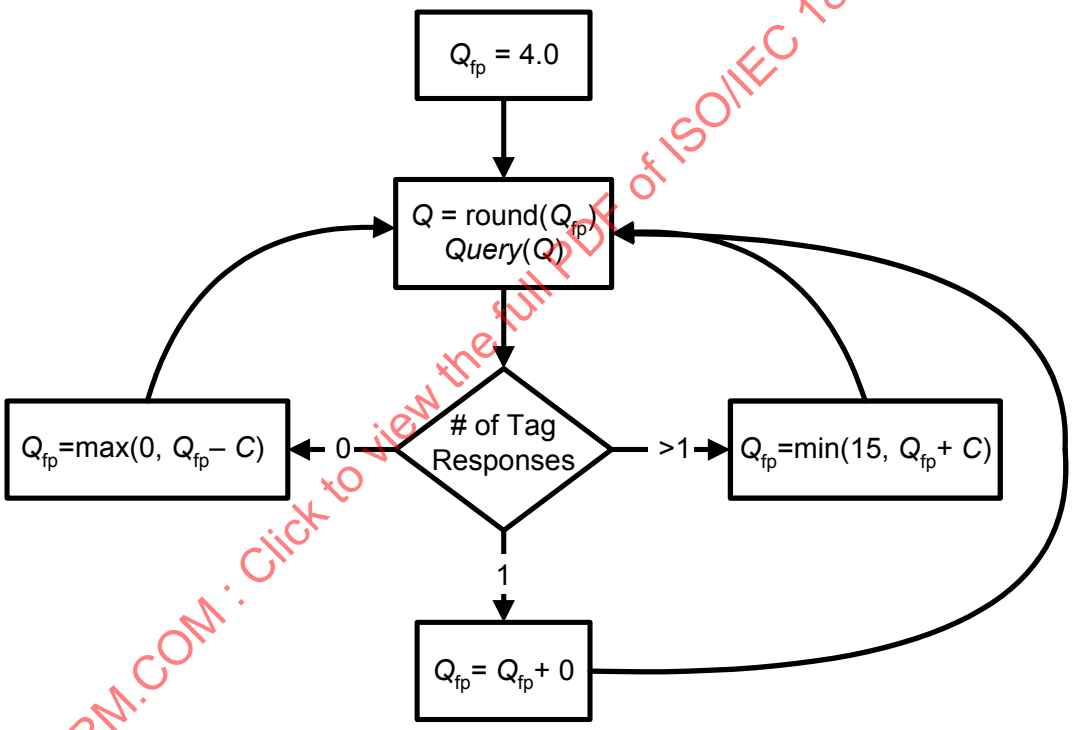


Figure D.1 — Example algorithm for choosing the slot-count parameter *Q*

Annex E
(informative)

Example of Tag inventory and access

E.1 Example inventory and access of a single Tag

Figure E.1 shows the steps by which an Interrogator inventories and accesses a single Tag that does not support extended protocol control.

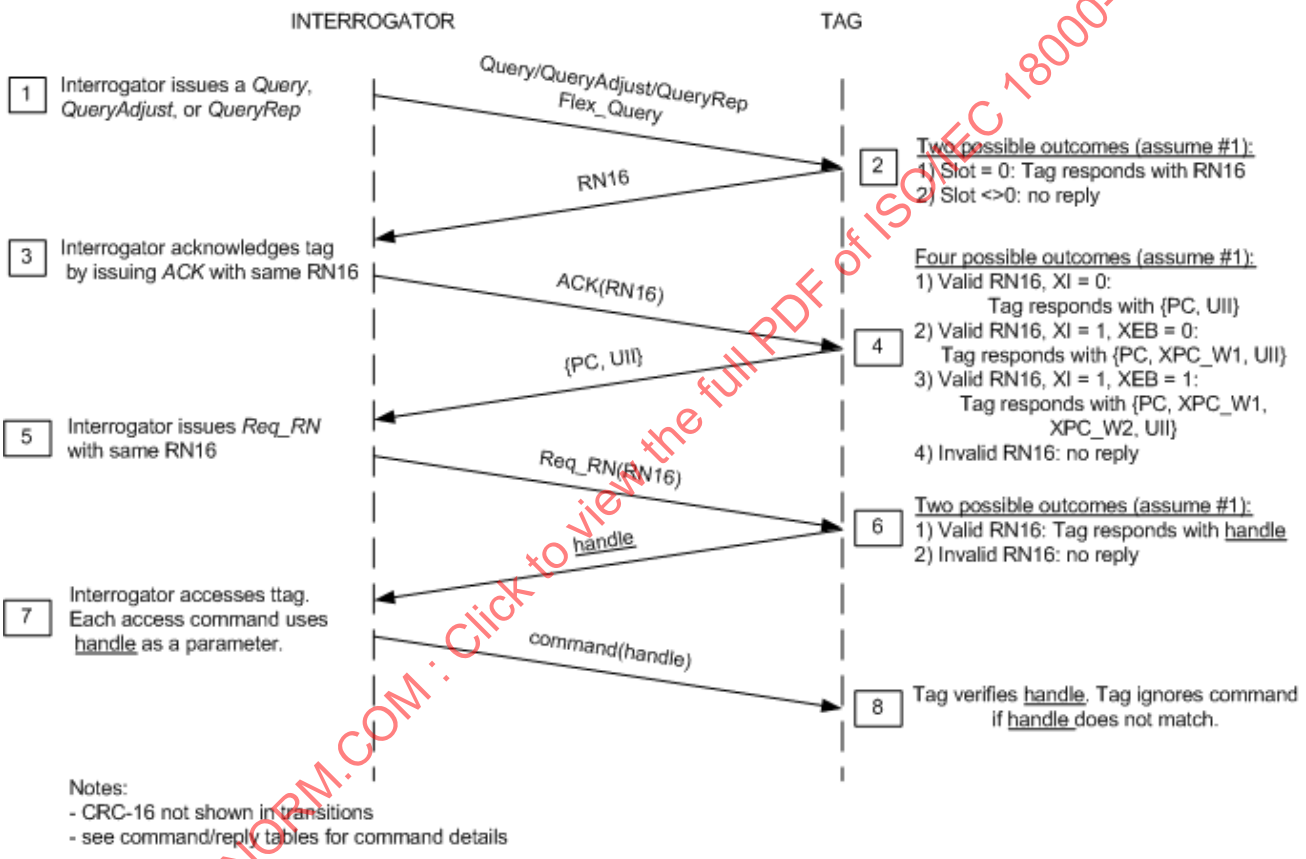


Figure E.1 — Example of Tag inventory and access if bit 16_h (XI) in the UII memory is not asserted

In contrast, Figure H.2 shows the steps by which an Interrogator inventories and accesses a single Tag that supports XPC_W1.

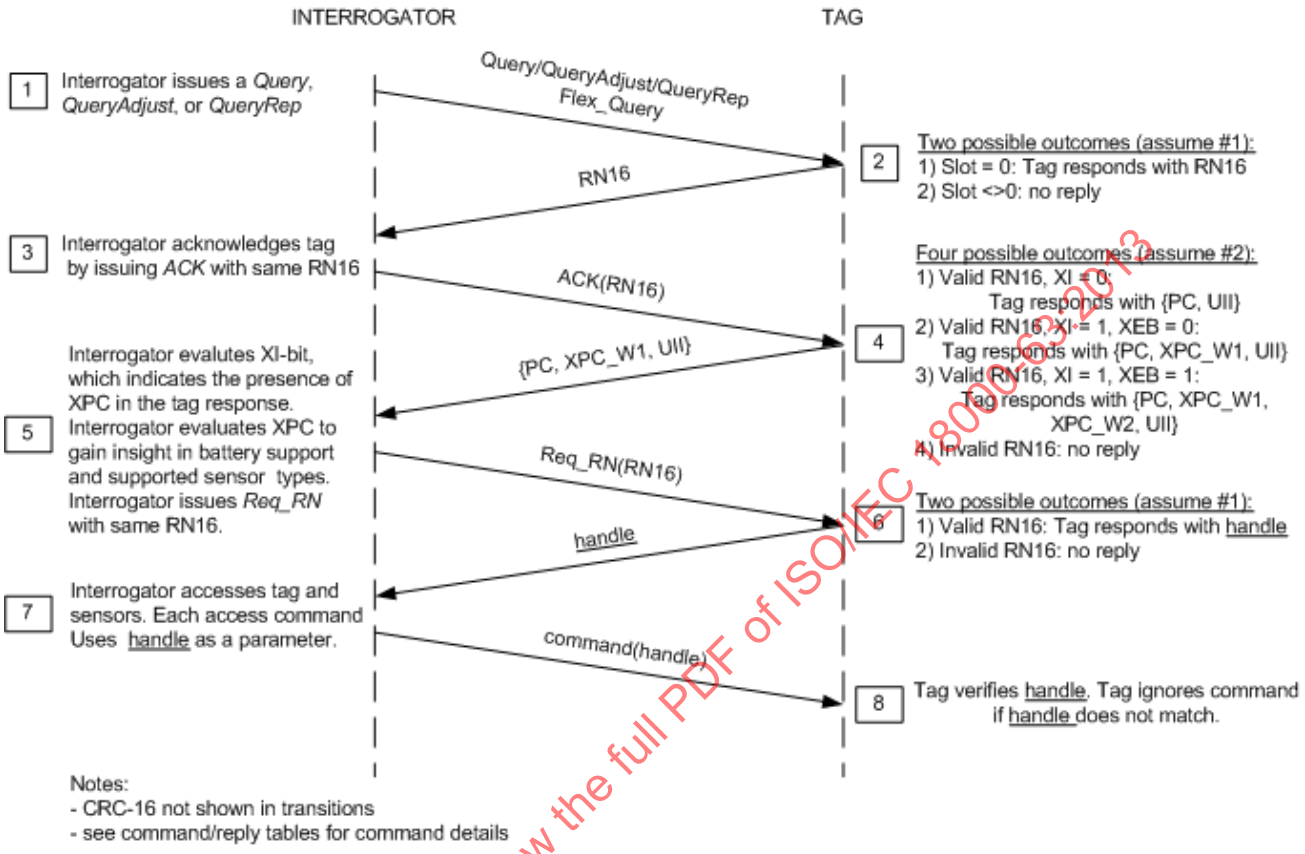


Figure E.2 — Example of Tag inventory and access if bit 16_h (XI) in the UII memory is asserted

Figure H.3 shows the same scenario as shown in Figure H.2 but additionally assumes that XPC_W2 is supported by the Tag.

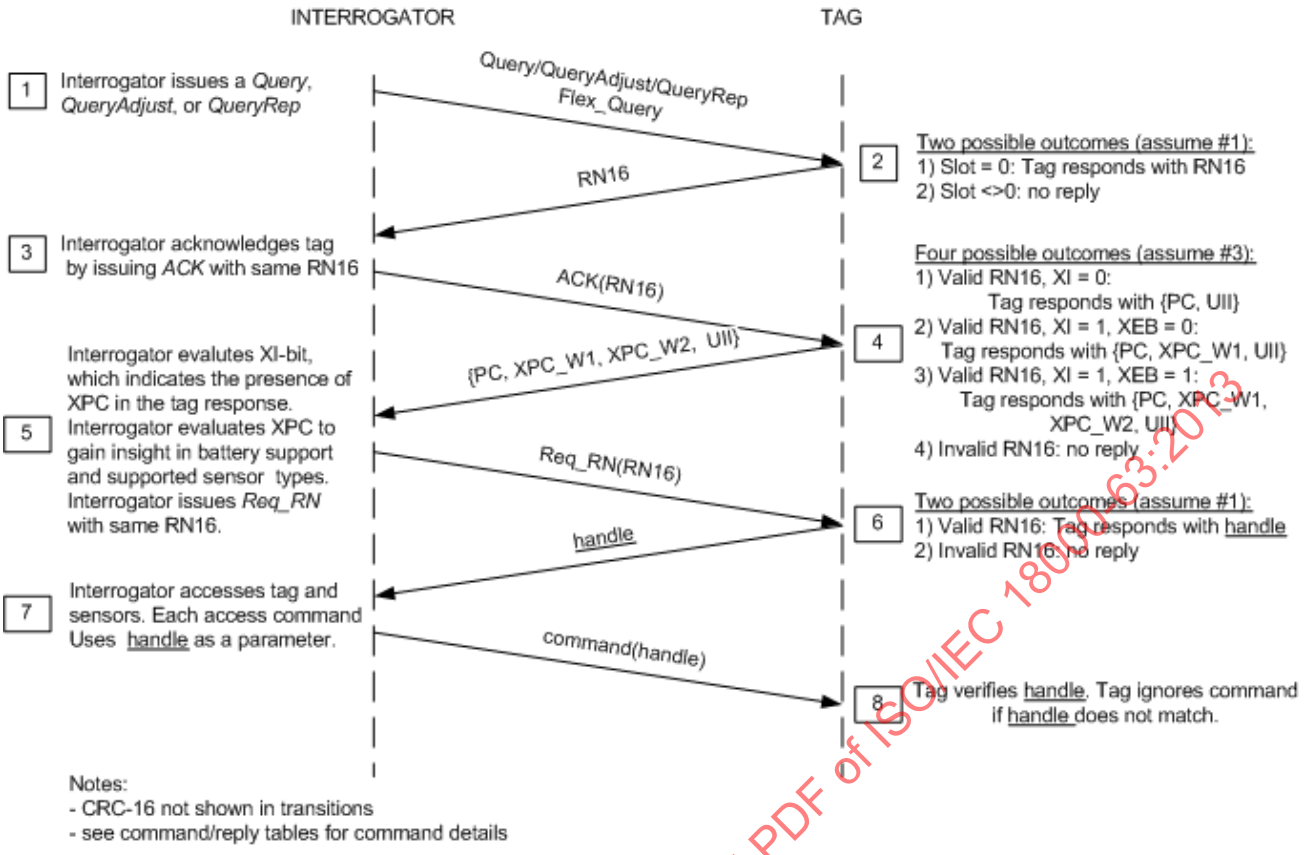


Figure E.3 — Example of Tag inventory and access if bit 16_h (XI) and bit 210_h (XEB) in the UII memory are asserted

If a Tag is equipped with a Simple Sensor, bit 215_h (SS) in the UII memory is asserted, see 7.6.2, and transmission of Simple Sensor Data subsequent to the UII in the response to an *ACK* command may be explicitly requested by the Interrogator by issuing an inventory command with SS Response flag asserted, see 8.3. Figure H.4 shows such a scenario.

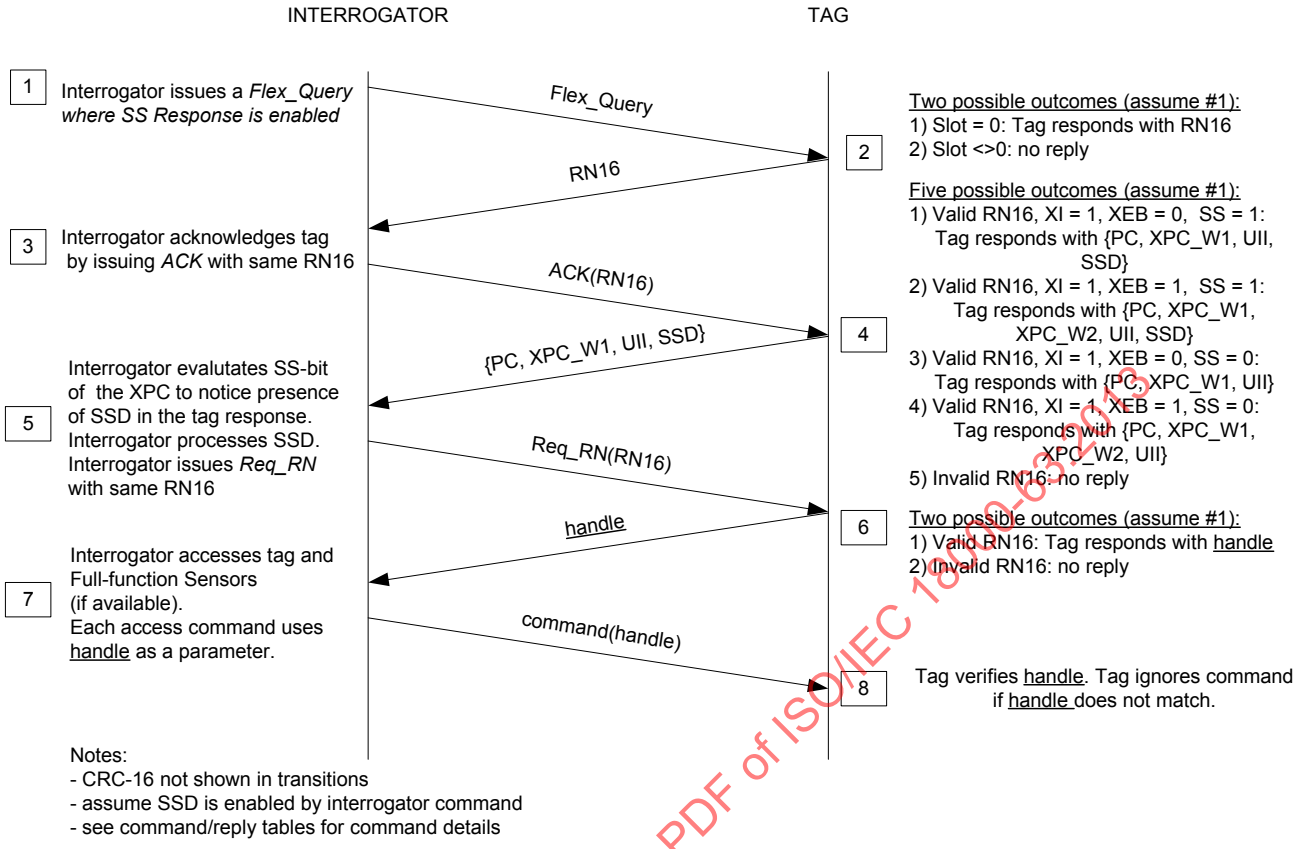


Figure E.4 — Example of Tag inventory and access if Simple Sensor Data Block is provided

Details on how to access Full Function Sensors as indicated in Step 7 can be found in 8.6.

Annex F
(informative)

Calculation of 5-bit and 16-bit cyclic redundancy checks

F.1 Example CRC-5 encoder/decoder

An exemplary schematic diagram for a CRC-5 encoder/decoder is shown in Figure F.1, using the polynomial and preset defined in Table 12.

To calculate a CRC-5, first preload the entire CRC register (i.e. Q[4:0], Q4 being the MSB and Q0 the LSB) with the value 01001_2 (see Table F.1), then clock the data bits to be encoded into the input labelled DATA, MSB first. After clocking in all the data bits, Q[4:0] holds the CRC-5 value.

To check a CRC-5, first preload the entire CRC register (C[4:0]) with the value 01001_2, then clock the received data and CRC-5 {data, CRC-5} bits into the input labelled DATA, MSB first. The CRC-5 check passes if the value in Q[4:0] =00000_2.

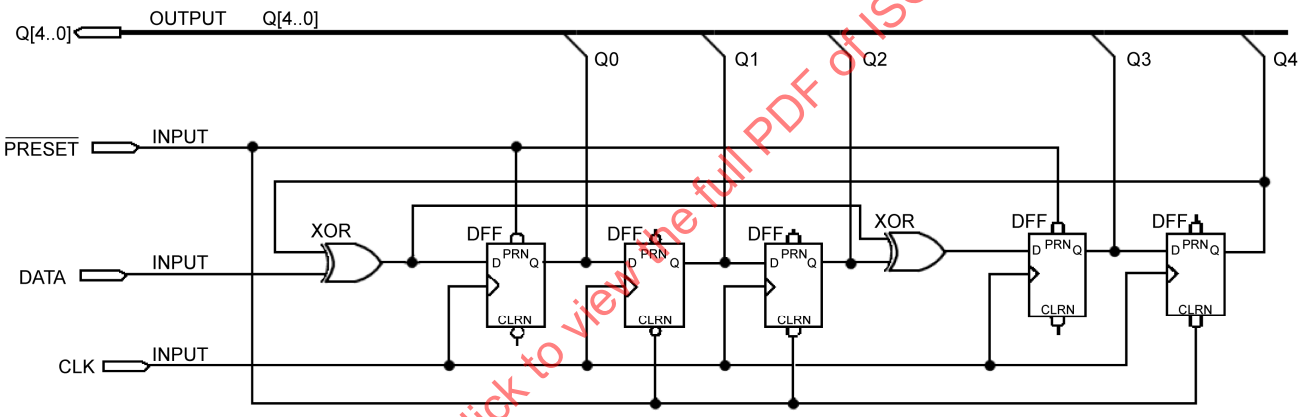


Figure F.1 — Example CRC-5 circuit

Table F.1 — CRC-5 register preload values

Register	Preload value
Q0	1
Q1	0
Q2	0
Q3	1
Q4	0

F.2 Example CRC-16 encoder/decoder

An exemplary schematic diagram for a CRC-16 encoder/decoder is shown in Figure F.2, using the polynomial and preset defined in Table 11 (the polynomial used to calculate the CRC-16, $x^{16} + x^{12} + x^5 + 1$, is the CRC-CCITT International Standard, ITU Recommendation X.25).