ISO/IEC JTC 1/SC 31 N3191

Date: 2010-05-17

ISO/IEC FDIS 18000-3:2010(E)

ISO/IEC JTC 1/SC 31/WG 4

Secretariat: ANSI

Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz

Technologies de l'information — Identification par radiofréquence (RFID) pour la gestion d'objets — Partie 3: Paramètres de communications d'une interface d'air à 13,56 MHz

Document type: International Standard
Document subtype:
Document stage: (50) Approval
Document language: E

D:\ISO\isomacroserver-prod\temp\DOC2PS\DOC2PS.appladmin@srvappl02_22\935008048506.doc STD Version 2.1c2

Contents

Page

Foreword

This second edition cancels and replaces the first edition (ISO/IEC 18000-3:2004), which has been technically revised.

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.

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

ISO/IEC 18000-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This third edition cancels and replaces the second edition (ISO/IEC FCD 18000-3.2), which has been technically revised.

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*

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

Introduction

ISO/IEC 18000 has been developed by ISO/IEC JTC 1, SC 31, WG 4, *Radio frequency identification for item management*, in order to provide a framework to define common communications protocols for Internationally useable frequencies for Radio Frequency Identification (RFID), and, where possible, to determine the use of the same protocols for ALL frequencies such that the problems of migrating from one to another are diminished; to minimise software and implementation costs; and to enable system management and control and information exchange to be common as far as is possible.

This part of ISO/IEC 18000 has been prepared in accordance with the requirements determined in ISO/IEC 18000-1.

ISO/IEC 18000-1 provides explanation of the concepts behind this part of ISO/IEC 18000.

This part of ISO/IEC 18000 has 3 MODES of operation, intended to address different applications. Clause 8 of this part of ISO/IEC 18000 summarises the differences between MODE characteristics. The detailed technical differences between the modes are shown in the parameter tables.

This part of ISO/IEC 18000 relates solely to systems operating at 13,56 MHz.

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.

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

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licenses 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 the ISO and IEC. Information may be obtained from the following companies.

Contact details	**Patent number**
EM Microelectronic SA Mr Marc Degrauwe, IP manager Rue des Sors 3 CH-2074 Marin, Switzerland (T) +41 32 755 51 11 (F) +41 32 755 54 03 info@ emmicroelectronic.com www.emmicroelectronic.com	EP 97 115772.2 EP 0 902 546 US 151803

Contact details	Patent number
Impinj, Inc. Chris Diorio, CTO 701 N 34th Street, suite 300 Seattle, WA 98103, USA (T) +1 206 834 1115 (F) +1 206 517 5262 diorio@ impinj.com www. impinj.com	
Intermec IP Corporation Phyllis T. Turner-Brim, Esq. Legal Department 6001 – 36th Avenue West Everett, WA 98203, USA (T) +1 425-265-2480 (F) +1 425-501-6587 phyllis.turnerbrim@intermec.com	US 5673037 DE 69530547.6 EP 0702323 FR EP0702323 GB EP0702323 KR 204748 TW 318306 US 6172596 US 6400274 (claims 1-10 only) US 6404325 US 5550547 TW 307079 KR 210830 US 5521601 US 5777561 US 5828318 EP 1020044 US 5912632 US 5942987 TW 352492 KR 244844 US 5995019
Intermec IP Corporation (continued)	US 6400274 (claims 11 et. Seq.) US 6288629 US 6812841 US 6812852 US 7427912

Contact details	Patent number
Magellan Technology Pty. Limited IP Manager 65 Johnston St Annandale, NSW 2038, Australia (T) +61 2 9562 9800 (F) +61 2 9518 7620 info@magellan-technology.com	US5302954 SG37971 DE3854478D EP0390822 US5485154 US10/927,957 US6967573 JP2002500465T JP2006-180816 DE69835452 EP1048126 AU2006202886 AU785098 US7248145 US7259654 US11/538,271 US11/538/242 JP2003 526148 JP2006 344227 DE60119910 EP1266458 EP07013773 EP1544782 EP1544788 EP1679635
NXP B.V. Marc Schouten Intellectual Property & Licensing High Tech Campus 32 5656 AE Eindhoven The Netherlands (T) +31 40 27 26951 (F) +31 40 27 42640 marc.schouten@nxp.com	

Contact details	Patent number
TAGSYS, SA Alastair McArthur, CTO 180. Chemin de Saint Lambert 13821 La Penne sur Huveaune FRANCE +33 491 27 57 00 +33 49t 27 57 01 alastair.mcarthur@ tagsysrfrd.com.au www.tagsysrfid.com	US 6,641,036 EP 1232471 US 6992567 EP 1256083 US 6946951 EP 1358644 US 6538564 EP 953181
Texas Instruments Inc. Robby Holland, Licensing Manager P.O. Box 655464, MS 3999 Dallas TX 75256 (T) +1 972 917 4367 (F) +1 972 917 4418 r-holland3@ti.com	EP1 038257 US 09/315708 JP 00-560700 EP 1 034644 US 6442215 CN 1273730A WO00/04686 EP 0669591B AT-PS 401127
Zebra Technologies Corporation Eric McAlpine, IP Counsel Legal Department 333 Corporate Woods Parkway Vernon Hills, IL 60061-3109 (T) +1 847 793 5640 (F) +1 847 955 4514 emcalpine@zebra.com	US 6784787 EP 1031046 EP 1291671 EP 05017862.3 US 5680459 US 5557280 US 5699066 EP 0585132 US 6198381 JP 10-272945 US 5537105 US 5966083 US 5995017

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 IP submissions to ISO can be found at:

http://www.iso.org/patents

Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz

1 Scope

This part of ISO/IEC 18000 provides physical layer, collision management system and protocol values for RFID systems for Item Identification operating at 13,56 MHz in accordance with the requirements of ISO/IEC 18000-1.

This part of ISO/IEC 18000 provides definitions for systems for each MODE determined in Clause 6 below.

This part of ISO/IEC 18000 defines three non-interfering MODES.

— The MODES are NOT interoperable.

— The MODES, whilst not interoperable, are non-interfering.

2 Conformance

2.1 Claiming conformance

In order to claim conformance with this part of ISO/IEC 18000, it is necessary to comply with all of the relevant clauses of this part of ISO/IEC 18000 except those marked ‘optional’. It is also necessary to operate within the local national radio regulations (which may require further restrictions).

Relevant conformance test methods are defined in ISO/IEC TR 18047-3.

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.

ARIB STD-T82, *Contactless IC Card System,* http://www.arib.or.jp/english/html/overview/st_ej.html

ISO/IEC 13239, Information technology – *Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*

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

ISO/IEC 15693 (all parts), *Identification cards — Contactless integrated circuit cards — Vicinity cards*

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 18000-1, *Information technology — Radio frequency identification for item management — Reference architecture and definition of parameters to be standardized*

ISO/IEC TR 18046*, Information technology — Automatic identification and data capture techniques — Radio frequency identification device performance test methods*

ISO/IEC TR 18047-3, *Information technology — Radio frequency identification device conformance test methods — Part 3: Test methods for air interface communications at 13,56 MHz*

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

U.S. CFR, Title 47, Chapter I, Part 15, *Radio-frequency devices, U.S. Federal Communications Commission,* http://www.access.gpo.gov/nara/cfr/waisidx_06/47cfr15_06.html

EPCglobal™ *Tag Data Standards (Version 1.3 and above)*

4 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
cover-coded text
information that is cover-coded.

4.2
cover-coding
method by which an interrogator obscures information that it is transmitting to a tag.

4.3
full-duplex communications
data is communicated while the transceiver transmits the activation field.

4.4
half-duplex communications
data transmission in either direction, one direction at a time.

4.5
handle
RN16; the 16 bit random number that is used to authenticate tags in the open or secured state.

4.6
PacketCRC
16-bit cyclic-redundancy check (CRC) code that a tag with nonzero-valued XI dynamically calculates over its PC, XPC, and UII and provides by loadmodulatation during inventory (see StoredCRC).

4.7
PacketPC
Protocol-control information that a tag with nonzero-valued XI dynamically calculates and provides by loadmodulation during inventory (see StoredPC).

4.8
Phase Jitter Modulation (PJM)
modulation technique that transmits data as very small phase changes in the powering field.

4.9
physical layer
data coding and modulation waveforms used in interrogator-to-tag and tag-to-interrogator communication.

4.10
pivot
average length of an R=>T data symbol: pivot = (0-length + 1-length) / 2. See also "R=>T" in section 5.1.

4.11
plaintext
information that is not cover-coded.

4.12
recommisioning
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.

4.13
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 may backscatter during inventory (see PacketCRC).

4.14
StoredPC
Protocol-control information stored in UII memory that a tag with zero-valued XI provides by loadmodulation during inventory (see PacketPC).

4.15
Tari
reference time interval for a data-0 in interrogator-to-tag communication.

5 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms given in ISO/IEC 18000-1, ISO/IEC 19762 and the following apply.

5.1 Symbols

DR	ASK Method: divide ratio PJM Method: Bit 0 of the reply channel selection
F_c	Carrier frequency
M(ASK)	Tag reply modulation type
M_h	RF signal envelope ripple (overshoot)
M_l	RF signal envelope ripple (undershoot)
M(PJM)	Bit 1 and bit 2 of the reply channel selection
M_s	RF signal level when OFF
Q	Slot-count parameter (parameter that an interrogator uses to regulate the probability of tag response)
R	Interrogator (also sometimes called Reader)

R=>T	Interrogator-to-tag
RTcal	Interrogator-to-tag calibration symbol
T	Tag
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_{pri}	Link pulse-repetition interval (T_{pri} = 1/LF)
T_r or $T_{r,10\text{-}90\%}$	RF signal envelope rise time
TRext	ASK Method: chooses whether the T=R preamble is prefixed with a pilot tone PJM Method: Bit 3 of the reply channel selection
T_s	RF signal settling time
T=>R	Tag-to-interrogator
TRcal	Tag-to-interrogator calibration symbol
X_{fp}	Floating-point value
$xxxx_2$	Binary notation
$xxxx_h$	Hexadecimal notation
≈	MODE 1 - the value is a rounded value (e.g. ≈ 75,52 µs)

5.2 Abbreviated terms

ARIB	Association of Radio Industries and Businesses
AFI	Application family identifier
AM	Amplitude modulation
ASK	Amplitude shift keying
BPSK	Binary Phase Shift Keying
CEPT	Conference of European Posts and Telecommunications
CFR	Code of Federal Regulations
CRC	Cyclic Redundancy Check
NOTE:	This specification uses two CRC algorithms: CRC-5 (5-bit CRC) and CRC-16 (16-bit CRC) and three different logical CRC-16s: StoredCRC, PacketCRC and CRC-16c. For the UII bank word 0 or ACK the following two logical CRC-16s are used: - StoredCRC = CRC-16 calculated at startup and mapped to UII word 0 - PacketCRC = CRC-16 calculated over the response data of the tag in case of the ACK command For all other cases and commands the following logical CRC-16 is used: - CRC-16c = CRC-16 calculated over the response data of the tag
CW	Continuous wave
dBch	Decibels referenced to the integrated power in the reference channel
DSB	Double sideband
DSB-ASK	Double-SideBand Amplitude-Shift Keying
DR	Divide ratio
ERC	European Radiocommunications Committee
ERM	Electromagnetic compatibility and Radio spectrum Matters

ETSI	European Telecommunications Specifications Institute
FCC	Federal Communications Commission
FT	Frequency Tolerance
ITF	Interrogator talks first (reader talks first)
LF	Link Frequency (LF = 1/Tpri)
MFM	Modified Frequency Modulation
N/A	Not Applicable
NSI	Numbering system identifier
PIE	Pulse-Interval Encoding
PJM	Phase Jitter Modulation
ppm	Parts-per-million
PC	Protocol control
RF	Radio frequency
RFU	Reserved for future use
RN16	16-bit random or pseudo-random number
RNG	Random or pseudo-random number generator
SRD	Short Range Devices
TDM	Time-division multiplexing or time-division multiplexed (as appropriate)
TID	Tag-identification or tag identifier, depending on context
UII	Unique item identifier
UMI	User-memory indicator
XI	XPC indicator
XPC	Extended protocol control
XPC_W1	XPC word 1
XPC_W2	XPC word 2
XTID	Extended TID indicator (see version 1.3 and above of the EPCglobal^{TM} Tag Data Standards)

5.3 Notation

Mode 3 of this specification uses the following notational conventions:

- 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 specification shall make an explicit statement. Example: *BeginRound*.
- Procedures are shown as ***italics underline***
- 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 communications signal air interface from an interrogator to a tag (reader-to-tag).
- The symbol, T=>R, refers to commands or communications signal air interface from a tag to an interrogator (tag-to-reader).

6 Requirements: Physical layer, collision management system and protocol values for 13,56 MHz systems

6.0 General and applicable to All Modes

6.0.1 Presentation as determined in ISO/IEC 18000-1

The context, form and presentation of this part, which provides physical layer, collision management system and protocol value definitions for RFID systems for item identification operating at 13,56 MHz are in accordance with the requirements of ISO/IEC 18000-1.

6.0.2 ISO/IEC 18000-3 Interoperability

This part of ISO/IEC 18000 specifies three MODES of operation at 13,56 MHz
These MODES are not interoperable, but they are expected to operate without causing any significant interference with each other. Any known causes of interference are listed in Annex O.

NOTE It is recommended that users select one MODE for any specific application.

NOTE Local national regulations may further limit either power, frequency or bandwidth allocations and such limitations may reduce the capability of a system within that country. Users shall have the responsibility to ensure that they use only systems that comply with these regulations. This implies a user responsibility to obtain proofs from manufacturers, and where appropriate have adequate tests carried out to assure that systems are in compliance.

NOTE At the time of preparation of this part of ISO/IEC 18000, the interrogator to tag link and tag to interrogator link physical layer emissions may be subject to type approval or certification. It is therefore necessary to make reference to local or regional radio regulations and radio standards in addition to this part of ISO/IEC 18000. All systems are required to comply with local radio regulations, which may affect performance.

6.0.3 ISO/IEC 18000-3 interrogator conformance/compliance

To claim compliance with this part of ISO/IEC 18000, an interrogator/ reader shall support either MODE 1, MODE 2, or MODE 3. The interrogator may support any or all modes as an option (the modes are not interoperable).

6.0.4 ISO/IEC 18000-3 tag compliance.

To claim compliance with this part of ISO/IEC 18000, a tag shall support either MODE 1, MODE 2, or MODE 3. The tag may support any or all modes as an option (the modes are not interoperable).

6.0.5 Command structure and extensibility

Clauses 6.1, 6.2, and 6.3, include definition of the structure of command codes between an interrogator and a tag and indicate how many positions are available for future extensions. Command specification clauses provide a full definition of the command and its presentation. Each command is labelled as being 'mandatory' or 'optional'. In accordance with ISO/IEC 18000-1, the clauses of this part of ISO/IEC 18000 make provision for 'custom' and 'proprietary' commands.

The types of permitted command options are defined in subclauses 6.0.6 to 6.0.9.

6.0.6 Mandatory commands

A mandatory command shall be supported by all tags that claim to be compliant. Interrogators which claim compliance shall support all mandatory commands.

6.0.7 Optional commands

Optional commands are commands that are specified within the International Standard. Interrogators shall be technically capable of performing all optional commands that are specified in the International Standard (although need not be set up to do so). Tags may or may not support optional commands.

If an interrogator or a tag implements an optional command, it shall implement it in the manner specified in this standard.

6.0.8 Custom commands

Custom commands may be enabled by an International Standard, but they shall not be specified in that International Standard.

A custom command shall not solely duplicate the functionality of any mandatory or optional command defined in the International Standard by a different method. An interrogator shall use a custom command only in accordance with the specifications of the tag manufacturer.

6.0.9 Proprietary commands

Proprietary commands may be enabled by an International Standard, but they shall not be specified in that International Standard.

A proprietary command shall not solely duplicate the functionality of any mandatory or optional command defined in the International Standard by a different method. Vendors shall not provide proprietary means to circumvent the protocol. Proprietary commands are intended for manufacturing purposes and shall not be used in field-deployed RFID systems.

6.1 Physical layer, collision management system and protocols for MODE 1

MODE 1 is not interoperable with any other MODES defined within this International Standard.

6.1.1 Read/Write system

MODE 1 describes a read/write system using a "interrogator talks first" technique.

6.1.2 Normative Aspects

The physical, collision management and transmission protocols determined in this MODE are defined in ISO/IEC 15693. Clauses 6.1.3 – 6.1.8 provide normative parts of MODE 1 by reference.

6.1.3 Conformance and performance measurement aspects

The performance and conformance measurement aspects for MODE 1 are given in the relevant clauses of Technical Reports (ISO/IEC TR 18046 and ISO/IEC TR 18047-3, respectively).

6.1.4 Physical Layer

The Physical layer for the MODE 1 air interface at 13,56 MHz shall be compliant with ISO/IEC 15693-2.

6.1.5 Protocol and collision management operating method

The collision management operating method for the MODE 1 air interface at 13,56 MHz shall be compliant with ISO/IEC 15693-3.

6.1.6 Commands

The commands for the MODE 1 air interface at 13,56 MHz shall be compliant with ISO/IEC 15693-3. Annex A shows the commands that are necessary to support encoding to ISO/IEC 15962, which is required for RFID for item management.

6.1.7 Parameter tables for interrogator to tag link

The parameter tables for interrogator to tag link for the MODE 1 air interface at 13,56 MHz shall be compliant with ISO/IEC 15693-2. See Table 1 — Parameter Tables for interrogator to tag link for details.

Table 1 — Parameter Tables for interrogator to tag link

Ref.	Parameter	Description/limits	Options/Comments
M1-Int: 1	Operating frequency range	1 interrogator to tag link channel at (centre frequency) 13,56MHz ± 7 kHz	
M1-Int: 1a	Default operating frequency	13,56 MHz	
M1-Int: 1b	Operating channels (for spread spectrum systems)	N/A	
M1-Int: 1c	Operating frequency accuracy	+/- 100 parts per million +/- 50 parts per million in Japan	
M1-Int: 1d	Frequency hop rate (for Frequency hopping [FHSS] systems)	N/A	
M1-Int: 1e	Frequency hop sequence (for Frequency hopping [FHSS] systems)	N/A	
M1-Int: 2	Occupied channel bandwidth	13,56 MHz ± 7 kHz with modulation as defined in 6.1.4	
M1-Int: 2a	Minimum receiver bandwidth	13,56 MHz ± (423,75± 40 kHz) 13,56 MHz ± (484,28 ± 40 kHz)	Centred at the sub carrier frequency.
M1-Int: 3	Interrogator transmit maximum Magnetic Field Strength Magnetic Field Strength limits within communication zone	The interrogator shall not generate a field higher than 12 A/m in any part of the volume where a tuned ISO card sized tag may be present Maximum operating field strength: 5 A/m for ISO card sized tags, as defined in ISO/IEC 7810. For other label form factors, the label manufacturer shall specify the maximum operating field strength.	(Inductive coupling) Where Local regulations restrict the Magnetic Field Strength Limits below those determined in this Clause, a degradation of local performance may be expected. Test methods for determining the interrogator operating field are defined in relative clauses of the Technical Report ISO/IEC TR 18047-3.
M1-Int: 3a	Minimum operating field strength	Minimum Operating Field Strength: 150 mA/m for tuned ISO card sized tags. Application requirements may result in a different minimum operating field strength. In that case the minimum operating field strength shall be declared by the tag manufacturer.	
M1-Int: 4	Interrogator transmit spurious emissions	compliant with regulations	

Ref.	Parameter	Description/limits	Options/Comments
M1-Int: 4a	Interrogator transmit spurious emissions, in-band (for Spread spectrum systems)	N/A	
M1-Int: 4b	Interrogator transmit spurious emissions, out-of-band	N/A	
M1-Int: 5	Interrogator transmitter spectrum mask	The modulation technique and bit coding enable maximum tag powering within the following regulations: US Jurisdictions : FCC 47 Part 15 EU : EN 300 – 330 Japan : ARIB STD – T82	
M1-Int:6	Timing	See below.	
M1-Int: 6a	Transmit to receive turn around time	<= 4320/fc	
M1-Int: 6b	Receive to transmit Turn around time	>= 4192/fc	
M1-Int: 6c	Dwell time or Interrogator transmit power on ramp	See 6.1.4	
M1-Int: 6d	Decay time or Interrogator transmit power down ramp	See 6.1.4	
M1-Int: 7	Modulation	Carrier amplitude modulation (ASK 100%, ASK 10%….)	
M1-Int: 7a	Spreading sequence (for Frequency hopping [FHSS] systems)	N/A	
M1-Int: 7b	Chip rate (for Spread Spectrum systems)	N/A	
M1-Int: 7c	Chip Rate Accuracy (for Spread spectrum systems)	N/A	
M1-Int: 7d	Modulation index	Two level amplitude modulation: 100% and 10% Modulation Index: **(a – b) / (a + b)** (NOTE Modulation Depth **(a – b)/ a**)	Either option at determination of the interrogator (See 6.1.4). Two modulation indexes are permitted and are determined by the interrogator. The RF tag shall decode both. Modulation shall use the principle of ASK. Depending on the choice made by the interrogator.
M1-Int: 7e	Duty cycle	See 6.1.4	
M1-Int: 7f	FM deviation	N/A	

Ref.	Parameter	Description/limits	Options/Comments
M1-Int: 8	Data coding	Data coding shall be implemented using pulse position modulation. Two data coding modes shall be supported by the RF tag. The selection shall be made by the interrogator and indicated to the RF tag within the Start of frame (SOF).	**Data coding mode:** **1 out of 256** The value of one single byte shall be represented by the position of one pulse. The position of the pulse on one of 256 successive time periods of 18,88 µs (256/fc), determines the value of the byte. **Data coding mode:** **1 out of 4** Pulse position modulation shall be used where the position determines two bits at a time. Four successive pairs of bits form a byte. The resulting data rate is 26,48 kbit/s (fc/512).
M1-Int: 9	Bit rate	1 out of 256 Data coding mode: 1,65 kbit/s (fc/8192) 1 out of 4 Data coding mode: 26,48 kbit/s (fc/512)	Either mode at the determination of the interrogator
M1-Int: 9a	Bit rate accuracy	Synchronous to the carrier frequency	
M1-Int: 10	Interrogator transmit modulation accuracy	N/A	

6.1.8 Parameter tables for tag to interrogator link

The parameter tables for tag to interrogator link for the MODE 1 air interface at 13,56 MHz shall be compliant with ISO/IEC 15693-2. See Table 2 — Parameter Tables for tag to interrogator link for details.

Table 2 — Parameter Tables for tag to interrogator link

Ref.	Parameter Name	Description/limits	Options/Comments
M1-Tag: 1	Operating frequency range Subcarrier frequencies	One or two subcarriers may be used. The selection between either shall be made by the interrogator using the first bit in the protocol header as defined in ISO/IEC 15693-3. The RF tag shall support both modes. When one subcarrier is used, the frequency f_{s1} of the subcarrier loadmodulation shall be f_c/32 (423,75 kHz). When two subcarriers are used, the frequency f_{s1} shall be f_c/32 (423,75 kHz), and the frequency f_{s2} shall be f_c/28 (484,28 kHz). When two subcarriers are used, continuous phase shall be used, as shown in supporting diagrams	
M1-Tag: 1a	Default operating frequency	13,56MHz ± 7 kHz	
M1-Tag: 1b	Operating channels (for Spread spectrum systems)	N/A	
M1-Tag: 1c	Operating frequency accuracy	13,56 MHz synchronous to the carrier	
M1-Tag: 1d	Frequency hop Rate (for Frequency hopping [FHSS] systems)	N/A	
M1-Tag: 1e	Frequency hop sequence (for Frequency hopping [FHSS] systems)	N/A	
M1-Tag: 2	Occupied channel bandwidth	13,56 MHz 4 subcarriers 1 sub carrier: 13,56 MHz ± (423,75 kHz ± 40 kHz) 2 sub carriers: 2 channel: 13,56 MHz ± (484,28 kHz ± 40 kHz)	
M1-Tag: 3	Transmit Maximum Magnetic Field Strength	In accordance with local regulations	
M1-Tag: 4	Transmit spurious emissions	N/A	
M1-Tag: 4a	Transmit spurious emissions, in-band (for Spread spectrum systems)	N/A	
M1-Tag: 4b	Transmit spurious emissions, Out-of-band	N/A	

Ref.	Parameter Name	Description/limits	Options/Comments
M1-Tag: 5	Transmit spectrum mask	In compliance with: US: FCC 47 Part 15 EU and other EN:300-330 (Transmit spectrum mask is not regulated in the Japan Radio Law)	
M1-Tag: 6	Timing	See below.	
M1-Tag: 6a	Transmit to receive turn around time	<= 4192/fc	
M1-Tag: 6b	Receive to transmit turn around time	Within 4320/fc – 4384/fc	
M1-Tag: 6c	Dwell time or Transmit power on ramp	N/A	
M1-Tag: 6d	Decay time or Transmit power down ramp	N/A	
M1-Tag: 7	Modulation (on the carrier)	The RF tag shall be capable of communication to the interrogator via an inductive coupling area whereby the carrier frequency is modulated to generate a subcarrier with frequency f_s. The subcarrier shall be generated by switching a load in the RF tag. Test methods for RF tag loadmodulation are defined in ISO/IEC TR 18047-3. The loadmodulation amplitude shall be at least 10 mV for ISO card sized tags when measured as described in the test methods. For other label form factors, the label manufacturer shall specify the loadmodulation.	
M1-Tag: 7a	Spreading sequence (for Frequency hopping [FHSS] systems)	N/A	
M1-Tag: 7b	Chip rate (for Spread spectrum systems)	N/A	
M1-Tag: 7c	Chip rate accuracy (for Spread spectrum systems)	N/A	
M1-Tag: 7d	On-off ratio	N/A	
M1-Tag: 7e	Subcarrier frequency	if one sub carrier is used : 423,75 kHz if two sub carriers are used : 423,75 kHz and 484,28 kHz	
M1-Tag: 7f	Subcarrier frequency accuracy Tolerance of direct generated tag to Interrogator link carrier	Derived from the carrier.	

Ref.	Parameter Name	Description/limits	Options/Comments
M1-Tag: 7g	Subcarrier modulation	One Subcarrier: A logic ***0*** starts with 8 pulses of 423,75 kHz (fc/32) followed by an unmodulated time of 18,88 µs. (256/fc). A logic ***1*** starts with an unmodulated time of 18,88 µs (256/fc) followed by 8 pulses of 423,75 kHz (fc/32). Two subcarriers: A logic ***0*** starts with 8 pulses of 423,75 kHz (fc/32) followed by 9 pulses of 484,28 kHz (fc/28). A logic ***1*** starts with 9 pulses of 484,28 kHz (fc/28) followed by 8 pulses of 423,75 kHz (fc/32)	Determined by the interrogator. RF tag shall support both.
M1-Tag: 7h	Duty cycle	N/A	
M1-Tag: 7 I	FM Deviation	N/A	
M1-Tag: 8	Data coding	Data shall be encoded using Manchester coding, of one or two subcarriers.	
M1-Tag: 9	Bit rate	Refer to 6.1.4	
M1-Tag: 9a	Bit rate accuracy	Derived from the carrier	
M1-Tag: 10	Tag transmit modulation accuracy (for Frequency hopping [FHSS] systems)	N/A	
M1-Tag: 11	Preamble	Framing has been chosen for ease of synchronization and independence of protocol. Frames are delimited by a Start of frame (SOF) and an End of frame (EOF) and are implemented using code violation.	
M1-Tag: 11a	Preamble length	See 6.1.4.	

Ref.	Parameter Name	Description/limits	Options/Comments
M1-Tag: 11b	Preamble waveform	**:SOF when using one subcarrier** The SOF comprises 3 parts: 1: an unmodulated time of 56,64 µs (768/fc), 2: 24 pulses of 423,75 kHz (fc/32), 3: followed by a logic ***1*** which starts with an unmodulated time of 18,88 µs. (256/fc) followed by 8 pulses of 423,75 kHz (fc/32). **:SOF when using two subcarriers** The SOF comprises 3 parts: 1: 27 pulses of 484,28 kHz (fc/28), 2: 24 pulses of 423,75 kHz (fc/32), 3: a logic ***1*** which starts with 9 pulses of 484,28 kHz (fc/28) followed by 8 pulses of 423,75 kHz (fc/32) **:EOF when using one subcarrier** The EOF comprises 3 parts: 1: a logic ***0*** which starts with 8 pulses of 423,75 kHz (fc/32), 2: an unmodulated time of 18,88µs (256/fc) 3: 24 pulses of 423,75 kHz (fc/32), an unmodulated time of 56,64 µs (768/fc) **:EOF when using two subcarriers** The EOF comprises 3 parts: 1: a logic ***0*** which starts with 8 pulses of 423,75 kHz (fc/32) followed by 9 pulses of 484,28 kHz (fc/28), 2: 24 pulses of 423,75 kHz (fc/32), 3: 27 pulses of 484,28 kHz (fc/28).	
M1-Tag: 11c	Bit sync sequence	N/A	
M1-Tag: 12	Scrambling (for Spread spectrum systems)	N/A	
M1-Tag: 13	Bit transmission order	Least significant bit first	
M1-Tag: 14	Reserved		
M1-Tag: 15	Polarization	N/A (Near Field)	
M1-Tag: 16	Minimum tag receiver bandwidth	see transmitted / received signals 6.1.4	

6.2 MODE 2: Physical layer, collision management system and protocols for MODE 2

MODE 2 is not interoperable with any other MODE defined in this part of ISO/IEC 18000.

MODE 2 is non-interfering with any other MODE defined in this part of ISO/IEC 18000.

The performance and conformance measurement aspects for MODE 2 shall be conformant with the relevant clauses of Technical Reports (ISO/IEC TR 18046 and ISO/IEC TR 18047-3, respectively).

6.2.1 Normative aspects: physical and media access control (MAC) parameters: interrogator to tag link

See Table 3 — Physical and media access control (MAC) parameters: interrogator to tag link

Table 3 — Physical and media access control (MAC) parameters: interrogator to tag link

Ref.	Parameter Name	Description
M2-Int: 1	Operating frequency range	13,56 MHz ± 7 kHz
M2-Int: 1a	Default operating frequency	13,56 MHz
M2-Int: 1b	Operating channels (for Spread spectrum systems)	N/A
M2-Int: 1c	Operating frequency accuracy	± 100 parts per million Japan: ± 50 parts per million
M2-Int: 1d	Frequency hop rate (for Frequency Hopping [FHSS] systems)	N/A
M2-Int: 1e	Frequency hop sequence (for Frequency Hopping [FHSS] systems)	N/A
M2-Int: 2	Occupied channel bandwidth	The modulation sidebands are very low in amplitude but spread wide. They satisfy the ETSI, and FCC regulations. (Occupied channel bandwidth is not regulated in the Japan Radio Law).
M2-Int: 2a	Minimum receiver bandwidth	Suitable to receive tag channel or channels of interest.
M2-Int: 3	Interrogator transmit maximum Magnetic Field Strength	The interrogator shall not generate a field higher than 12 A/m in any part of the volume where a tuned ISO card sized tag may be present
M2-Int: 3a	Minimum Operating Field Strength	Minimum Operating Field Strength: 150 mA/m for tuned ISO card sized tags. Application requirements may result in a different minimum operating field strength. In that case the minimum operating field strength shall be declared by the tag manufacturer.
M2-Int: 4	Interrogator transmit spurious Emissions	See below.
M2-Int: 4a	Interrogator transmit spurious emissions, in-band (for Spread spectrum systems)	N/A
M2-Int: 4b	Interrogator transmit spurious emissions, out-of-band	Compliant with ETSI, ARIB STD-T82 and FCC maximum out of Band allowed field strength.
M2-Int: 5	Interrogator transmitter spectrum mask	Compliant with ETSI, ARIB STD-T82 and FCC maximum out of Band allowed field strength.

Ref.	Parameter Name	Description
M2-Int: 6	Timing	See below.
M2-Int: 6a	Transmit to receive turn around time (the time between the end of a command and when the interrogator is first ready to receive a reply)	0 - 50 µs
M2-Int: 6b	Receive to transmit turn around time "(the time between the end of a reply and when the interrogator can transmit a command)	Greater than 0 µs
M2-Int: 6c	Dwell time or Interrogator transmit power on ramp	0 - 10 µs
M2-Int: 6d	Decay time or Interrogator transmit power down ramp	0 - 10 µs
M2-Int: 7	Modulation	PJM (Phase Jitter Modulation) min. level +/- 3,0 ° max. level +/- 4,0 °
M2-Int: 7a	Spreading sequence (for frequency hopping [FHSS] systems)	N/A
M2-Int: 7b	Chip rate (for Spread spectrum systems)	N/A
M2-Int: 7c	Chip rate accuracy (for Spread spectrum systems)	N/A
M2-Int: 7d	Modulation index	N/A (System is not amplitude modulation)
M2-Int: 7e	Duty cycle	N/A
M2-Int: 7f	FM Deviation	N/A
M2-Int: 8	Data coding	Modified Frequency Modulation (MFM) (see Figure 4 — Command MFM encoding and timing of binary 000100)
M2-Int: 9	Bit rate	423,75 kbit/s
M2-Int: 9a	Bit rate accuracy	Synchronous to the carrier frequency.
M2-Int: 10	Interrogator transmit modulation accuracy	N/A
M2-Int: 11	Preamble	Includes an MFM encoding violation
M2-Int: 11a	Preamble length	16 bits
M2-Int: 11b	Preamble waveform	The command flag defines the start of a command and the bit interval timings. The flag comprises three parts: 1: a synchronising string of 8 bits of valid MFM data. 2: an MFM encoding violation not present in normal MFM data. The violation consists of a sequence of 5 state changes separated by a 1 bit interval, a 2 bit interval, a 1,5 bit interval and a 2 bit interval. The edge of the fifth (last) transition defines the beginning of a bit interval. 3: a trailing MFM 0 defining the end of a flag and the beginning of the command. (See Figure 1 — Two possible command flags)
M2-Int: 11c	Bit sync sequence	See M2 Int: 11b
M2-Int: 11d	Frame sync sequence	See M2 Int: 11b

Ref.	Parameter Name	Description
M2-Int: 12	Scrambling (for Spread spectrum systems)	N/A
M2-Int: 13	Bit transmission order	LSB first
M2-Int: 14	Wake-up process	Interrogator Talks First (ITF) System. Tag cannot respond unless it receives valid command from interrogator.
M2-Int: 15	Polarization	N/A

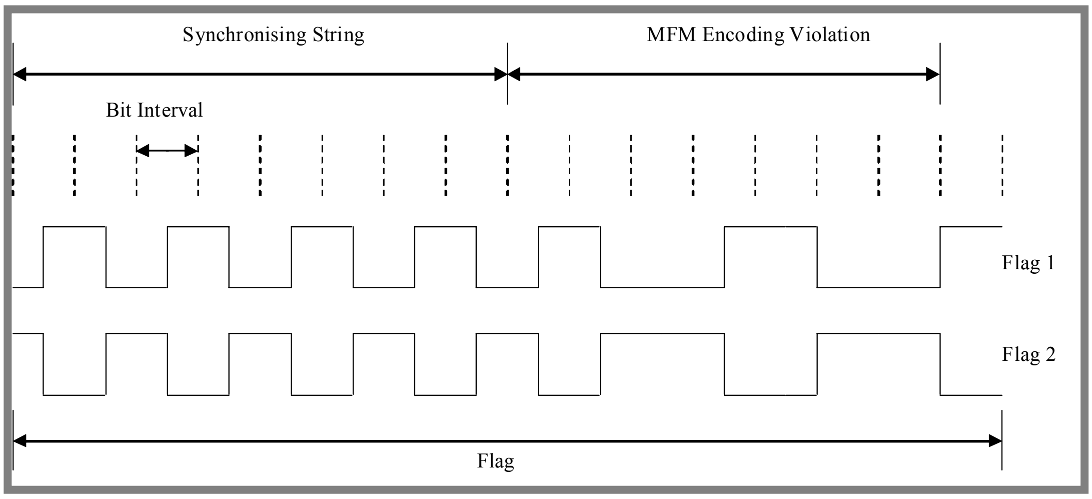


Figure 1 — Two possible command flags

6.2.2 Tag to interrogator link

See Table 4 — Tag to interrogator link and Figure 2 — Two possible reply flags.

Table 4 — Tag to interrogator link

Ref.	Parameter Name	Description
M2-Tag:1	Operating frequency range	13,56 MHz ± 3,013 MHz
M2-Tag: 1a	Default operating frequency	N/A (System does not rely on a default operating frequency)
M2-Tag: 1b	Operating channels (for Spread spectrum systems)	Multi-frequency operating system where tags can select from 8 reply channels. Tags transmit the whole of their reply using a selected channel. The tag may use one of eight subcarriers. The subcarriers are derived by division of the powering field's frequency.

Channel	Frequency kHz	Division Ratio
A	969	14
B	1233	11
C	1507	9
D	1808	7,5
E	2086	6,5
F	2465	5,5
G	2712	5
H	3013	4,5

Ref.	Parameter Name	Description
M2-Tag: 1c	Operating frequency accuracy	Synchronous to the carrier frequency.
M2-Tag: 1d	Frequency hop rate (for Frequency hopping [FHSS] systems)	Tags transmit the whole of a reply on a selected channel
M2-Tag: 1e	Frequency hop sequence (for Frequency hopping [FHSS] systems)	Reply channel is randomly selected by the tag.
M2-Tag: 2	Occupied channel bandwidth	106 kHz for each of 8 reply channels
M2-Tag: 3	Transmit Maximum Magnetic Field Strength	In accordance with local regulations
M2-Tag: 4	Transmit spurious emissions	N/A
M2-Tag: 4a	Transmit spurious emissions, In-band (for Spread spectrum systems)	N/A
M2-Tag: 4b	Transmit spurious emissions, Out-of-band	N/A
M2-Tag: 5	Transmit spectrum mask	N/A
M2-Tag: 6	Timing	See below.
M2-Tag: 6a	Transmit to receive turn around time (the time between the end of a reply and when a the tag is first ready to receive a command)	0 – 200 μs

Ref.	Parameter Name	Description
M2-Tag: 6b	Receive to transmit turn around time (time between the end of the last bit of a read command and a tag beginning the first bit of a tag reply)	77 to 88 μs
M2-Tag: 6c	Dwell time or transmit power on ramp	N/A
M2-Tag: 6d	Decay time or Transmit power down ramp	N/A
M2-Tag: 7	Modulation	Loadmodulation
M2-Tag: 7a	Spreading sequence (for Frequency hopping [FHSS] systems)	Tags transmit the whole of a reply in a randomly selected or an interrogator selected channel.
M2-Tag: 7b	Chip rate (for Spread spectrum systems)	N/A
M2-Tag: 7c	Chip rate accuracy (for Spread spectrum systems)	N/A
M2-Tag: 7d	On-Off ratio	N/A
M2-Tag: 7e	Subcarrier frequency	8 subcarrier frequencies available:

Channel	Frequency kHz	Division Ratio
A	969	14
B	1233	11
C	1507	9
D	1808	7,5
E	2086	6,5
F	2465	5,5
G	2712	5
H	3013	4,5

Ref.	Parameter Name	Description
M2-Tag: 7f	Subcarrier frequency accuracy	Synchronous to the carrier frequency.
M2-Tag: 7g	Subcarrier modulation	BPSK (Binary phase shift keying)
M2-Tag: 7h	Duty cycle	N/A
M2-Tag: 7I	FM Deviation	N/A
M2-Tag: 8	Data coding	MFM (Modified frequency modulation)(see Figure 7 — Reply MFM encoding and timing of binary 000100)
M2-Tag: 9	Bit rate	105,9375 kbit/s
M2-Tag: 9a	Bit rate accuracy	Synchronous to the carrier frequency.
M2-Tag: 10	Tag transmit modulation accuracy (for Frequency hopping [FHSS] systems)	N/A
M2-Tag: 11	Preamble	Includes an MFM encoding violation
M2-Tag: 11a	Preamble length	16 bits

Ref.	Parameter Name	Description
M2-Tag: 11b	Preamble waveform	The reply flag defines the start of a reply and the bit interval timings. The flag comprises three parts: 1: a synchronising string of 9 bits of valid MFM data. 2: a MFM encoding violation not present in normal data. The violation consists of a sequence of 4 state changes separated by a 2 bit interval, a 1,5 bit interval and a 2 bit interval. The edge of the fourth transition defines the beginning of a bit interval 3: a trailing zero defining the end of the flag. (See Figure 2 — Two possible reply flags)
M2-Tag: 11c	Bit sync sequence	See M2-Tag:11b
M2-Tag: 11d	Frame sync sequence	See M2-Tag:11b
M2-Tag: 12	Scrambling (for Spread spectrum systems)	N/A
M2-Tag: 13	Bit transmission order	LSB first
M2-Tag: 14	Reserved	
M2-Tag: 15	Polarization	N/A
M2-Tag: 16	Minimum tag receiver bandwidth	See Figure 3 — Command modulation scheme

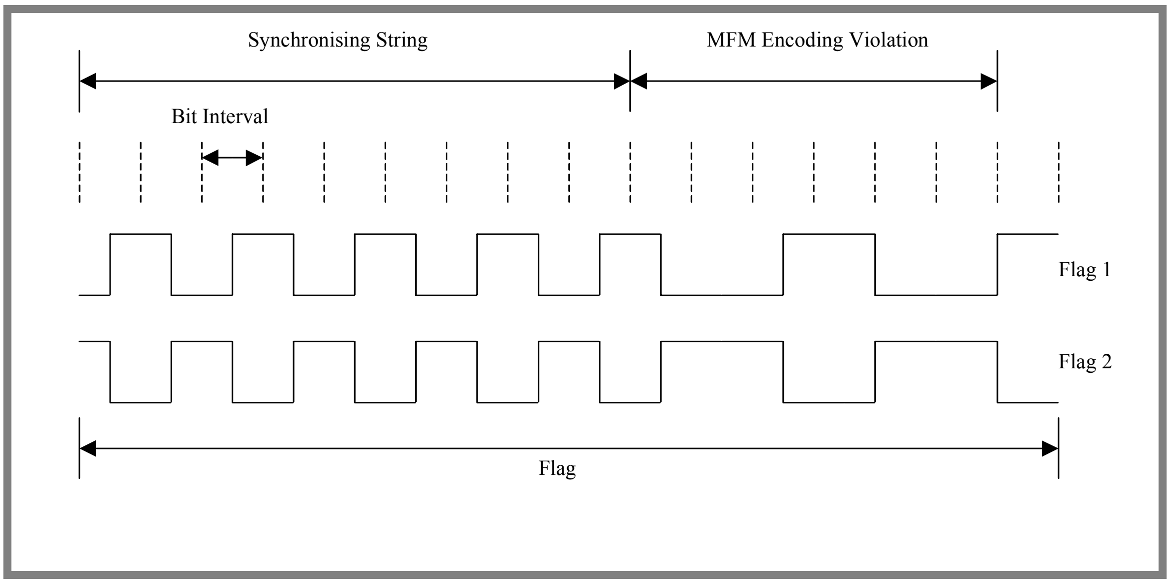


Figure 2 — Two possible reply flags

6.2.3 Description of operating method

6.2.3.1 General

This clause defines the characteristics of the air interface between the interrogator and the tag. It details the transfer of power and the bi-directional communications between the interrogator and the tag.

Tags may be passive. In which case, power is transferred from the interrogator to the tag by a high frequency magnetic field using coupled antennae in the interrogator and the tag. The powering field frequency f_c is 13,56 MHz $\pm$ 7 kHz. The interrogator shall be capable of powering a tag at all positions inside the interrogator operating volume.

Commands are transmitted from the interrogator to the tag by Phase Jitter Modulation (PJM) of the powering field. PJM transmits data as very small phase changes in the powering field. These phase changes are between +/- 3,0 ° and +/- 4,0 ° There is no reduction in the transfer of power to the tag during PJM. The bandwidth of PJM is no wider than the original double-sided spectrum of the data. The sideband levels and data rates are decoupled. This allows the sideband levels be set at any arbitrary level without affecting the data rate. Phase Jitter Modulation is described in Annex B.

The command data rate is 423,75 kbit/s encoded using Modified Frequency Modulation (MFM).

An interrogator can be full or half duplex. If an interrogator is full duplex then it can transmit commands whist simultaneously receiving multiple tag replies. Tags are half duplex.

Tags reply to the interrogator by inductive coupling whereby the voltage across the tag antenna coil is modulated by a subcarrier. The subcarrier is derived from division of the powering field’s frequency.

Tags can select from one of eight subcarrier frequencies between 969 kHz and 3013 kHz. The reply data rate is 105,9375 kbit/s encoded using Modified Frequency Modulation (MFM) and modulated onto the subcarrier as Binary Phase Shift Keying (BPSK).

To ensure that tags replying on different channels are simultaneously received tag replies are band limited to reduce data and subcarrier harmonic levels.

6.2.3.2 Communications signal interface interrogator to tag

Commands are transmitted from the interrogator to the tag by PJM of the powering field. The command data rate is 423,75 kbit/s and all commands are MFM encoded prior to the PJM modulator.

6.2.3.2.1 Modulation

Commands are transmitted from the interrogator to the tag by PJM of the powering field. In PJM data is transmitted as very small phase reversals in the powering field. This allows the sideband levels be set at any arbitrary level without affecting the data rate.

The tags operate as intended with minimum PJM sideband levels that comply with the appropriate FCC and ETSI regulations.

The PJM phase shift waveform of the interrogator magnetic field is described in Figure 3 — Command modulation scheme and Table 5 — Command Modulation Parameters.

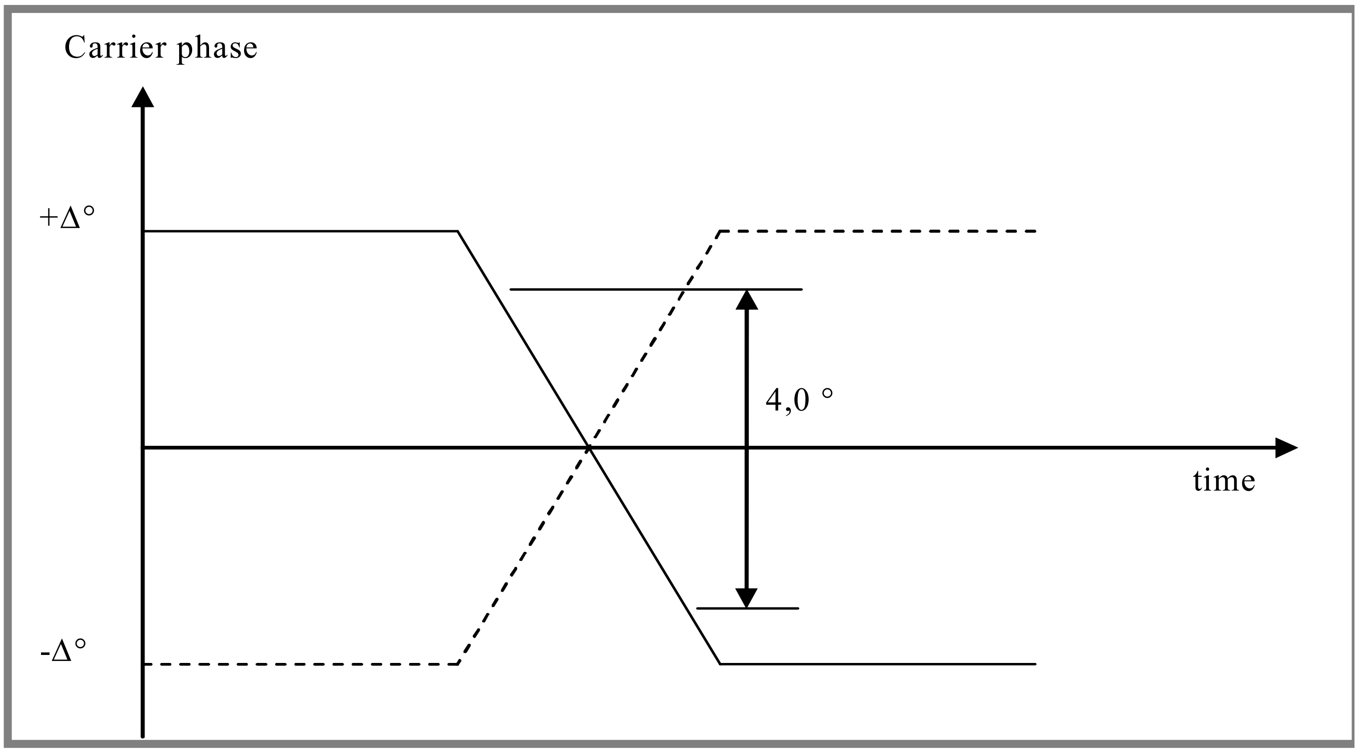


Figure 3 — Command modulation scheme

Table 5 — Command Modulation Parameters

Parameter	Minimum	Maximum	Units
phase shift +Δ°, -Δ°	3,0	4,0	deg.
time after which the phase shift shall remain above the 95% point	0	1,0	µs

NOTE: 4,0° of the total phase shift must be completed in 500 ns or less. The phase shift cannot exceed the final value by 5% at any time. The rise and fall phase shifts shall not have a difference greater than 40 ns at any position between the 10% to 90% points. The phase shift magnitude shall remain constant during the whole command.

6.2.3.2.2 Data rate and data coding

The encoded command data rate is 423,75 kbit/s (f_c / 32). The period of a bit interval used for encoding command data is 2,3599 µs.

All commands are MFM encoded prior to the PJM modulator. Bits are encoded using MFM encoding rules. MFM has the lowest bandwidth occupancy of the binary encoding methods. The bit value is defined by a change in state. These encoding rules are defined as follows:

- A bit 1 is defined by a state change at the middle of a bit interval.
- A bit 0 is defined by a state change at the beginning of a bit interval.
- Where a bit 0 immediately follows a bit 1 there is no state change.

An example of command MFM encoding of the binary string 000100 is shown in Figure 4 — Command MFM encoding and timing of binary 000100.

Typically the edges shown in Figure 4 — Command MFM encoding and timing of binary 000100 and Figure 5 — MFM Encoding and timing for two possible command flags shall be synchronised to the powering field. If not synchronised then these edges shall be generated at the interrogator within +/- 0,04 µs of the times shown.

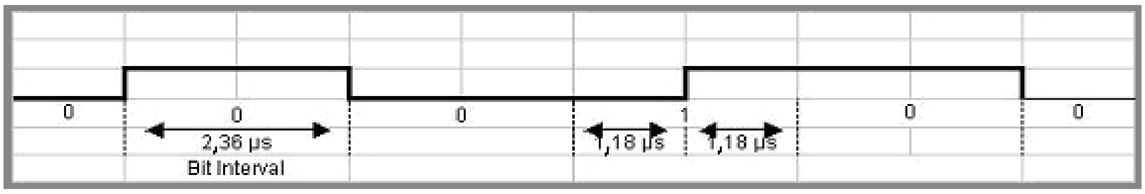


Figure 4 — Command MFM encoding and timing of binary 000100

6.2.3.2.3 Interrogator to tag frames

The command flag defines the start of a command and the bit interval timings. The flag comprises three parts:

- A synchronising string of 8 bits of valid MFM data.
- An MFM encoding violation not present in normal MFM data. The violation consists of a sequence of 5 state changes separated by a 1 bit interval, a 2 bit interval, a 1,5 bit interval and a 2 bit interval.
- A trailing MFM 0 defining the end of a flag and the beginning of the command.

The synchronising string, encoding violation and a trailing zero for two possible command flags are illustrated in Figure 5 — MFM Encoding and timing for two possible command flags.

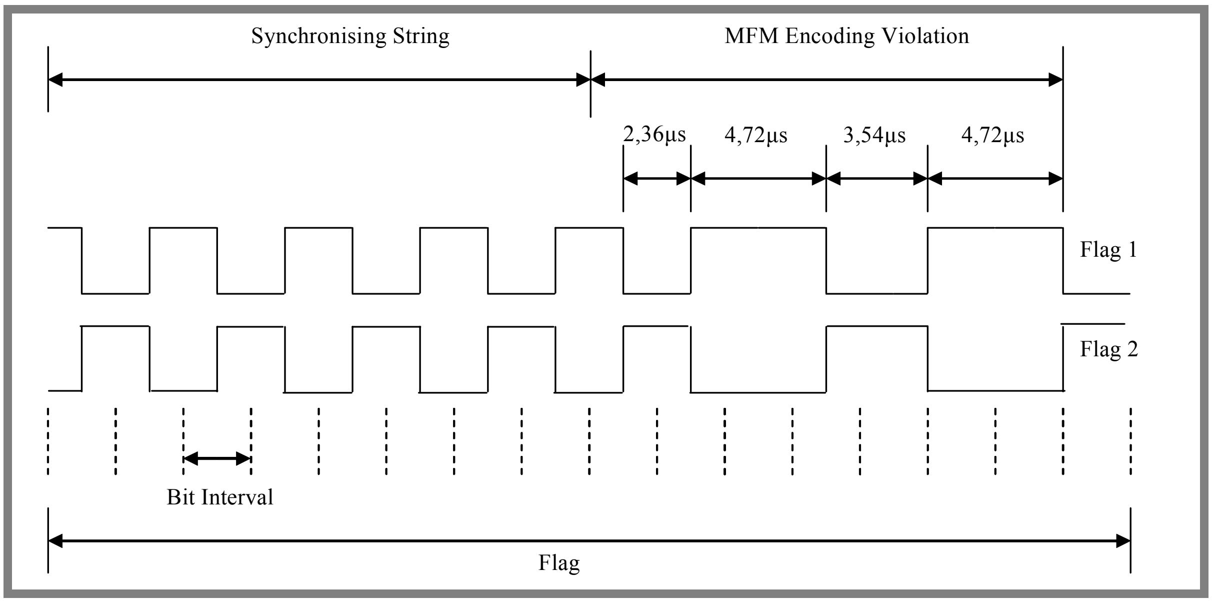


Figure 5 — MFM Encoding and timing for two possible command flags

6.2.3.3 Communications signal interface tag to interrogator

Replies are MFM encoded prior to the BPSK modulator. Tags reply using one of eight selectable modulated subcarriers.

6.2.3.3.1 Subcarriers

The tag may use one of eight subcarriers. The subcarriers are derived by division of the powering field's frequency. The channel frequencies and division ratios are tabulated in Table 6 — Channel frequencies and division ratios.

Table 6 — Channel frequencies and division ratios

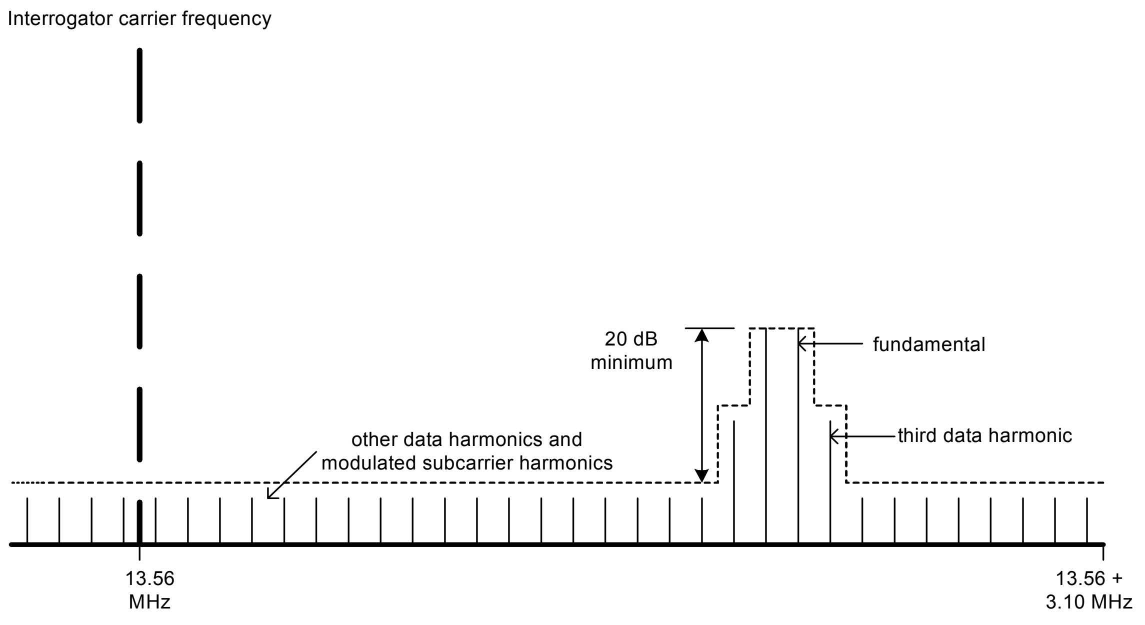
Channel	Frequency kHz	Division Ratio
A	969	14
B	1233	11
C	1507	9
D	1808	7,5
E	2086	6,5
F	2465	5,5
G	2712	5
H	3013	4.5

6.2.3.3.2 Modulation

The tag replies to the interrogator by inductive coupling whereby the voltage across the tag antenna coil is modulated by a subcarrier. Modulation is based on impedance modulation. Encoded data is modulated on to the subcarrier as BPSK modulation.

Systems shall respect local regulations. In order to ensure that tags replying on different channels are simultaneously received, all tag replies shall be band limited to reduce data and subcarrier harmonic levels.

NOTE: For example, the tag reply spectrum for 105,9375 kbit/s MFM encoded all zeros data stream, may be defined to be within the tag reply mask provided in Figure 6 — Tag reply mask below. Such mask limitation would be appropriate for ITU region 1, 2 and meets Japanese regulations.


Interrogator carrier frequency
20 dB minimum
fundamental
third data harmonic
other data harmonics and modulated subcarrier harmonics
13.56 MHz
13.56 + 3.10 MHz

Note. This figure shows the upper side band of the Tag reply. The same mask shall be applied to the lower sideband.

Figure 6 — Tag reply mask

6.2.3.3.3 Data rate and data encoding

The encoded reply data rate is 105,9375 kbit/s (fc / 128). The period of a bit interval used for encoding command data is 9,4395 μs. The modulation used is BPSK.

Tags reply using one of eight selectable modulated subcarriers. The subcarriers are derived by division of the powering field's frequency. Replies are encoded using MFM and modulated onto the subcarrier as Binary Phase Shift Keying (BPSK).

MFM encoding rules are shown in 6.2.3.2.2. An example of reply MFM encoding of the binary string 000100 is shown in Figure 7 — Reply MFM encoding and timing of binary 000100. Typically the edges shown in Figure 7 — Reply MFM encoding and timing of binary 000100 and Figure 8 — MFM Encoding and timing for two possible reply flags shall be synchronised to the powering field. If not synchronised then these edges shall be generated at the tag within +/- 0.15 μs of the times shown.

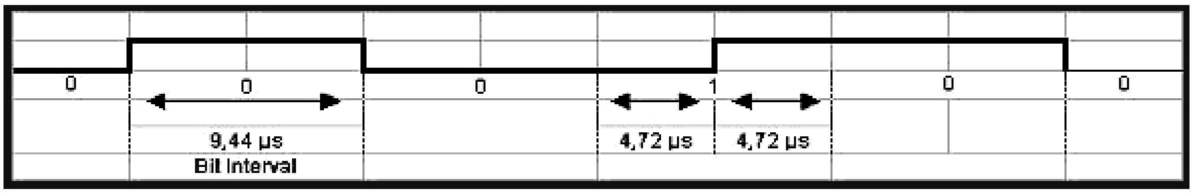


Figure 7 — Reply MFM encoding and timing of binary 000100

6.2.3.3.4 Tag to interrogator frames

The reply flag defines the start of a reply and the bit interval timings. The reply flag comprises three parts:

- A synchronising string of 9 bits of valid MFM data.
- A MFM encoding violation not present in normal data. The violation consists of a sequence of 4 state changes separated by a 2 bit interval, a 1,5 bit interval and 2 bit interval. The edge of the fourth transition defines the beginning of a bit interval.
- A trailing 0.

A synchronising string, encoding violation and a trailing zero for two possible reply flags are illustrated in Figure 8 — MFM Encoding and timing for two possible reply flags.

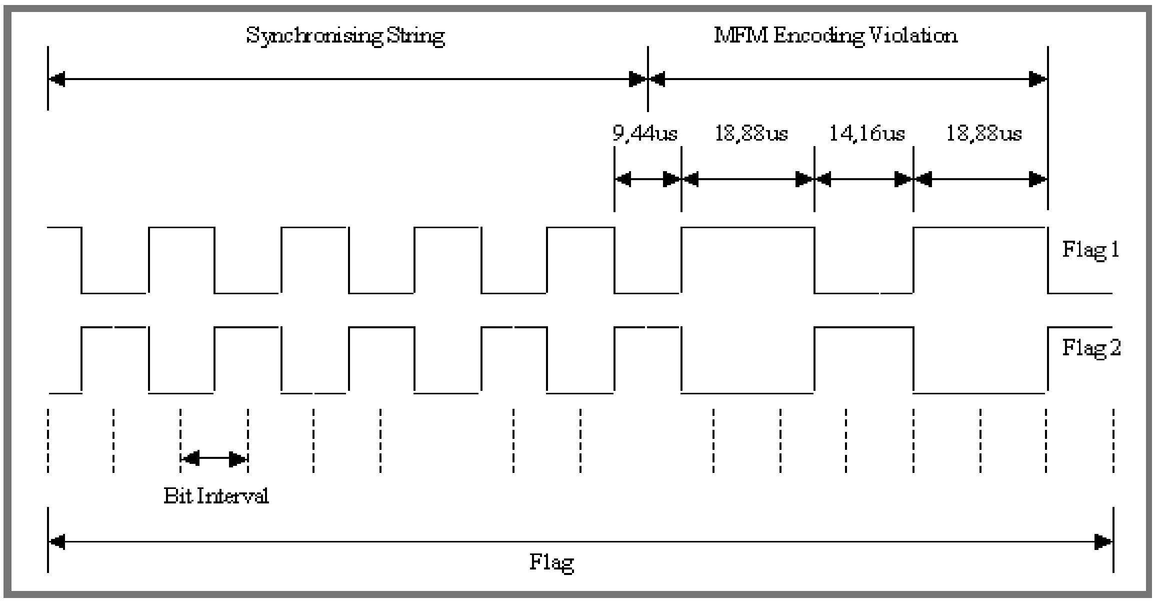


Figure 8 — MFM Encoding and timing for two possible reply flags

6.2.4 Protocol parameters

See Table 7 — Protocol parameters.

Table 7 — Protocol parameters

Ref.	Parameter Name	Description
M2-P: 1	Who talks first	Interrogator Talks First (ITF) System. Tag cannot respond unless it receives valid command from interrogator.
M2-P: 2	Tag addressing capability	Yes, tags can be addressed individually or as groups.
M2-P: 3	Tag ID	
M2-P: 3a	Tag ID length	64 bits (32 bit specific identifier, 16 bit application group identifier and 16 bit manufacturing identifier).
M2-P: 3b	Tag ID format	See clauses 6.2.5.6 to 6.2.5.9
M2-P: 4	Read size	Two bytes (16 bits) to maximum memory size.
M2-P: 5	Write size	Minimum and Maximum Write size is dependent upon memory technology, according to instruction defined in 6.2.5.14.
M2-P: 6	Read transaction time	1,282ms + 150 μs per 16 bits (2 bytes).
M2-P: 7	Write transaction time	1,282ms + 75,5 μs per 16 bits (2 bytes), not including memory erase and write time.
M2-P: 8	Error detection	16 bit CRC interrogator to tag, 32 bit tag to interrogator.
M2-P: 9	Error correction	None.
M2-P: 10	Memory size	Product Dependant. No technical limitation.
M2-P: 11	Command structure and extensibility	Command field is 16 bits long, extendable without limit in 16 bit blocks. 8 command types, each with 16 extension types presently available.

6.2.5 Description of protocol operating method

6.2.5.1 Overview

Tags may be passive; in which case power is transferred from the interrogator to the tag by a High Frequency (HF) magnetic field using coupled antennas in the interrogator and the tag.

Dialogue between the interrogator and the tag is conducted on an Interrogator-Talks-First (ITF) basis. Following activation of the tag by a powering field the tag waits silently for a valid command. After receiving a valid command the tag transmits a reply to the command.

Phase Jitter Modulation (PJM) is used (described in 6.2.3 above).

The tag memory is expandable beyond 1 Megabit so that the system is inherently upgradeable, subject to product definition/design.

Multiple tag identification is performed using a combination of Frequency and Time Division Multiple Access (FTDMA). There are eight reply channels available for tags to use. In response to a valid command each tag randomly selects a channel on which to transmit its reply. The reply is transmitted once using the selected channel. Upon receiving the next valid command each tag randomly selects a new channel and transmits the reply using the new selected channel. This method of reply frequency hopping using random channel selection is repeated for each subsequent valid command. In addition to random channel selection the tag can randomly mute individual replies. When a reply is muted the tag will not transmit that reply. Random muting is necessary when identifying very large populations of tags. All FTDMA frequency and time parameters are defined by the command.

All commands are time stamped and tags store the first time stamp received after entering an interrogator. The stored time stamp defines precisely when the tag first entered the interrogator and provides a high resolution method of determining tag order which is decoupled from the speed of identification.

Tag temporary settings shall be stored in tag memory (for example using a technique such as Temporary Random Access Memory TRAM) that retains data contents during power outages caused by switching of the powering field in orientation insensitive interrogators.

6.2.5.2 Definition of data elements

Read data is the data read from chip memory by a valid command.

Write data is the data written to chip memory by a valid command.

Stored data is the data stored in chip memory.

Hardcode data is the data in virtual chip ROM.

6.2.5.3 RF tag memory organization

This part of ISO/IEC 18000 describes the tag memory in virtual terms only and does not intend to restrict the physical implementation of any tag’s memory.

The tag memory is split into the three areas shown in Table 8 — Tag memory areas.

Table 8 — Tag memory areas

Memory Area	Comment
manufacturing system memory area	contains all fields that are set and locked during chip/tag manufacture and manufacturing test
user system memory area	set and locked as required by the user
user memory area	set and locked as required by the user

Tags with 4 kbit or less of virtual memory will use 8 bit address and 8 bit length fields. Tags with greater than 4 kbit of virtual memory shall allow for both 8 and 16 bit address and length fields.

The tag memory includes tag identifiers, configuration and user defined fields. The virtual memory map, see Table 9 — Virtual memory map, includes the defined fields. The bit order of any defined fields is such that the least significant field bit is stored at the lower virtual memory bit address.

The virtual memory is organised and addressed as 16 bit words. MODE 2 makes provision for tag types with varying memory block sizes, where a block is one or more 16 bit words. Read commands read zero or more words. Write commands write whole words, where the number of words that can be written is determined by the memory construction.

Memory may be locked. Once locked, memory cannot be overwritten.

6.2.5.4 Virtual tag memory

Table 9 — Virtual memory map shows the virtual memory layout of the tag.

Table 9 — Virtual memory map

Word no	memory type	comment	register	bit number															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	manufacturing system memory	defined fields	RFM	reserved for manufacturer															
1			MC	manufacturing code															
2			SID0	specific identifier 0															
3			SID1	specific identifier 1															
4	user system memory	defined fields	GID	application group identifier															
5			CID	conditional identifier															
6			CW	configuration word															
7	user memory	undefined if password is not required	PW0	Password 0															
8			PW1	Password 1															
9			PW2	Password 2															
10 and above		undefined fields																	

Word 10 and above shall be user memory followed by any remaining system memory.

Specific identifier 0 and password 0 are the least significant words of these multi word fields.

6.2.5.5 Lock pointer

The lock pointer is a 16 bit virtual system memory field, used to prevent tag memory from being over written. The field points to a word in memory. All complete sub-blocks of memory at addresses less than the number stored in the lock pointer cannot be over written. Interrogator commands cannot decrement the lock pointer.

6.2.5.6 Unique identification (Tag ID)

The unique identification for MODE 2 may be set permanently at manufacturing.

It shall comprise a logical 64 bit block, consistent with ISO/IEC 15963.

The physical implementation on the tag is left to the manufacturer.

The 64 logical bits are organized into three parts. See Figure 9 — Logical organization of tag ID.

	MODE 2 TAG Tag ID MASK	
MASK 1	***MASK 2***	***Mask 3***
Manufacturing code	**Application Group Identifier**	**Specific Identifier**
16 Bits	16 Bits	32 Bits
	AFI Not Determined	Serially issued binary value

Figure 9 — Logical organization of tag ID

6.2.5.7 Manufacturing code

The manufacturing code is a mask of 16 bits.

The physical operation of the system requires only that this mask is populated with a binary value.

However, in order to uniquely identify manufacturers, registration shall be in accordance with ISO/IEC 15963.

Transmission from the tag for this Mode shall be LSB first, and the data available shall conform to the disposition in Table 10 — Manufacturing code field.

Example implementation according to ISO/IEC 15963.

The manufacturing code is a 16 bit field set at manufacturing test. The encoding for the manufacturing code is shown in Table 10 — Manufacturing code field

Table 10 — Manufacturing code field

MSB	LSB
16 9	**8 1**
'E0'	manufacturer code according to ISO/IEC 7816-6

6.2.5.8 Application group identifier (AFI)

The second mask is of 16 bits.

The physical operation of the system requires only that this mask is populated with a binary value.

The purpose of this mask is to identify applications or families of tags.

It is recommended that the first octet of this mask is used to encode an AFI as defined in ISO/IEC 15961-3; if so the value of the AFI shall be as registered with the RA as defined in ISO/IEC 15961-2.

The second octect of this mask is not specified.

6.2.5.9 Specific identifier

The third mask is of 32 bits.

It shall be allocated at manufacture, and shall provide a unique binary value which shall be sequentially issued by the manufacturer and shall not be reused by that manufacturer. It shall provide a unique tag ID for that manufacturer.

The combination of this Specific Identifier value, Application Group Identifier and the Manufacturing code, shall provide a permanent and unambiguous unique tag identification.

This identifier allows a tag to be specifically identified and specifically communicated to. The manufacturing test software shall ensure that the specific identifier is incremented once each time a tag is loaded with this identifier.

6.2.5.10 Conditional identifier

In addition to the tag Tag ID the tag memory shall carry an additional mask, known as the Conditional Identifier (CID). The mask shall be 2 octets (16 bits).

The physical operation of the system requires only that this mask is populated with a binary value.

The CID may be provided at manufacture (as for example a date of manufacture) or may be provided later.

This part of ISO/IEC 18000 is not prescriptive as to whether the CID is programmed at manufacture or later, but manufacturers shall decide which of these options and shall use the CID consistently thereafter.

Example: The CID may provide a date-code to enable conditional access/disbarment according to the data condition, thus, for example tags manufactured before the CID, or after the CID could be eliminated or selected.

6.2.5.11 Configuration word

The configuration word is a 16 bit field set by the user. The tag configuration is usually set to suit the application. The encoding for the configuration word is shown in Table 11 — Configuration word field.

Table 11 — Configuration word field

Bit number	Field	State	Description
Bit 0 to 6	RFU		Shall be set to ‘0’
Bit 7 to 14	Custom Settings		
Bit 15	Password	0	password not required
Bit 15	Password	1	password required

6.2.5.12 Password

The password field is a 48 bit identifier set by the user. It is used to provide a level of security for memory access. If the tag is configured to ‘password not required’ the password memory space is free for user memory. If the tag is configured to ‘password required’ the password cannot be read.

6.2.5.13 User memory

The user sets the user memory.

6.2.5.14 Hardcode

Hardcode is formatted as 16 bit words and is included in some tag types. This part of ISO/IEC 18000 describes the hardcode in virtual terms only and does not intend to restrict the physical implementation of hardcode. The hardcode defines the tag parameters, including memory size and memory block size.

In a normal reply all hardcode words are transmitted first, followed by the time stamp and then the rest of the reply. The MSB of all hardcode words shall be set to ‘1’. The encoding for the hardcode is shown in Table 12 — Hardcode field.

Table 12 — Hardcode field

Bit number	Field	State	Description
Bit 0 to 6	parameter/ function	00_h	memory size in 4 word units (LSB)
		01_h	memory size in 4 word units (MSB)
		02_h	memory block size in words
		03_h	memory sub-block size in words
		04_h	memory erase + write time in 100 μs units
		05_h to 07_h	reserved for custom hardcodes

Bit number	Field	State	Description
Bit 7 to 14	code/value	00_h to FF_h	hexvalue or code associated with parameter or function
Bit 15	MSB		shall be set to ‘1’

For the purposes of this document 'memory erase + write time' shall be the time between the end of the last bit of a write command and the beginning of the first bit of a tag reply. Also 'memory size' shall be from word 0 to the top of user memory.

For example a tag with:

- 8192 bits of memory (512 words, 128 * 4 word units)
- 4 word memory block size
- 1 word memory sub-block size
- 4 ms memory erase + write time (40 * 100 μs units)

would have the following hardcode:

- memory size $C000_h$
- block size 8202_h
- sub-block size 8083_h
- erase + write time 9404_h

In the above example:

- The memory size MSB is not used, as all the value bits are ‘0’.
- A sub-block size of 1 word and a block size of 4 words indicates that 1, 2, 3, and 4 memory word writes are allowed where all writes shall be within a block, (no writes across block boundaries).

The block size origin is word 0.

Memory can be locked at sub-block intervals, if the sub-block interval is not given then the memory can only be locked at block intervals.

6.2.5.14.1 Default tag parameters

If the memory size is not defined in hardcode then the memory size is 64 words.

If the memory block size is not defined in hardcode then the memory block size is 4 words.

If the memory sub-block size is not defined in hardcode then the memory sub-block size is 1 word.

If the memory erase + write time is not defined in hardcode then the memory erase + write time is 5.0 ms or less.

6.2.5.15 Block security status

Refer to Clauses 6.2.5.5 Lock Pointer and 6.2.5.12 Password.

6.2.5.16 Description of operating methodology

Tag and interrogator communications is based on the Interrogator-Talks-First system. The tag will only respond to a command if the command received is valid. Commands are generally used to identify tags and to read, write and lock memory. Commands also determine the reply type (short or normal) and the reply mode (fixed channel or random channel etc.). A short reply is used to speed up communications, whereas a normal reply includes all hardcode and some system memory.

All tags, irrespective of their types can be identified using a universal group command. The universal group command can set the tags to transmit a normal reply. The hardcode and system memory included in a normal reply provides sufficient information so that a user can send other valid commands to the tags.

The various reply modes are selected to suit the interrogator types (single or multi channel) and to speed up communication depending upon tag population within the interrogator operating volume.

6.2.5.16.1 Methods

See 6.2.5.1 and 6.2.5.2

6.2.5.16.2 Command format

The format of command fields is shown in Table 13 — Command fields. All fields are transmitted least significant bit first. For multi word fields the least significant bit of the least significant word defines the least significant bit of the field. All commands defined in this subsection are Mandatory as classified in 6.0 above.

Table 13 — Command fields

Code	Field	Bit	Comment
F	Flag	16	MFM violation sequence
Cd	command	16	command field
Cn	command number	16	command number field
SS	Specific identifier	32	identifier field
G	application group identifier	16	identifier field
Ci	conditional identifier	16	identifier field
PPP	password	48	identifier field
R	read address and length	16	8 bit address and 8 bit length fields for memory read
W	write address and length	16	8 bit address and 8 bit length fields for memory write
Ra	read address	16	16 bit address field for memory read
Rl	read length	16	16 bit length field for memory read
Wa	write address	16	16 bit address field for memory write
Wl	write length	16	16 bit length field for memory write
D	write data	16	data to be written
C	CRC	16	validation CRC

The format for valid commands is shown in ,Table 14 — Valid command format where:

The password field shall only be provided if required by the tag.

The read/write commands shown include a single word write.

Table 14 — Valid command format

Command type	Start fields	Identifier fields	Address and length fields	Data	CRC
group read	F [Cd] Cn	G Ci	[R] or [Ra Rl]		C
specific read	F [Cd] Cn	SS	[R] or [Ra Rl]		C
group read/write	F [Cd]Cn	G Ci |PPP|	[R W] or [Ra Rl Wa Wl]	D	C
Specific read/write	F [Cd]Cn	SS |PPP|	[R W] or [Ra Rl Wa Wl]	D	C

The minimum length command is 7 words (112 bits).

For a zero length write no data will be provided. For a multi-word length writes each word to be written is followed by a CRC. The command format for a two word write is shown in Table 15 — Valid multi-word read/write command format.

Table 15 — Valid multi-word read/write command format

Command Type	Start Fields	Identifier Fields	Address and Length Fields	Data	CRC	Data	CRC
group read/write	F [Cd] Cn	G Ci |PPP|	[R W] or [Ra Rl Wa Wl]	D	C	D	C
Specific read/write	F [Cd] Cn	SS |PPP|	[R W] or [Ra Rl Wa Wl]	D	C	D	C

For all write commands if the start, identifier or address and length fields are invalid no data will be written to tag memory.

If any CRC is invalid the tag will not reply.

6.2.5.16.3 Command fields

6.2.5.16.3.1 Flag field

The flag field contains a MFM violation that is not present during normal data. The field indicates the beginning of a command.

6.2.5.16.3.2 Command field

The encoding of the command field is shown in Table 16 — Command field bit encoding.

Table 16 — Command field bit encoding

Bit number	Field	State	Description
bit 0	command type	0	read command
		1	read/write command
bit 1	identifier type	0	specific command
		1	application group conditional (group) command
bit 2	reply type	0	short reply
		1	normal reply
bit 3	fixed/random	0	fixed channel reply

Bit number	Field	State	Description
		1	random channel reply
bits 4 to 6	channel/mute ratio	000	[fixed channel A] or [random channel unmuted]
		001	[fixed channel B] or [random channel ½ muted]
		010	[fixed channel C] or [random channel ¾ muted]
		011	[fixed channel D] or [random channel $^{7}/_{8}$ muted]
		100	[fixed channel E] or [random channel $^{31}/_{32}$ muted]
		101	[fixed channel F] or [random channel $^{127}/_{128}$ muted]
		110	[fixed channel G] or [random channel $^{511}/_{512}$ muted]
		111	[fixed channel H] or [random channel fully muted]
bit 7	address and length	0	8 bit address and 8 bit length fields
		1	16 bit address and 16 bit length fields
bit 8	test command enable	0	normal command
		1	proprietary test command
bits 9 to 13	RFU		shall be set to '0' for mandatory commands
bit 14	command class	0	mandatory command
		1	custom command
bit 15	command extension	0	indicates to the tag that this word is the last command field
		1	indicates to the tag that the following word is a command field extension

For a command to be valid the command field shall be set equal to one of the combinations shown in Table 16 — Command field bit encoding. In addition the address and length field (bit 7) shall be set to '0' for tags with 4 kbit or less of virtual memory The function provided by bit 8 is only intended for use during wafer/chip testing and must be disabled by the manufacturer so that only normal commands are available to a user.

The following describes the fields within the command field.

6.2.5.16.3.3 Command type

The command type field determines if the command is a read or a read/write.

A read command is used to read tag memory. For fast tag identification the read length field can be set to zero.

A read/write command is used to read and write to tag memory. For write only operation the read length is set to zero. To lock tag memory the write length is set to zero and the write address is set to the lowest unlocked memory address.

6.2.5.16.3.4 Identifier type

The identifier type field determines if the command is a specific or an application group conditional command.

Specific commands are used to identify and communicate to individual tags.

Application group conditional (group) commands are used to identify and communicate to a group of tags that meet a conditional test or all groups of tags that meet a conditional test.

6.2.5.16.3.5 Reply type

The reply type field determines if the tag reply is short or normal.

A short reply is used to minimise communication time.

A normal reply is used to when the interrogator requires the hardcode and system memory data to be included in a reply.

6.2.5.16.3.6 Fixed/random

The fixed/random field determines if the tag reply is on a fixed selected channel or a random selected channel.

6.2.5.16.3.7 Channel/mute ratio

The channel/mute ratio field determines either the channel selected or the mute ratio selected.

This field is linked to the fixed/random field described above. If fixed channel is selected then the channel/mute ratio field determines the actual channel to be used in the reply. If random channel is selected then the channel/mute ratio field determines the mute ratio.

For valid random channel commands:

- If unmuted is selected the tag will transmit replies repetitively on randomly selected channels.
- If ½ to $^{511}/_{512}$ muted is selected the tag will randomly choose to transmit (unmuted) or not transmit (muted) individual replies. The muted ratio supplied in the command (½ muted, ¾ muted etc.) determines the probability of the reply being muted.
- If fully muted is selected the tag will not reply and the tag will be set to the temporary mute state. While the tag is in the temporary mute it will only respond to interrogator command with a new interrogator identifier, see 6.2.5.16.3.9.

6.2.5.16.3.8 Address and length

The address and length field defines whether a command includes 8 bit address and 8 bit length fields or 16 bit address and 16 bit length fields.

6.2.5.16.3.9 Command number

The command number is used to set a local time stamp and identify interrogators.

The encoding for the command number is shown in Table 17 — Command number fields.

Table 17 — Command number fields

Bit number	Field	State	Description
Bit 0 to 7	local time stamp		set as required by the interrogator
bits 8 to 14	Interrogator identifier		set as required by the interrogator
Bit 15	MSB		shall be set to ‘0’

The most significant byte of the command number shall be other than 00_h.

When a tag enters a new interrogator the tag will store the command number from the first valid command received. The tag shall store this number during brief powered down periods. All tag replies include this stored number, which is called the time stamp. The least significant byte of the command number is periodically incremented by the interrogator and used in subsequent commands. The tag does not update the time stamp. The time stamp therefore indicates when a tag first received a valid command.

A tag determines that it has entered a new interrogator when the most significant byte of the command number (included in all valid commands) and the time stamp are different, or if the tag detects that it has been powered down for longer than a defined period.

In a multi tag scenario if two or more tags reply at the same time on the same channel no reply will be received. If the population is large it is possible that a tag may have to transmit a few times before it selects a unique channel and is therefore received. There is also a chance that the tag will select a unique channel for its first transmission. Thus the order in which tags are received cannot be used to determine the order in which they entered the interrogator. However the time stamp included in the reply will give the correct tag order.

6.2.5.16.3.10 Tag identifiers

6.2.5.16.3.11 Specific identifier field

The specific identifier is used to communicate to an individual tag. For a command to be valid the specific identifier in the command shall be set equal to the specific identifier stored in tag to be communicated to.

6.2.5.16.3.12 Application group identifier field

The application group identifier is used to communicate with tags from the same application group or from all application groups. For a command to be valid the application group identifier in the command shall be set equal to either $FFFF_h$ or to the application group identifier stored in the tags to be communicated to.

If the command identifier is set $FFFF_h$ and the rest of the command is valid then all application groups will be communicated to.

6.2.5.16.3.13 Conditional identifier field

The conditional identifier is used to communicate to tags that meet a conditional test. For a command to be valid the conditional identifier in the command shall be less than or equal to the conditional identifier stored in the tag.

6.2.5.16.3.14 Password field

The password is used to restrict writes to tag memory. For a command to be valid the password shall only be provided if the tag is configured to password required, and then only for read/write commands. For a read/write command to a password protected tag to be valid the password in the command shall be set equal to the password stored in the tag.

6.2.5.16.3.15 Address and length fields

The command field determines if the command includes 8 bit address and 8 bit length fields or 16 bit address and 16 bit length fields.

The address and length fields define the start address and length in words for memory reads and writes. For a command to be valid, the address and length field shall only select from valid memory addresses. Valid memory addresses are as follows:

- For a memory read of tags configured to password not required:
  word 0 to the maximum user memory word is available.
- For a memory read of tags configured to password required word:
  10 to the maximum user memory word is available.
- For a memory write:
  the address stored in the lock pointer to the maximum user memory word.

- When locking tag memory:
  the address stored in the lock pointer and above.
- To lock tag memory the write length is set to zero and the write address is set to the lowest unlocked memory address, if the command is valid the tag will set the lock pointer equal to the write address.
- In addition, for a command to be valid the address and length fields shall be set as required by different tag types to allow for any block addressing restrictions.

If clearly defined in the manufacturers chip data book:

- valid memory addresses for memory reads can include system memory that is above the user memory area
- valid memory addresses for memory reads can include words 0 to 6 for tags configured to password required.

6.2.5.16.3.16 8 bit address and 8 bit length fields

Where the command field selects 8 bit address and 8 bit length fields the address and length field shall be encoded as shown in Table 18 — 8 bit address and 8 bit length fields memory reads and Table 19 — 8 bit address and 8 bit length fields for memory writes.

Table 18 — 8 bit address and 8 bit length fields memory reads

Bit number	Field	State	Description
Bits 0 to 7	8 bit address field		set as required by the user
Bits 7 to 15	8 bit length field		set as required by the user

Table 19 — 8 bit address and 8 bit length fields for memory writes

Bit number	Field	State	Description
Bits 0 to 7	8 bit address field		set as required by the user
Bits 7 to 15	8 bit length field		set as required by the user

6.2.5.16.3.17 16 Bit address and 16 bit length fields

Where the command field selects 16 bit address and 16 bit length fields the address and length field shall be 16 bits each.

6.2.5.17 Write data

The write data is the data to be written to the tag.

6.2.5.18 CRC

All command CRCs are calculated from the end of the flag field.

The 16 bit command CRC algorithm used is the IBM "Synchronous Data-Link Control" (SDLC) polynomial, standardised by CCITT for use in the X.25 packet-switching protocol, and is implemented in most serial communications tags and is embodied in ISO/IEC 13239. The CRC shall be calculated as per the definition in ISO/IEC 13239.

The algorithm used is as follows:

$$g(X) = X^{16} + X^{12} + X^{5} + 1$$

The generation algorithm departs slightly from the basic CRC16 technique, in that the generator is pre-loaded with ‘1’ bits rather than ‘0’ bits. Additionally the checkbits are inverted before transmission. Consequently, a valid message is not recognised by a remainder of zero (as is the case with CRC16), but by a specific constant.

An example interrogator command and CRC is given below:

flag field	
command field	0000
command number	1234
specific identifier 0	1234
specific identifier 1	5678
read address and length	1001
CRC	8C16

The method and an example of this 16 bit CRC is given in Annex M.

6.2.5.19 Reply format

Reply fields are shown in Table 20 — Reply fields. All fields are transmitted least significant bit first. For multi byte fields the least significant bit of the least significant byte defines the least significant bit of the field.

Table 20 — Reply fields

Code	Field	Bits	Comment
F	Flag	16	MFM violation sequence
H	Hardcode	16	hardcode field
T	time stamp	16	identifier field
L	lock pointer	16	identifier field
M	manufacturing code	16	identifier field
SS	specific identifier	32	identifier field
G	application group identifier	16	identifier field
Ci	conditional identifier	16	identifier field
Co	configuration word	16	identifier field
D	read data	16	read data
CC	CRC	32	validation CRC

Table 21 — Valid reply format shows the format for valid replies.

Table 21 — Valid reply format

Reply Type	Start Fields	System Memory Fields	Data	CRC
Normal	F [H] T	L M SS G Ci Co	[D]	CC
Short	F T	SS	[D]	CC

The minimum length reply is 96 bits.

The reply type is determined by the interrogator command.

6.2.5.20 Reply fields

6.2.5.20.1 Flag field

The flag field contains a MFM violation that is not present during normal data. The field indicates the beginning of a reply.

6.2.5.20.2 Hardcode fields identifier fields

If the tag design includes hardcode then all hardcode data is sent in normal replies. The MSB of the hardcode is set to ‘1’ and the MSB of the following timestamp is set to ‘0’. The interrogator can detect the end of hardcode data by examining the MSB of received words.

6.2.5.20.3 Time stamp field

The time stamp is used to provide superior tag order identification. The time stamp is set equal to the command number included in the first valid command received after the tag entered a new interrogator.

6.2.5.20.4 System memory fields

The reply system memory fields are as per the system memory stored in tag virtual memory.

6.2.5.21 Read data

The read data is the data requested by a valid command.

6.2.5.22 CRC field

The reply CRC is calculated from the end of the start flag field.

The reply CRC is the 32 bit Ethernet CRC. Its properties are similar to the 16 bit IBM CRC used in the command path, in that the register is loaded with ones, the output word is inverted and the final computation results in a specific constant rather than zero.

The algorithm used is as follows:

$$g(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$$

An example tag reply and CRC is given below.

flag field	
time stamp	1234
specific identifier 0	1234
specific identifier 1	5678
read data	ABCD
CRC	E8C5 8742

The method and an example of this 32 bit CRC is given in Annex N.

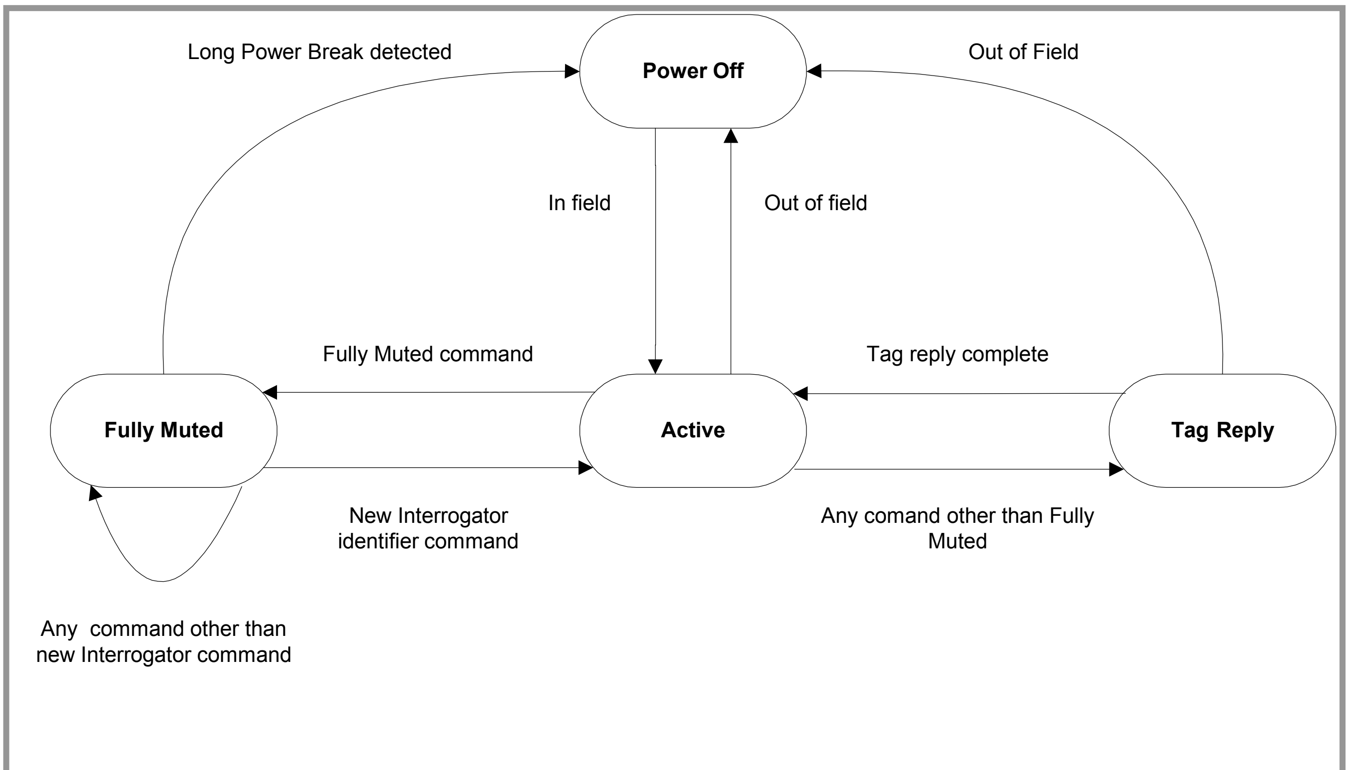
6.2.5.23 RF tag states

6.2.5.23.1 State diagram

A tag can be in one of four following states:

- Power Off
  The tag is in the Power Off state when it cannot be activated by the interrogator. Power off is also achieved as a result of an LPB detection.
- Active
  The tag is in the active state when it is activated by the interrogator. In this state it shall process any interrogator command.
- Tag Reply
  The tag is in the tag Reply state when it has received a valid command. If the tag remains powered then it will stay in this state until it has completed the reply, after which it will return to the active state.
- Fully Muted
  The tag is in the fully muted state when it has received a valid fully muted command. If the tag remains powered then it will stay in this state until it receives a valid new interrogator command.

The transition between these states is specified in Figure 10 — Tag state transition diagram.


Long Power Break detected
Power Off
Out of Field
In field
Out of field
Fully Muted command
Tag reply complete
Fully Muted
Active
Tag Reply
New Interrogator identifier command
Any comand other than Fully Muted
Any command other than new Interrogator command

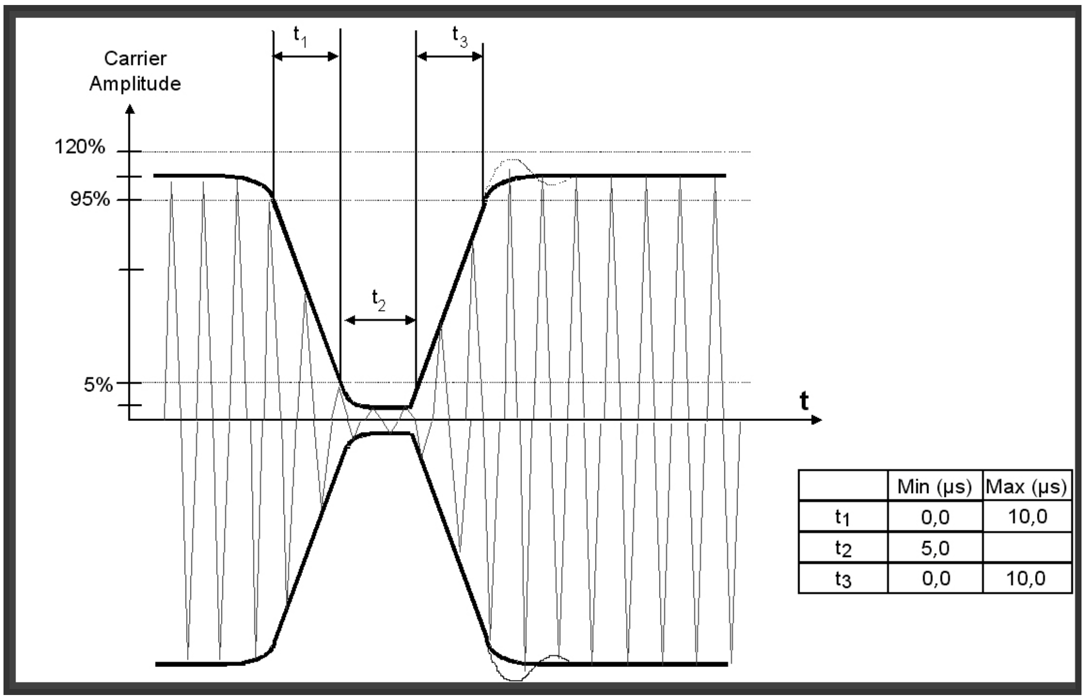
Note For invalid commands the Tag will remain in its current state

Figure 10 — Tag state transition diagram

6.2.5.23.2 Short power breaks

A break in the interrogator carrier received by the tag of 5 µs or greater but less than a Long Power Break, see below, shall be considered as a Short Power Break (SPB). If a SPB occurs then the tag will initialise and then go to the active state.

Figure 11 — Interrogator powering field breaks describes interrogator carrier breaks.

•


	Min (µs)	Max (µs)
t_1	0,0	10,0
t_2	5,0	
t_3	0,0	10,0

Figure 11 — Interrogator powering field breaks

6.2.5.23.3 Long power breaks

The tag includes TRAM or equivalent, see below. If a tag is powered down for a long time the TRAM values may be corrupted. Using a Long Power Break (LPB) detector the tag will a detect power break long enough to corrupt the TRAM after which all TRAM values will be cleared.

TRAM or equivalent must remain valid for at least 50 ms. Therefore a Long Power Break shall be 50 ms or greater

6.2.5.23.4 TRAM

This part of ISO/IEC 18000 requires temporary storage of data. A technique called TRAM is referred to in several clauses as a means of achieving this requirement. Other techniques that achieve the same function may be used. This clause describes TRAM and its functionality for those who choose to use this technique. TRAM, as referred to in this Mode, is volatile memory, such as SRAM or DRAM, which has been designed to maximise the storage time after power has been removed from the memory. If power is applied before the TRAM has discharged the memory will be automatically refreshed.

TRAM provides instant write times and short term memory storage. Fast write times are required for setting the fully mute state and writing the time stamp and other temporary settings.

Note at power up the tag inspects the state of the LPB detector, if a long power break has occurred the tag will clear all TRAM.

6.2.5.23.5 Fully muted state

When tagged items are moving on a conveyor the position and orientation of the attached tags are uncontrolled. In order for the conveyor interrogator to power and communicate with tags independent of tag position and orientation it shall generate an interrogator field that is switched cyclically between the X, Y and Z direction orthogonal axes. A consequence of cycling the field is that tags periodically lose power. During these power outages any information held in volatile memory in the tag shall not be lost.

Special regard shall been given to management of power outages arising from the operation of orientation insensitive interrogators. For example where multiple tags are being identified there is a requirement for identified tags to be temporarily muted or silenced so as not to interfere with the identification of any remaining tags.

A technique, such as Temporary Random Access Memory (TRAM), shall be used to retain the temporary mute state during these power outages.

6.2.6 Collision management parameters

See Table 22 — Table of collision management parameters.

Table 22 — Table of collision management parameters

Ref.	Parameter Name	Description
M2-A: 1	Type (probabilistic or deterministic)	Probabilistic.
M2-A: 2	Linearity	Linear for populations less than 10 000 tags, polynomial for populations larger than 10 000 tags.
M2-A: 3	Tag inventory capacity	Greater than 32 000 per application group/conditional identifier
M2-A: 4	Multiple tag identification rate	100 tags per 150 ms

6.2.7 Description of collision management parameters operating method (informative)

6.2.7.1 General description

In this Mode multiple tag identification is performed using a combination of Frequency and Time Division Multiple Access (FTDMA).

There are eight reply channels available for tags to use. In response to a valid command each tag randomly selects a channel on which to transmit its reply. The reply is transmitted once using the selected channel. Upon receiving the next valid command each tag randomly selects a new channel and transmits the reply using the new selected channel. This method of reply frequency hopping using random channel selection is repeated for each subsequent valid command.

In addition to random channel selection the tag can randomly mute individual replies. When a reply is muted the tag will not transmit that reply. Random muting is necessary when identifying very large populations of tags. Once a tag has been identified it is temporarily muted by command and will then only respond as described in 6.2.5.16.3.7.

All FTDMA frequency and time parameters are defined by command. FTDMA provides superior performance over single frequency TDMA solutions, because multiple tag replies can be simultaneously received on different channels.

In addition to the multiple tags that can be expected in an interrogator there may also be a large population of old or expired tags. For an application that uses disposable tags the population of old and expired tags can be much greater that the population of current tags. A high speed identification RFID system shall be able to identify current tags whilst excluding or ignoring old and expired tags.

In this system tags include a conditional identifier. Each tag's conditional identifier field is programmed when issued. The field can be programmed with a date-time stamp, which will be tested against a conditional identifier transmitted in each command. Tags will only respond to commands if the conditional identifier test is met. In this way old and expired tags are excluded from the identification process.

If an interrogator is full duplex then it can command tags while simultaneously receiving replies from up to eight other tags, each tag transmitting on one of eight unique channels.

6.2.7.2 Reply channels

The system uses eight reply channels between 969 kHz and 3 013 kHz. Different reply modes are used to maximise the tag identification rates for different interrogator types and tag populations. The reply mode used by a tag is selected by interrogator command.

6.2.7.3 Reply modes

6.2.7.3.1 Fixed channel reply mode

If an interrogator selects fixed channel reply mode the tag will transmit its complete reply once on the channel selected by the interrogator. This mode can be used for single channel interrogators where the tag population is one.

It can also be used in multi channel interrogators to command identified tags to reply on different fixed channels. Using this mode up to eight tags can be received simultaneously.

6.2.7.3.2 Random channel reply mode

6.2.7.3.2.1 Unmuted

If the interrogator selects unmuted random channel reply mode, tags will transmit the complete reply once on a channel randomly selected by the tag. This mode can be used by multiple channel interrogators for medium tag populations.

This mode can also be used by single channel interrogators for small tag populations. In a single channel interrogator all tag transmissions will clash if fixed channel reply mode is selected with a multiple tag population. Using unmuted random channel reply mode for a single channel interrogator is similar to a TDMA system. Eventually each tag will transmit on the interrogator’s channel while all other tags are transmitting on other channels.

6.2.7.3.2.2 Random muted

In the random muted/channel reply mode tags randomly choose to either mute or unmute individual replies. For an unmuted reply the tag randomly selects the reply channel. This mode can be used by multiple channel interrogators for large tag populations.

The ratio of muted replies to possible replies can be varied by interrogator command between ½ and $^{511}/_{512}$. The ratio is ideally increased as the tag population increases. The muted ratio controls the average number of tags replying during a reply period. This method allows for the identification of many thousands of tags simultaneously presented to the interrogator. The muted ratio reduces the average number of tags transmitting to manageable levels.

6.2.7.3.2.3 Fully muted reply mode

The interrogator can set the tag to fully muted reply mode. In this mode the tag will not reply to commands from the same interrogator and therefore will not clash with other tag replies. This mode can be used for multiple tag populations to improve tag identification rates. The tag will exit the fully muted mode when it enters a new interrogator.

A tag determines that it has entered a new interrogator based on data included in the command, or if the tag detects that it has been powered down for longer than a defined period.

6.2.7.4 Random number generator (RNG)

To generate random numbers for reply channels and muted replies the tag uses a RNG equivalent to or better than a 32 bit maximal length linear shift register.

To stop excessively long mute sequences that occur with such a PRBS generator the tag may include means to force an unmuted reply after a defined number of muted replies.

- When ½ muted is selected the maximum consecutive muted replies shall be 3.
- When ¾ muted is selected the maximum consecutive muted replies shall be 7.
- When $^{7}/_{8}$ muted is selected the maximum consecutive muted replies shall be 15.
- When $^{31}/_{32}$ muted is selected the maximum consecutive muted replies shall be 63.

Channel G is preferred for simple single channel interrogators. To identify multiple tags with such an interrogator, tags are commanded to random channel reply mode or random channel and random muted reply mode. To prevent excessively long sequences without a transmission on channel G the tag may include means to force a reply on channel G after a sequence of 15 unmuted replies that occur on channels other than G.

The counter value or equivalent used to force the above replies shall use a storage technique, such as TRAM or equivalent, in-order to retain values during short power breaks, see 6.2.5.23.4.

6.2.7.5 Command parameters

Refer to 6.2.5.16.2.

6.2.7.6 Request processing by the RF tag

Tags will only respond to valid commands. Tag function is fully defined by the command, see 6.2.5.16.2. The command defines the muting and channel selection behaviour of the tag. Tags will randomly select reply muting and a reply channel when directed by the command. Random selection is done using a PRBS generator resident in the chip. Once a tag has been fully muted it will not respond to any further normal commands.

6.2.7.7 Explanation of an collision management sequence

In this system multiple tag identification is performed using a combination of Frequency and Time Division Multiple Access (FTDMA).

In response to a valid command each tag randomly selects a channel on which to transmit its reply. The reply is transmitted once using the selected channel. Upon receiving the next valid command each tag randomly selects a new channel and transmits the reply using the new selected channel.

This method of reply frequency hopping using random channel selection is repeated for each subsequent valid command.

In addition to random channel selection the tag can randomly mute individual replies. When a reply is muted the tag will not transmit that reply. Random muting is necessary when identifying very large populations of tags. Once a tag has been identified it is temporarily muted by command.

All FTDMA frequency and time parameters are defined by command. FTDMA provides superior performance over single frequency TDMA solutions, because multiple tag replies can be simultaneously received on different channels.

Reading data when multiple tags are present takes advantage of the full duplex operation of the interrogator (if available). Data from up to 8 tags can be received simultaneously on the 8 channels when the interrogator specifically commands tags to reply on different channels.

Probability calculations are required to evaluate the average number of tags that will be identified after each valid read command. For a group of *n* tags with *r* channels available and a muting ratio of ***m*,** where *m* is the probability that a tag will transmit a reply then the average number of tags *N* identified after each valid read command is:

$$N = n.m.\left(\frac{r-1}{r}\right)^{(n.m-1)}$$

The highest identification rate is achieved when the number of tags replying at any time is equal to the number of available channels. The interrogator maximises the identification rate by adjusting the muting ratio so that the product:

n.m ≈ r

The identification rate *N* is plotted against tag numbers *n* from 1 to 10 000 tags for the different muting ratios available to the tag in Figure 12 — Identification rate. By suitable selection of muting ratio the identification rate can be maintained between 2 and 3 for up to 8 000 tags.

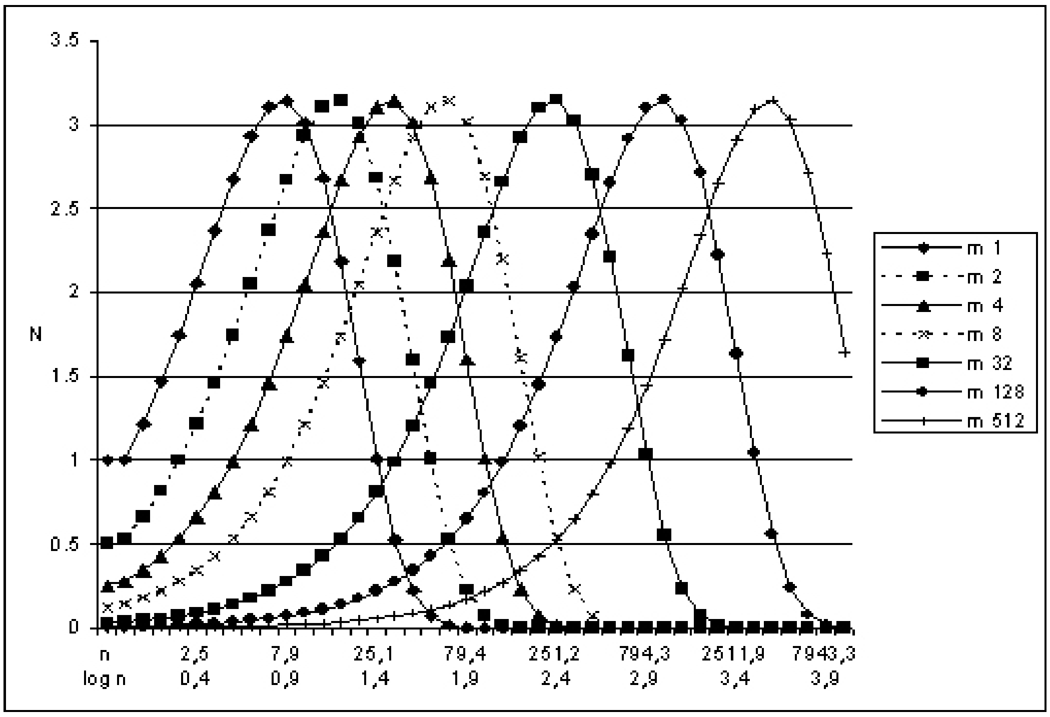


Figure 12 — Identification rate

6.2.7.8 Collision management sequence for small numbers of tags

When identifying and reading data from a small number of tags the muting ratio is set to 1. The average number of tags received is between 1 and 3 tags per read command. The sequence of operation for identifying and reading 8 tags is as follows:

a) Tags placed in the read zone of the interrogator.

b) A zero length read is issued.

c) As tags are identified they are temporarily muted.

d) The sequence is repeated until all tags are identified.

e) Data is read using a specific read command as a single operation following the identification process.

f) Commands are concatenated so that data from 8 tags is read simultaneously.

The sequence of operation for the identification of 8 tags is shown in Table 23 — Identification of 8 tags below.

Table 23 — Identification of 8 tags

Action	Result	Number of tags identified
Start		
Interrogator sends a zero length read (random channel) command	All tags reply on randomly selected channels	0
Interrogator receives 3 tag replies		3
Interrogator sends specific mute commands to each of the identified tags	The muted tags will temporarily not respond to further normal commands	
Interrogator sends a zero length read (random channel) command	The 5 unmuted tags reply on randomly selected channels	3
Interrogator receives 3 tag replies		6
Interrogator sends specific mute commands to each of the identified tags	The muted tags will temporarily not respond to further normal commands	
Interrogator sends a zero length read (random channel) command	The 2 unmuted tags reply on randomly selected channels	
Interrogator receives 2 tag replies		8
Interrogator sends specific mute commands to each of the identified tags	The muted tags will temporarily not respond to further normal commands	
End	Total time identify 8 tags is 5,772 ms. (including minimum turn around times)	8 tags identified

Data can be read as a single operation following the identification process. Reading data as a single operation is the most time efficient method and is well suited to a static tag population.

For each read command the interrogator selects a specific tag and an unused channel for the tag to reply on (Tag I on channel A etc.). The sequence of operation in detail for reading data from 8 tags is shown below in Figure 13 — Reading data from 8 tags.

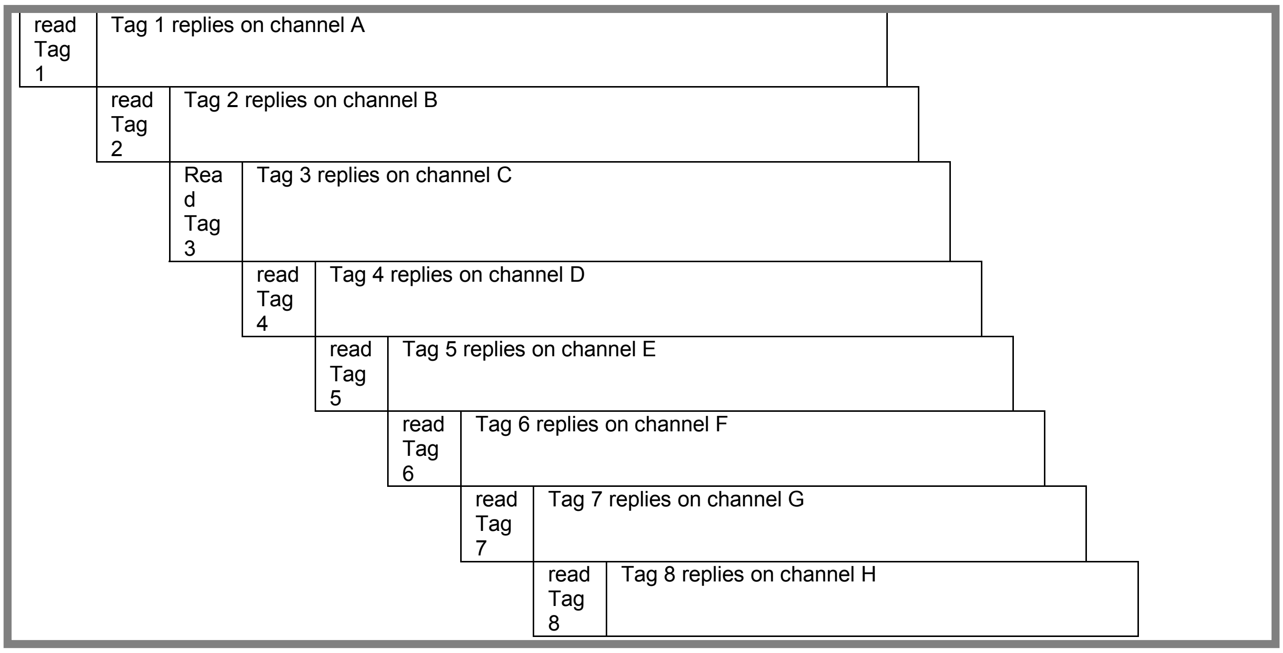

read Tag 1
Tag 1 replies on channel A
read Tag 2
Tag 2 replies on channel B
Rea d Tag 3
Tag 3 replies on channel C
read Tag 4
Tag 4 replies on channel D
read Tag 5
Tag 5 replies on channel E
read Tag 6
Tag 6 replies on channel F
read Tag 7
Tag 7 replies on channel G
read Tag 8
Tag 8 replies on channel H

Figure 13 — Reading data from 8 tags

Full duplex transmissions between the interrogator and the tags allow the interrogator to concatenate sequential commands to the tags. This allows up to 8 tags to reply simultaneously.

The total time to read 8 tags is the time for 8 read commands and a single reply from the last tag.

6.2.7.9 Collision management sequence for large numbers of tags

When identifying and reading data from a large number of tags the muting ratio is set to reduce the number of tags replying at any one time to be approximately equal to the number of channels. When correctly set the average number of tags received is between 2 and 3 tags per read command.

The sequence of operation for identifying 500 tags and reading 50 words of data from each tag is as follows:

a) 500 tags placed in the interrogator.

b) A zero length read is issued and the number of tags received is monitored.

c) The muting ratio is increased till the average number of tags received is between 2 and 3 tags per read.

d) As tags are identified they are temporarily muted.

e) The sequence is repeated till all tags are identified. The muting ratio is adjusted as tag numbers decrease so that at least 2 tags are received per read.

f) Data is read using a specific read command either as a single operation following the identification process or as a continuous process during the identification process.

g) Commands are concatenated so that data from 8 tags is read simultaneously.

The sequence of operation in detail for the identification process is shown in Table 24 — Identification of 500 tags below.

Table 24 — Identification of 500 tags

Action	**Result**	**Number of commands and replies**	**Number of tags identified**
Start			
Interrogator sends a zero length read (random channel)	All 500 tags reply on randomly selected channels		0
Mute ratio on successive commands is increased till m = 1/128	3,9 (average) tags reply on randomly selected channels.	6 reads 6 short replies	
Interrogator sends zero length reads with muting set to 1/128	175 tags identified	70 reads 70 short replies 175 mutes	175
Interrogator sends zero length reads with muting set to 1/32	245 tags identified	98 reads 98 short replies 245 mutes	420
Interrogator sends zero length reads with muting set to 1/8	60 tags identified	24 reads 24 short replies 60 mutes	480
Interrogator sends zero length reads with muting set to 1/4	15 tags identified	6 reads 6 short replies 15 mutes	495
Interrogator sends zero length read with muting set to ½	2 tags identified	1 read 1 short reply 2 mutes	497
Interrogator sends zero length read with muting set to 1	2 tags identified	1 read 1 short reply 2 mutes	499
Interrogator sends zero length read with muting set to 1	1 tag identified	1 read 1 short reply 1 mutes	500
End and Totals	Total time to identify 500 tags is less than 0,390 s (including minimum turn around times).		500 tags identified

Data can be read as a single operation following the identification process or as a continuous process during the identification process. Reading data as a single operation is the most time efficient method and is well suited to a static tag population. Reading tag data continuously during the identification process is less efficient but is well suited to a dynamic tag population.

The sequence of operation in detail for the single data reading process is shown in Table 25 — Reading data from 500 static tags below:

Table 25 — Reading data from 500 static tags

Action	**Result**	**Number of commands and replies**	**Tags read**
Start			
Specific read command (50 words) on channels A to H	Reads data from tag 1 on channel A, tag 2 on channel B,…., tag 8 on channel H	8 reads 1 read reply	8
Wait till last 264 µs of tag reply on channel A	Tag 9 ready to receive read command and will commence reply on channel A after tag 1 finishes reply		
Specific read command (50 words) on channels A to H	Reads data from tag 9 on channel A, tag 10 on channel B,…., tag 16 on channel H	1 read reply	16
Sequence of operation continues until all tags are read	Sequence repeated a total of 63 times	61 read replies	500
End and totals	Total time to read 500 static tags is 0,540 s		500 static tags read

The total time to identify and read 50 words of data from 500 tags as a single operation is less than 0,390 s + 0,540 s = 0,930 s.

The sequence of operation in detail for the continuous data reading process is shown in Table 26 — Reading data from 500 dynamic tags below. For the continuous read process the mutes tabulated in the identification sequence are not required and are instead required as part of the read sequence.

Table 26 — Reading data from 500 dynamic tags

Action	**Result**	**Number of commands and replies**	**Number of tags identified**
Start			
interrogator identifies 8 tags	8 tags ready to have data read		
Specific read command (50 words) on channels A to H	Reads data from tag 1 on channel A, tag 2 on channel B,…., tag 8 on channel H	8 reads 1 read reply	8
Mute tags after data received		1 mute	
Repeat sequence for every 8 identified tags	Sequence repeated a total of 63 times	492 reads 62 read reply 62 mutes	500
End and Totals	Total time to read 500 dynamic tags is 0,686 s		500 dynamic tags read

The total time to identify and read 50 words of data from 500 tags as a continuous operation is less than 0,258 s + 0,686 s = 0,944 s.

6.2.7.10 Timing definition

The multiple tag identification and reading process requires zero length read commands & replies, mute commands, and read commands & replies. These commands and replies require the times set out in Table 27 — Command and reply timings for multiple identification.

Table 27 — Command and reply timings for multiple identification

Command/Reply Type	Details	Time
Zero length read command	Short zero length read command used for identification	264 μs
Zero length read reply	Short length reply with zero length data	906 μs
Mute command	Temporarily mute specific tag identified by read	264 μs
Read command (W words)	Read command for ***W*** words of data	264 μs
Read reply (W words)	Read reply with ***W*** words of data	906 + 151 ***W*** μs

6.2.8 Tag order sequencing

For applications requiring item order for sortation purposes, the RFID system shall unambiguously correlate each tag to each item regardless of the presence of multiple tags or tag separation. Such resolution is resolvable with MODE 2.

Example, on a processing conveyor moving at 3,6 m/s, with tag to tag separation of 15 cm, with the effect of interrogator axis switching included, the determination of tag order needs take place in under 13,9 ms. In systems, which rely upon tag identification to provide tag order there is not enough time to perform this process.

In this situation tag order is resolved by determining the time that tags enter the interrogator. All commands are time stamped and tags store the first time stamp received after entering an interrogator. The stored time stamp is transmitted in all tag replies. Timing resolution in the millisecond level is achieved. The determination of the tag order is decoupled from the speed of identification.

A technique such as Temporary Random Access Memory (TRAM) shall be used to retain the timestamp value during short power breaks, see 6.2.5.23.4.

6.2.9 Commands

All commands are described in the Clauses above.

6.2.10 Air interface application layer

The Application layer shall be determined and controlled solely within the interrogator and shall not be carried across the air interface.

6.2.11 Optional Functionality

The interrogator and tag may optionally support the following functionality.

6.2.11.1 1 out of 4 mode encoding

In addition to MFM encoding, commands may be 1 out of 4 mode encoded. 1 out of 4 mode allows for greater modulation depth, compared with MFM, while still meeting transmit spectrum masks and still providing a data rate of 423,75 kbit/s (f_c / 32). For the 1 out of 4 mode the PJM phase changes will be between +/- 3,0 ° and +/- 4,0 °.

6.2.11.2 Data encoding using 1 out of 4 mode

1 out of 4 mode encodes bit pairs by uses the position of a symbol within a two bit period frame. For this protocol:

- an encoded symbol is transmitted as a single small PJM phase reversal of the interrogator powering field
- the 1 out of 4 mode can be slightly modified so the absence of a symbol in the two bit period frame encodes data 00, see below
- data is transmitted least significant bit pair first.

The bit pair encoding positions for 1 out of 4 mode is shown in Figure 14 — 1 out of 4 mode encoding.

Typically the edges shown in Figure 14 — 1 out of 4 mode encoding and Figure 15 — 1 out of 4 mode example shall be synchronised to the powering field. If not synchronised then these edges will be generated at the interrogator within +/- 0,04 µs of the times shown.

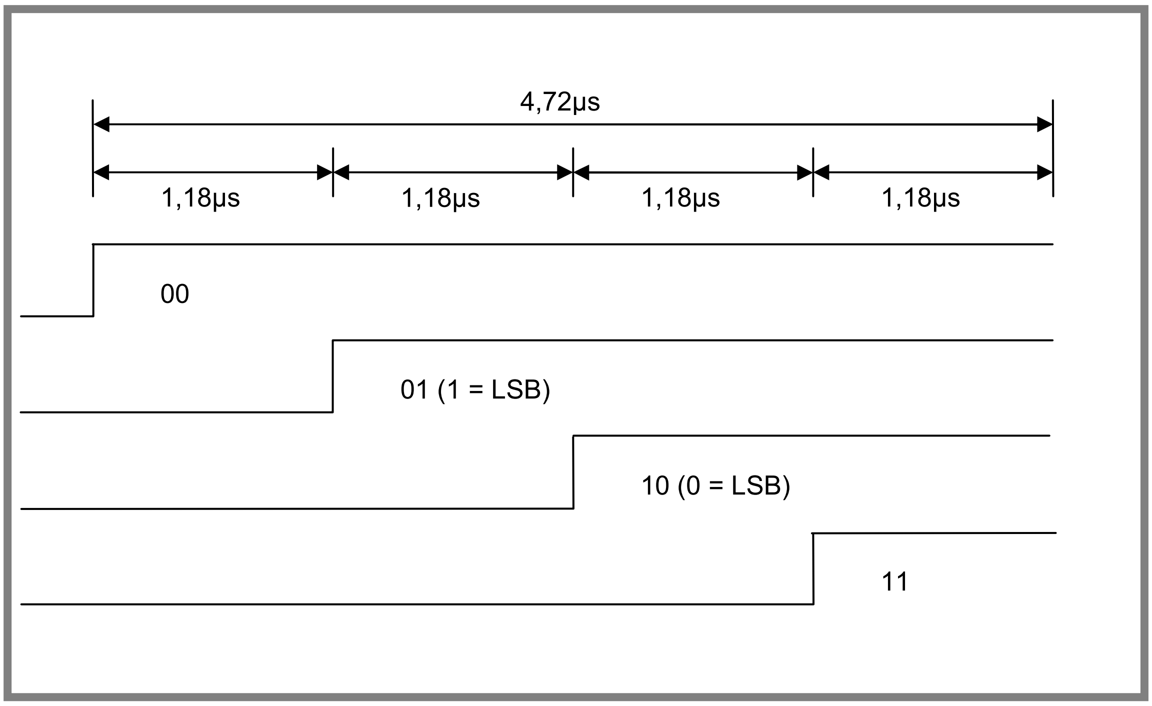


Figure 14 — 1 out of 4 mode encoding

For this protocol if the current data pair to be encoded is a 00 and the previous frame included a symbol, then no symbol shall be transmitted. If the symbol is not transmitted the resulting command transmit sidebands may be reduced. This method ensure that symbols are transmitted periodically for a long sequence of data 00s and also stops short pulses that occur with 0011 data sequences.

An example of command 1 out of 4 encoding of the binary string 11100001 is shown in Figure 15 — 1 out of 4 mode example.

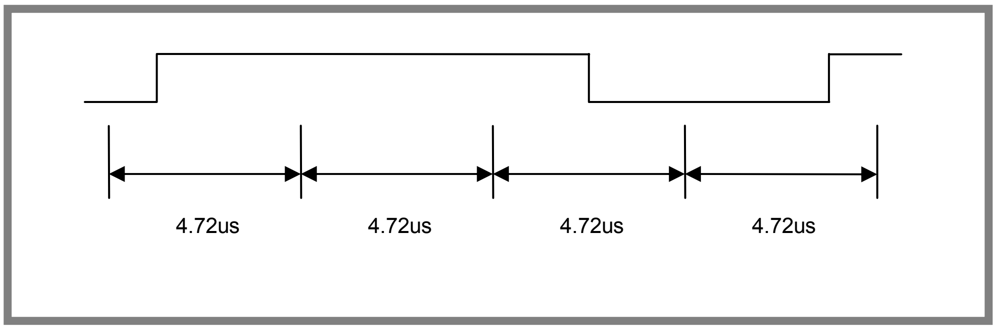


Figure 15 — 1 out of 4 mode example

6.2.11.3 Command flag for 1 out of 4 mode

The command flag defines the start of a command and the bit interval timings. The 1 out of 4 command flag is different to the MFM flag which allows tags to decode MFM or 1 out of 4 mode commands on the fly without any configuration. The flag comprises three parts:

- A synchronising string of 6 bits of valid MFM data.
- A 1 out of 4 mode encoding violation not present in normal MFM or 1 out of 4 mode encoding. The violation consists of a sequence of 6 state changes separated by a 1 bit interval, a 2 bit interval, a 2 bit interval, a 1,5 bit interval and a 2 bit interval. The edge of the sixth (last) transition defines the beginning of a bit interval.
- A trailing MFM 0 defining the end of a flag and the beginning of the command.

The synchronising string, encoding violation and a trailing zero for two possible command flags are illustrated in Figure 16 — 1 out of 4 mode encoding and timing for two possible command flags.

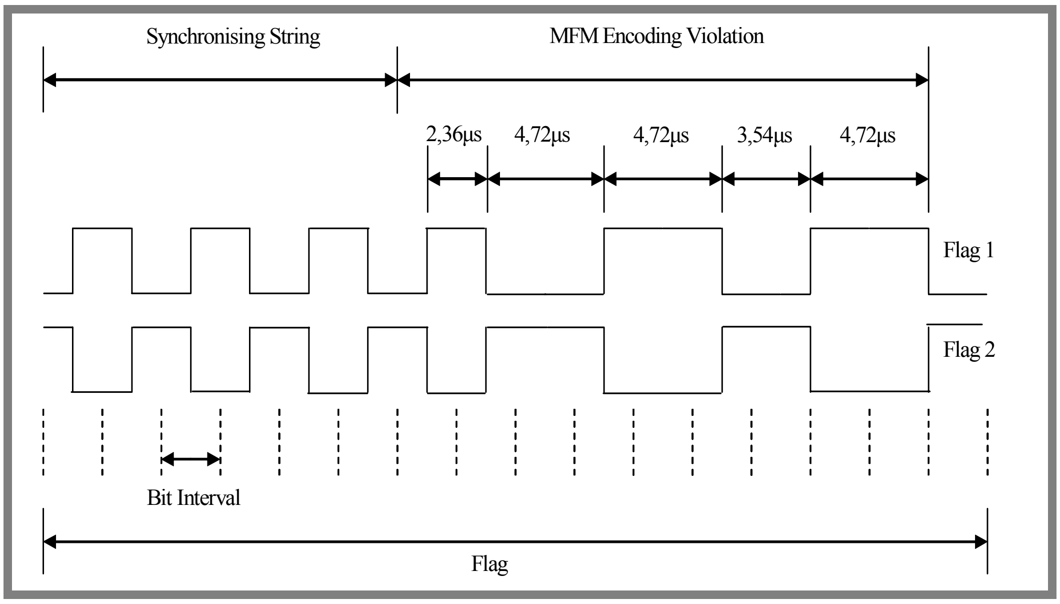


Figure 16 — 1 out of 4 mode encoding and timing for two possible command flags

6.3 MODE 3: Physical layer, collision management system and protocols for MODE 3

MODE 3 is not interoperable with any other MODE defined in this part of ISO/IEC 18000.

MODE 3 is non-interfering with any other MODE defined in this part of ISO/IEC 18000.

Annex L provides a summary of the features that are available for tags that are compliant with MODE3.

6.3.1 Protocol overview

6.3.1.1 Physical layer

An Interrogator sends information to one or more Tags by modulating an RF carrier using double-sideband amplitude shift keying (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 loadmodulated reply. Tags communicate information by loadmodulating the amplitude and/or phase of the RF carrier. The encoding format, selected in response to Interrogator commands, is either Manchester-, or Miller-modulated subcarrier or FM0 baseband. The communications link between Interrogators and Tags is half-duplex, meaning that Tags shall not be required to demodulate Interrogator commands while loadmodulating. A tag shall not respond to a mandatory or optional command using full-duplex communications.

Interrogators as well as Tags may optionally provide a physical layer with phase jitter modulation (PJM) using MFM (modified frequency modulation) encoding for Interrogator to Tag transmission and a MFM encoded, binary phase shift keying (BPSK) modulated subcarrier for Tag to Interrogator transmission.

6.3.1.2 Tag-identification layer

An Interrogator manages Tag populations using three basic operations:

1) **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.

2) **Inventory.** The operation of identifying Tags. An Interrogator begins an inventory round by transmitting a *BeginRound* command in one of two sessions. One or more Tags may reply. The Interrogator detects a single Tag reply and requests the PC/XPC word(s), UII, and if required the packet CRC-16 from the tag. Inventory comprises multiple commands. An inventory round operates in one and only one session at a time.

3) **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 may employ one-time-pad based cover-coding of the R=>T link.

6.3.2 General

6.3.2.1 MODE 3 interoperability

MODE 3 operates at 13,56 MHz. MODE 3 is not interoperable with ISO/IEC 18000-3 Mode 1 and Mode 2, but all three are expected to operate in the same environment

6.3.3 Physical layer, collision management system and protocols

6.3.3.1 Normative aspects: physical and media access control (MAC) parameters

Table 28 — Interrogator to tag (R=>T) communications and Table 29 — Tag-to-interrogator (T=>R) communications provide an overview of parameters for R=>T and T=>R communications according to this specification; for detailed requirements refer to the referenced Sub-clause. For those parameters that do not apply, or are not used in this specification, the notation "N/A" shall indicate that the parameter is "Not Applicable". The table is as specified in ISO/IEC 18000-1.

Table 28 — Interrogator to tag (R=>T) communications

Ref.	Parameter Name	Description	Sub-clause
M3-Int: 1	Operating Frequency Range	Fixed single frequency see Int:1a	6.3.3.3.1
M3-Int: 1a	Default Operating Frequency	13,56 MHz	6.3.3.3.1
M3-Int: 1b	Operating Channels (spread-spectrum systems)	N/A	N/A
M3-Int: 1c	Operating Frequency Accuracy	In accordance with local regulations	6.3.3.3.1.2.1
M3-Int: 1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	N/A	N/A
M3-Int: 1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	N/A	N/A
M3-Int: 2	Occupied Channel Bandwidth	N/A	N/A
M3-Int: 2a	Minimum Receiver Bandwidth	N/A	N/A
M3-Int: 3	Interrogator transmit maximum Magnetic Field Strength Magnetic Field Strength limits within communication zone	The interrogator shall not generate a field higher than 12 A/m in any part of the volume where a tuned ISO card sized tag may be present Maximum operating field strength: 5 A/m for ISO card sized tags as defined in ISO/IEC 7810. Test methods are defined in ISO/IEC TR 18047-3. For other label form factors, the label manufacturer shall specify the maximum operating field strength.	N/A
M3-Int: 3a	Minimum operating field strength	Minimum Operating Field Strength: 150 mA/m for tuned ISO card sized tags. Application requirements may result in a different minimum operating field strength. In that case the minimum operating field strength shall be declared by the tag manufacturer.	N/A
M3-Int: 4	Interrogator Transmit Spurious Emissions	In accordance with local regulations	6.3.3.3.1.2.5
M3-Int: 4a	Interrogator Transmit Spurious Emissions, In-Band (spread-spectrum systems)	In accordance with local regulations	6.3.3.3.1.2.5
M3-Int: 4b	Interrogator Transmit Spurious Emissions, Out-of-Band	In accordance with local regulations	6.3.3.3.1.2.5

Ref.	Parameter Name	Description	Sub-clause
M3-Int: 5	Interrogator Transmitter Spectrum Mask	In accordance with local regulations	N/A
M3-Int: 6	Timing	See below.	6.3.3.3.1.6
M3-Int: 6a	Transmit-to-Receive Turn-Around Time	Less than 73.1 µs maximum (1024/Fc -32/Fc)	6.3.3.3.1.6, Figure 40 — Link timing – Both Modes, and Table 41 — Link timing parameters
M3-Int: 6b	Receive-to-Transmit Turn-Around Time	When communicating with a tag, 151 µs minimum; 1208 µs maximum when tag is in **reply** & **acknowledged** states; no maximum limit otherwise	6.3.3.3.1.6 Figure 40 — Link timing – Both Modes, and Table 41 — Link timing parameters
M3-Int: 6c	Dwell Time or Interrogator Transmit Power-On Ramp	1500 µs, maximum settling time	6.3.3.3.1.2.6, Table 34 — Interrogator power-up waveform parameters
M3-Int: 6d	Decay Time or Interrogator Transmit Power-Down Ramp	500 µs, maximum	6.3.3.3.1.2.7, Table 35 — Interrogator power-down waveform parameters
M3-Int: 7	Modulation	ASK Method: min. 10%, max. 30% Optional PJM method : min. deviation +/- 3,0 deg. max. deviation +/- 6,0 deg.	6.3.3.3.1.2.2
M3-Int: 7a	Spreading Sequence (direct-sequence [DSSS] systems)	N/A	N/A
M3-Int: 7b	Chip Rate (spread-spectrum systems)	N/A	N/A
M3-Int: 7c	Chip Rate Accuracy (spread-spectrum systems)	N/A	N/A
M3-Int: 7d	Modulation Index	ASK Method: (A-B)/(A+B) index 10% minimum to 30% maximum	6.3.3.3.1.2.5, Figure 19 — ASK Method: Interrogator-to-tag RF envelope, Table 32 — ASK Method: RF envelope parameters
M3-Int: 7e	Duty Cycle	As specified	6.3.3.3.1.2.5
M3-Int: 7f	FM Deviation	N/A	N/A
M3-Int: 8	Data Coding	ASK Method: PIE Optional PJM Method: MFM	6.3.3.3.1.2.3, Figure 17 — ASK Method: PIE symbols and optionally Figure 18 — PJM Method: Command MFM encoding and timing of binary 000100
M3-Int: 9	Bit Rate	ASK Method: 26,7 kbit/s to 100 kbit/s (assuming equally probable data) Optional PJM method Mode: 212 kbit/s	6.3.3.3.1.2.4 and optionally 6.3.3.3.1.2
M3-Int: 9a	Bit Rate Accuracy	+/– 1%, minimum	6.3.3.3.1.2.3
M3-Int: 10	Interrogator Transmit Modulation Accuracy	As specified	6.3.3.3.1.2.3
M3-Int: 11	Preamble	Required	6.3.3.3.1.2.8
M3-Int: 11a	Preamble Length	As specified	6.3.3.3.1.2.8

Ref.	Parameter Name	Description	Sub-clause
M3-Int: 11b	Preamble Waveform(s)	As specified	Figure 22 — ASK Method: R=>T preamble and frame-sync and optionally Figure 23 — PJM Method: MFM Encoding and timing for two possible command flags
M3-Int: 11c	Bit Sync Sequence	None	N/A
M3-Int: 11d	Frame Sync Sequence	Required	6.3.3.3.1.2.8
M3-Int: 12	Scrambling (spread-spectrum systems)	N/A	N/A
M3-Int: 13	Bit Transmission Order	MSB is transmitted first	6.3.3.3.1.4
M3-Int: 14	Wake-up process	As specified	6.3.3.3.1.2.6
M3-Int: 15	Polarization	N/A	N/A

Table 29 — Tag-to-interrogator (T=>R) communications

Ref.	Parameter Name	Description	Sub-clause
M3-Tag: 1	Operating Frequency Range	13,56MHz +/- tag subcarrier frequencies as specified in Tag:7e	6.3.3.3.1.1, Table 36 — ASK Method: Tag-to-interrogator link frequencies and Table 37 — ASK Method: Tag-to-interrogator data rates
M3-Tag: 1a	Default Operating Frequency	Fixed carrier frequency as specified in Int:1a.	6.3.3.3.1.1
M3-Tag: 1b	Operating Channels (spread-spectrum systems)	N/A	N/A
M3-Tag: 1c	Operating Frequency Accuracy	As specified	6.3.3.3.1.2.1
M3-Tag: 1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	N/A	N/A
M3-Tag: 1e	Frequency Hop Sequence tags respond to interrogator signals that satisfy (frequency-hopping [FHSS] systems)	N/A	N/A
M3-Tag: 2	Occupied Channel Bandwidth	In accordance with local regulations	N/A
M3-Tag: 3	Transmit Maximum Magnetic Field Strength	In accordance with local regulations	N/A
M3-Tag: 4	Transmit Spurious Emissions	In accordance with local regulations	N/A
M3-Tag: 4a	Transmit Spurious Emissions, In-Band (spread spectrum systems)	In accordance with local regulations	N/A
M3-Tag: 4b	Transmit Spurious Emissions, Out-of Band	In accordance with local regulations	N/A
M3-Tag: 5	Transmit Spectrum Mask	In accordance with local regulations	N/A
M3-Tag: 6	Timing	See below.	6.3.3.3.1.6,
M3-Tag: 6a	Transmit-to-Receive Turn-Around Time	Less than 151 µs;	6.3.3.3.1.6, Figure 40 — Link timing – Both Modes, and Table 41 — Link timing parameters
M3-Tag: 6b	Receive-to-Transmit Turn-Around Time	75,5 µs nominal (1024/Fc)	6.3.3.3.1.6, Figure 40 — Link timing – Both Modes, and Table 41 — Link timing parameters

Ref.	Parameter Name	Description	Sub-clause
M3-Tag: 6c	Dwell Time or Transmit Power-On Ramp	Ready to receive commands in less than 1500 µs	6.3.3.3.1.2.6
M3-Tag: 6d	Decay Time or Transmit Power-Down Ramp	N/A	N/A
M3-Tag: 7	Modulation	ASK Method: by Loadmodulation	6.3.3.3.1.3.1
M3-Tag: 7a	Spreading Sequence (direct sequence [DSSS] systems)	N/A	N/A
M3-Tag: 7b	Chip Rate (spread spectrum systems)	N/A	N/A
M3-Tag: 7c	Chip Rate Accuracy (spread spectrum systems)	N/A	N/A
M3-Tag: 7d	On-Off Ratio	Tag dependent; not specified by this document	N/A
M3-Tag: 7e	Subcarrier Frequency	ASK Method: 423 kHz (fc/32) or 847 kHz (fc/16) (selected by the interrogator) Optional PJM method: Channel A 969 kHz (divide by 14) Channel B 1233 kHz (divide by 11) Channel C 1507 kHz (divide by 9) Channel D 1808 kHz (divide by 7,5) Channel E 2086 kHz (divide by 6,5) Channel F 2465 kHz (divide by 5,5) Channel G 2712 kHz (divide by 5) Channel H 3013 kHz (divide by 4,5) for tag replies	6.3.3.3.1.3.11, Table 36 — ASK Method: Tag-to-interrogator link frequencies and optionally 6.3.3.3.1.3.11, Table 38 — PJM Method: Subcarrier selection commands
M3-Tag: 7f	Subcarrier Frequency Accuracy	Carrier frequency synchronized	6.3.3.3.1.3.11, Table 36 — ASK Method: Tag-to-interrogator link frequencies
M3-Tag: 7g	Subcarrier Modulation	ASK Method: Manchester or Miller, see Tag: 9 and Table 35 — Interrogator power-down waveform parameters Optional PJM method: BPSK at 106 kbit/s	6.3.3.3.1.3.5, 6.3.3.3.1.3.7 and optionally 6.3.3.3.1.3.9
M3-Tag: 7h	Duty Cycle	FM0: 50%, nominal Subcarrier: 50%, nominal	6.3.3.3.1.3.3 6.3.3.3.1.3.5
M3-Tag: 7I	FM Deviation	N/A	N/A
M3-Tag: 8	Data Coding	ASK Method: Baseband FM0 or Manchester or Miller-modulated subcarrier (selected by the interrogator) optional PJM Method: MFM for replies	6.3.3.3.1.3.2
M3-Tag: 9	Bit Rate	ASK Method: FM0, 424 kbit/s or 848 kbit/s; Subcarrier modulated, 53 kbit/s to 212 kbit/s Optional PJM method Mode: 106 kbit/s on each reply channel	6.3.3.3.1.3.11, Table 37 — ASK Method: Tag-to-interrogator data rates and optional Table 38 — PJM Method: Subcarrier selection commands
M3-Tag: 9a	Bit Rate Accuracy	Same as Subcarrier Frequency Accuracy; see Tag:7f	N/A
M3-Tag: 10	Tag Transmit Modulation Accuracy (frequency-hopping [FHSS] systems	N/A	N/A
M3-Tag: 11	Preamble	Required	6.3.3.3.1.3.2
M3-Tag: 11a	Preamble Length	As specified	6.3.3.3.1.3.2
M3-Tag: 11b	Preamble Waveform	As specified	6.3.3.3.1.3.2,

Ref.	Parameter Name	Description	Sub-clause
M3-Tag: 11c	Bit-Sync Sequence	None	N/A
M3-Tag: 11d	Frame-Sync Sequence	ASK Method: None Optional PJM method: as specified for replies	N/A and optionally 6.3.3.3.1.3.10
M3-Tag: 12	Scrambling (spread-spectrum systems)	N/A	N/A
M3-Tag: 13	Bit Transmission Order	MSB is transmitted first	6.3.3.3.1.4
M3-Tag: 14	Reserved	Purposely left blank	N/A
M3-Tag: 15	Polarization	N/A	N/A
M3-Tag: 16	Minimum tag Receiver Bandwidth	N/A	N/A

6.3.3.2 Logical – Operating procedure parameters

Table 30 — Tag inventory and access parameters and Table 31 — Collision management parameters identify and describe parameters used by an interrogator during the selection, inventory, and access of tags according to this specification. For those parameters that do not apply to or are not used in this specification, the notation “N/A” shall indicate that the parameter is “Not Applicable”.

Table 30 — Tag inventory and access parameters

Ref.	Parameter Name	Description	Sub-clause
P:1	Who talks first	Interrogator	6.3.3.3
P:2	Tag addressing capability	As specified	6.3.3.4.1
P:3	Tag UII	Contained in tag memory	6.3.3.4.1.2
P:3a	UII Length	As specified	6.3.3.4.1.2
P:3b	UII Format	NSI < 100_h: As specified in EPCglobal Tag Data Standards (Version 1.3 and above) NSI >= 100_h: As specified in ISO/IEC 15961	6.3.3.4.1.2.2, 6.3.3.4.1.2.3, and 6.3.3.4.1.2.4
P:4	Read size	Multiples of 16-bits	6.3.3.4.11.3.2, Table 62 — *Read* command
P:5	Write Size	Multiples of 16-bits	6.3.3.4.11.3.3, Table 64 — *Write* command, Figure 46 — Successful *Write* sequence, Table 75 — *BlockWrite* command
P:6	Read Transaction Time	Varied with R=>T and T=>R link rate and number of bits being read	6.3.3.4.11.3.2
P:7	Write Transaction Time	20ms (maximum) after end of *Write* command	6.3.3.4.11.3.3, Figure 46 — Successful *Write* sequence
P:8	Error detection	Interrogator-to-tag: CRC-5 or CRC-16 as defined in the relative clauses Tag-to-interrogator: CRC-5 or CRC-16 as defined in the relative clauses	6.3.3.3.1.5
P:9	Error correction	None	N/A
P:10	Memory size	Tag dependent	N/A
P:11	Command structure and extensibility	As specified	Table 46 — Commands

Table 31 — Collision management parameters

Ref.	Parameter Name	Description	Sub-clause
A:1	Type (Probabilistic or Deterministic)	Probabilistic	6.3.3.4.6
A:2	Linearity	Linear up to 2^{15} tags in the interrogator's RF field	6.3.3.4.8
A:3	Tag inventory capacity	$>2^{15}$ tags	6.3.3.4.8

6.3.3.3 Description of operating procedure

The operating procedure defines the physical and logical requirements for an interrogator-talks-first (ITF), random slotted anti-collision, RFID system operating at 13,56 MHz frequency.

This specification details two Methods of operation:

- ASK Method – (Mandatory). The interrogator shall communicate with one or more tags using ASK modulated PIE communication. A tag shall reply to interrogator ASK commands using the tag to interrogator link modulation specified in clause 6.3.3.3.1.3.11 Table 36 — ASK Method: Tag-to-interrogator link frequencies and Table 37 — ASK Method: Tag-to-interrogator data rates. The tag shall not change the modulation format and data rate.
- PJM Method – (Optional). The interrogator may optionally communicate with one or more tags using PJM-modulated, MFM-encoded communication. Tags that support PJM Method shall reply to interrogator PJM Method commands using the tag to interrogator link modulation specified in clause 6.3.3.3.1.3.11 and Table 38 — PJM Method: Subcarrier selection commands. Tags that do not support PJM Method shall not respond. The tag shall not change the modulation format and data rate.

Both modes use a common memory structure and protocol engine with a shared logical command set.

Note: Reference to specific ASK or PJM functions or commands in this specification are referenced "ASK Method:" or "PJM Method:" as appropriate.

6.3.3.3.1 Communications Signal Air Interface

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

6.3.3.3.1.1 Operational frequencies

Tags shall be capable of receiving power from and communicating with interrogators at the frequency of 13,56 MHz.

6.3.3.3.1.2 Interrogator-to-tag (R=>T) communications

ASK Method: An interrogator shall communicate with one or more tags by modulating an RF carrier using 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 6.3.3.4.8.

PJM Method: Interrogators shall communicate to one or more tags by modulating the RF carrier using PJM with MFM encoding at a fixed data rate of 212 kbit/s.

6.3.3.3.1.2.1 Interrogator frequency accuracy

Interrogator frequency accuracy shall comply with local radio regulations.

6.3.3.3.1.2.2 Modulation

ASK Method: All interrogators shall communicate using ASK, detailed in Annex J.

PJM Method: Interrogators may support PJM as detailed in Annex B.

6.3.3.3.1.2.3 Data encoding

ASK Method: The R=>T link shall use PIE encoding, shown in Figure 17 — ASK Method: PIE symbols. Tari is the reference time interval for interrogator-to-tag communication, and is the duration of a data-0. High values represent transmitted CW; low values represent attenuated CW. The tolerance on all parameters shall be +/– 1%, except as otherwise specified.

Pulse modulation depth, rise time, fall time, and PW shall be as specified in Table 32 — ASK Method: RF envelope parameters, 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, and Tari for the duration of an inventory round. The RF envelope shall be as specified in Figure 19 — ASK Method: Interrogator-to-tag RF envelope.

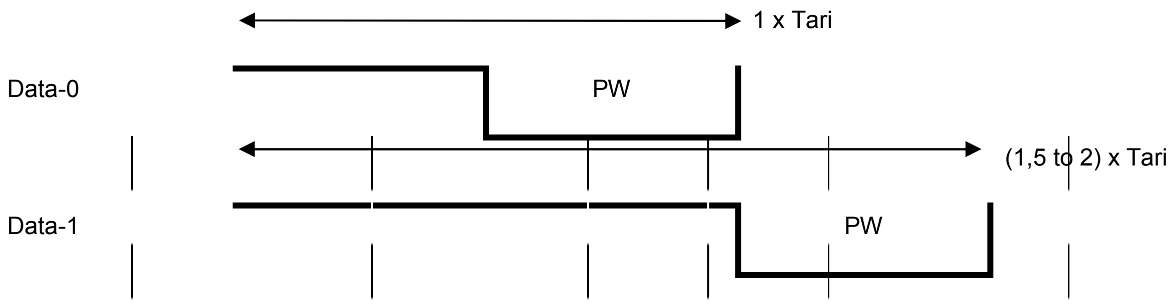


Figure 17 — ASK Method: PIE symbols

PJM Method: The R=>T link shall use PJM with MFM encoding at 212 kbit/s. The period of a bit interval used for encoding a command is 4,72 us (64 periods of the 13.56 MHz carrier). The rate of phase change is specified in Figure 20 — PJM Method: Command modulation scheme. Interrogators shall use a fixed phase change value for the duration of an inventory round.

The bit value is defined by a change in state. Bits are encoded using MFM encoding rules. These encoding rules are defined as follows:

- A data-1 is defined by a state change at the middle of a bit interval.
- A data-0 is defined by a state change at the beginning of a bit interval.
- Where a data-0 immediately follows a data-1 there is no state change.

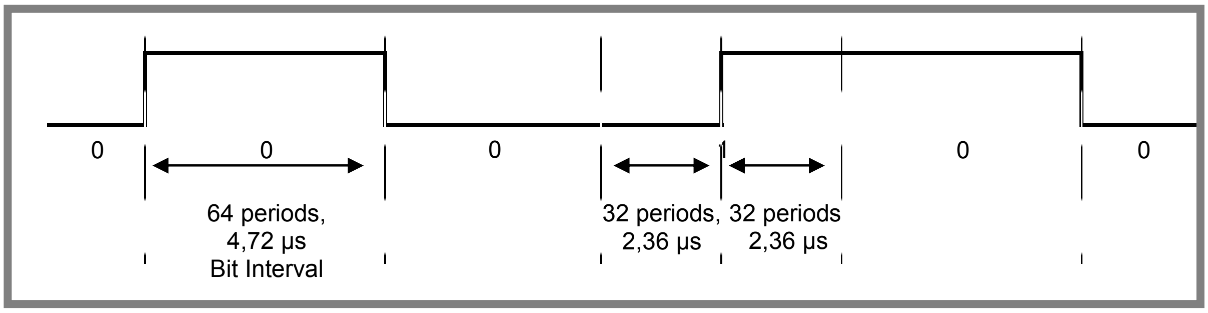


Figure 18 — PJM Method: Command MFM encoding and timing of binary 000100

An example of command MFM encoding of the command binary string is shown in Figure 18 — PJM Method: Command MFM encoding and timing of binary 000100. The edges shown represent small (+/- 3 deg for example) phase changes. Typically the edges shown in Figure 18 — PJM Method: Command MFM encoding and timing of binary 000100 shall be synchronised to the frequency carrier. If not synchronised then these edges will be generated at the interrogator with a tolerance of +/-one period of the frequency carrier.

6.3.3.3.1.2.4 ASK Method: Tari values

Interrogators shall communicate using Tari values between 8 μs and 25 μs, inclusive. An interrogator shall use fixed data-0 and data-1 symbol lengths for the duration of an inventory round, where “inventory round” is defined in 6.3.3.4.8. The choice of Tari value shall be in accordance with local radio regulations.

Note: Interrogator compliance shall be evaluated using at least one Tari value between 8 μs and 25 μs with at least one value of the parameter x = 1,5 Tari and x = 2,0 Tari.

6.3.3.3.1.2.5 R=>T RF envelope

ASK Method: The R=>T RF envelope shall comply with Figure 19 — ASK Method: Interrogator-to-tag RF envelope and Table 32 — ASK Method: RF envelope parameters. The magnetic field strength A is the maximum amplitude of the RF envelope, measured in A/m, Tari is defined in Figure 17 — ASK Method: PIE symbols. The pulse width is measured at the 50% point on the pulse.

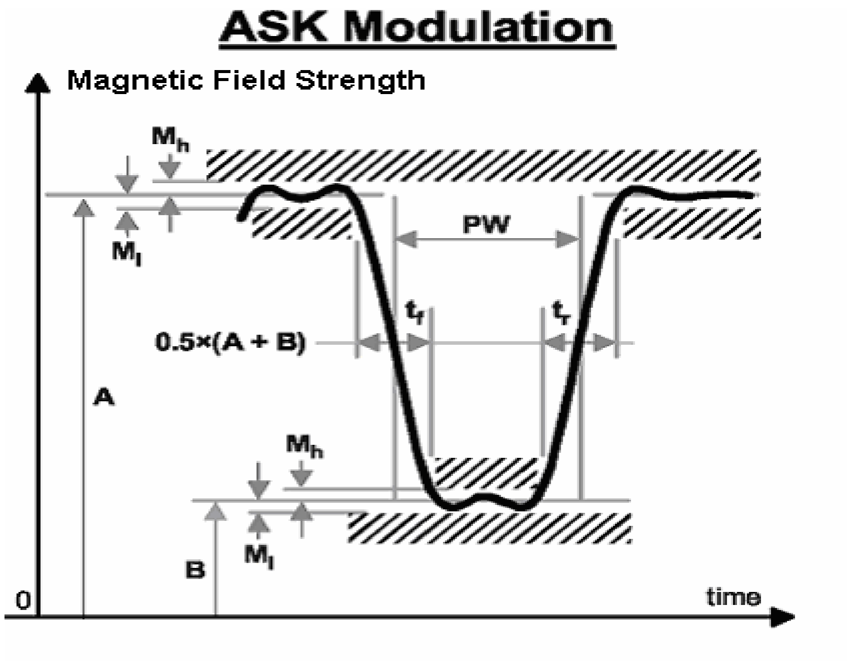


Figure 19 — ASK Method: Interrogator-to-tag RF envelope

Table 32 — ASK Method: RF envelope parameters

Tari	Parameter	Symbol	Minimum	Typical	Maximum	Units
8 μs to 25 μs	Modulation Index	(A–B)/(A+B)	10	15	30	%
	RF Envelope Overshoot Ripple	M_h	0		0,1 (A–B)	A/m
	RF Envelope Undershoot Ripple	M_l	0		0,1 (A–B)	A/m
	RF Envelope Rise Time	$t_{r,10–90\%}$	0		MIN(0,33Tari, 4,5)	μs
	RF Envelope Fall Time	$t_{f,10–90\%}$	0		MIN(0,33Tari, 4,5)	μs
	RF Pulsewidth	PW	MAX(0,265Tari, 4)		MIN(0,525Tari, 9,44)	μs

PJM Method: The R=>T link shall use PJM. PJM Method data is transmitted as very small phase changes in the interrogator RF carrier. The PJM Method phase shift waveform of the interrogator magnetic field is described in Figure 20 — PJM Method: Command modulation scheme and Table 33 — Command modulation parameters.

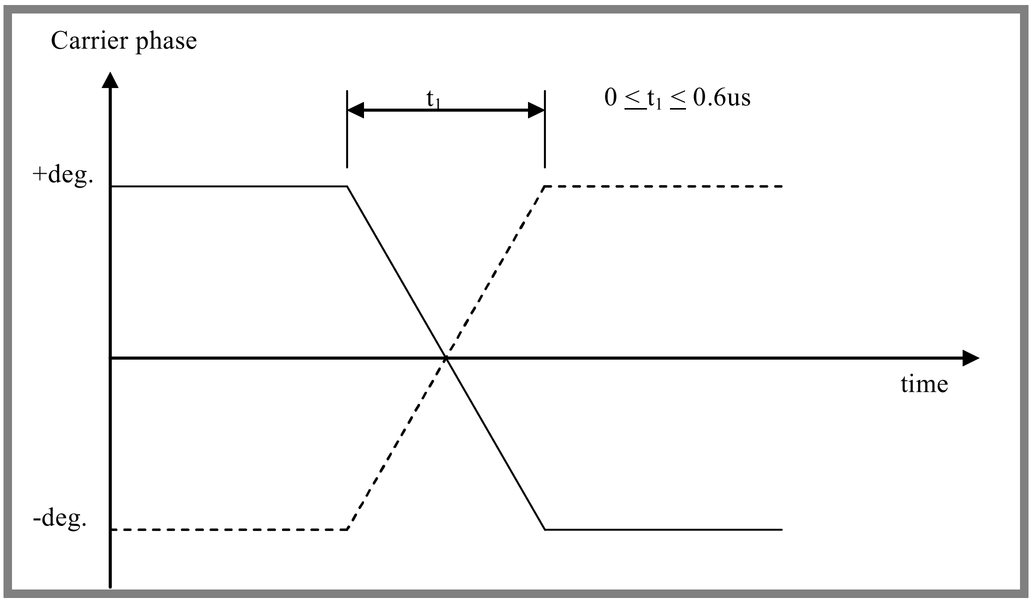


Figure 20 — PJM Method: Command modulation scheme

Table 33 — Command modulation parameters

Parameter	**Symbol**	**Nominal**	**Maximum**	**Units**
Phase Shift	+deg., -deg.	3,0	6,0	deg.
Transition Time	t_1	0,0	0,6	μs

Note: the phase shift cannot exceed the final phase value at any time and the transition time (t_1) is the time allowed to reach 95% of the full phase change.

6.3.3.3.1.2.6 Interrogator power-up waveform

The interrogator power-up RF envelope shall comply with Figure 21 — Interrogator power-up and power-down RF envelope and Table 34 — Interrogator power-up waveform parameters. 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 21 — Interrogator power-up and power-down RF envelope or rise above the 110% point during interval Ts and after interval Ts shall not fall below 99% or rise above 101%. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table 34 — Interrogator power-up waveform parameters (*i.e.* before Ts). Interrogators shall meet the frequency-accuracy requirement specified in 6.3.3.3.1.2.1 by the end of interval T_s in Figure 21 — Interrogator power-up and power-down RF envelope.

Note: When specifying compliance test there must be no moving tag in the field.

6.3.3.3.1.2.7 Interrogator power-down waveform

The interrogator power-down RF envelope shall comply with Figure 21 — Interrogator power-up and power-down RF envelope and Table 35 — Interrogator power-down waveform parameters. Once the carrier level has fallen below the 99% 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 at least 1ms before powering up again.

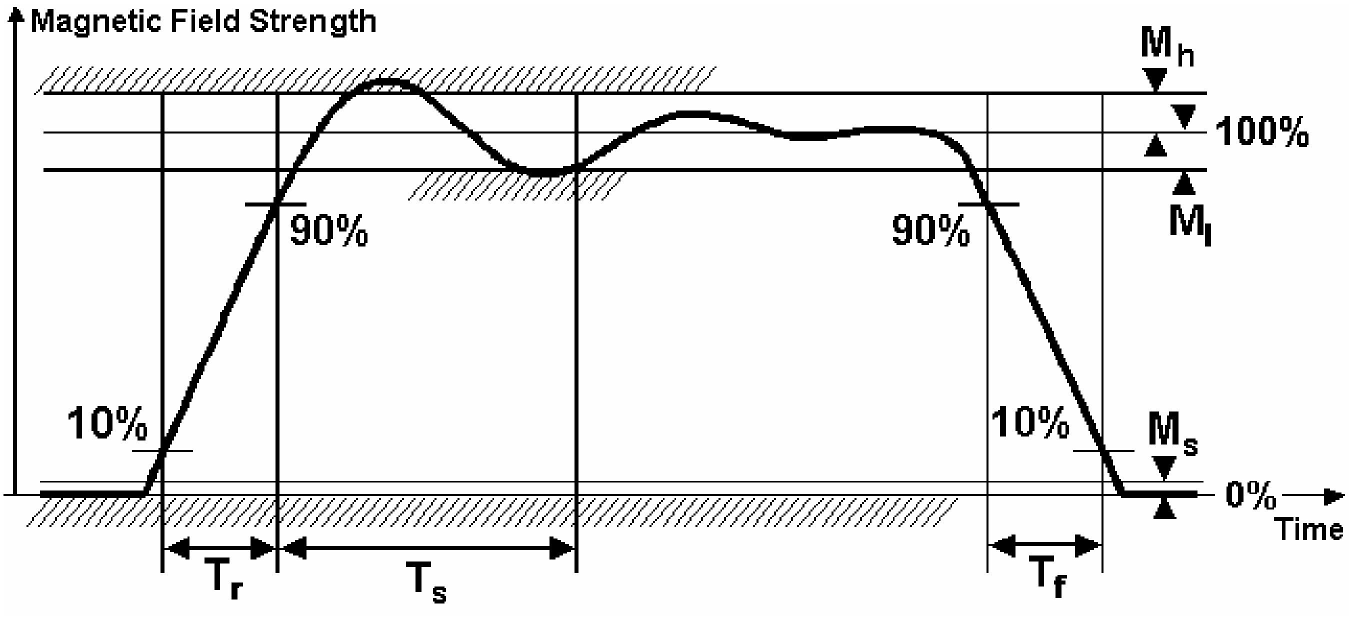


Figure 21 — Interrogator power-up and power-down RF envelope

Table 34 — Interrogator power-up waveform parameters

Parameter	Definition	Minimum	Typical	Maximum	Units
T_r	Rise time	1		500	µs
T_s	Settling time			1500	µs
M_s	Signal level when OFF			0.1	% full scale
M_l	Undershoot			10	% full scale
M_h	Overshoot			10	% full scale

Table 35 — Interrogator power-down waveform parameters

Parameter	Definition	Minimum	Typical	Maximum	Units
T_t	Fall time	1		500	µs
M_s	Signal level when OFF			0.1	% full scale

6.3.3.3.1.2.8 R=>T preamble and frame-sync

ASK Method: An interrogator shall begin all R=>T communication with either a preamble or a frame-sync, both of which are shown in Figure 22 — ASK Method: R=>T preamble and frame-sync. A preamble shall precede a *BeginRound* command (see 6.3.3.4.11.2.1) and denotes the start of an inventory round. All other communication 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 32 — ASK Method: RF envelope parameters The RF envelope shall be as specified in Figure 19 — ASK Method: Interrogator-to-tag RF envelope. A tag may compare the length of the data-0 with the length of RTcal to validate the preamble.

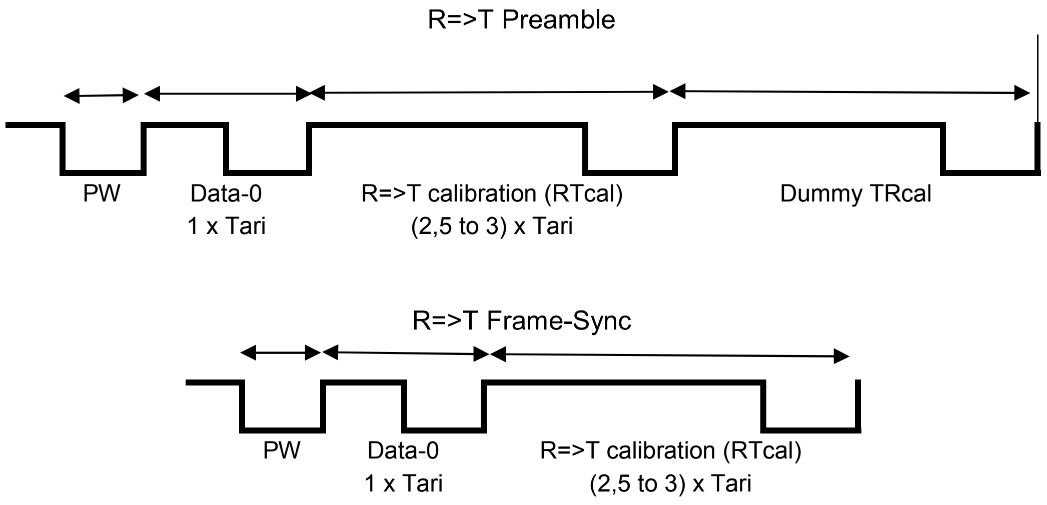


Figure 22 — ASK Method: R=>T preamble and frame-sync

A preamble shall comprise a modulation with the same length as used in the following data-0 symbol, a data-0 symbol, an R=>T calibration (RTcal) symbol, and a dummy 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 = 0length + 1length). A tag shall measure the length of RTcal and compute pivot = RTcal / 2. The tag shall interpret subsequent interrogator symbols shorter than pivot to be data-0s, and subsequent interrogator symbols longer than pivot to be data-1s. The 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: For the HF protocol, the TRcal is not used to define the T=>R return link rate. The TRcal value shall be between 1,1*RTcal and 3*RTcal (1.1*RTcal<=TRcal<=3*RTcal).
- 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.

PJM Method: The R=>T link shall use PJM with MFM encoding. An interrogator shall begin all R=>T communication with a MFM flag (or frame sync). The flag defines the start of a command and the bit interval timings. The flag comprises three parts:

- A synchronising string of 9 bits of valid MFM encoded data. As an example Figure 23 — PJM Method: MFM Encoding and timing for two possible command flags shows 9 data bits with the value "1".
- A MFM encoding violation not present in normal data. The violation consists of a sequence of 4 state changes separated by a 2 bit interval, a 1,5 bit interval and 2 bit interval. The edge of the fourth transition defines the beginning of a bit interval.
- A trailing 0 defining the end of a flag.

The synchronising string, encoding violation and a trailing zero for two possible command flags are illustrated in Figure 23 — PJM Method: MFM Encoding and timing for two possible command flags.

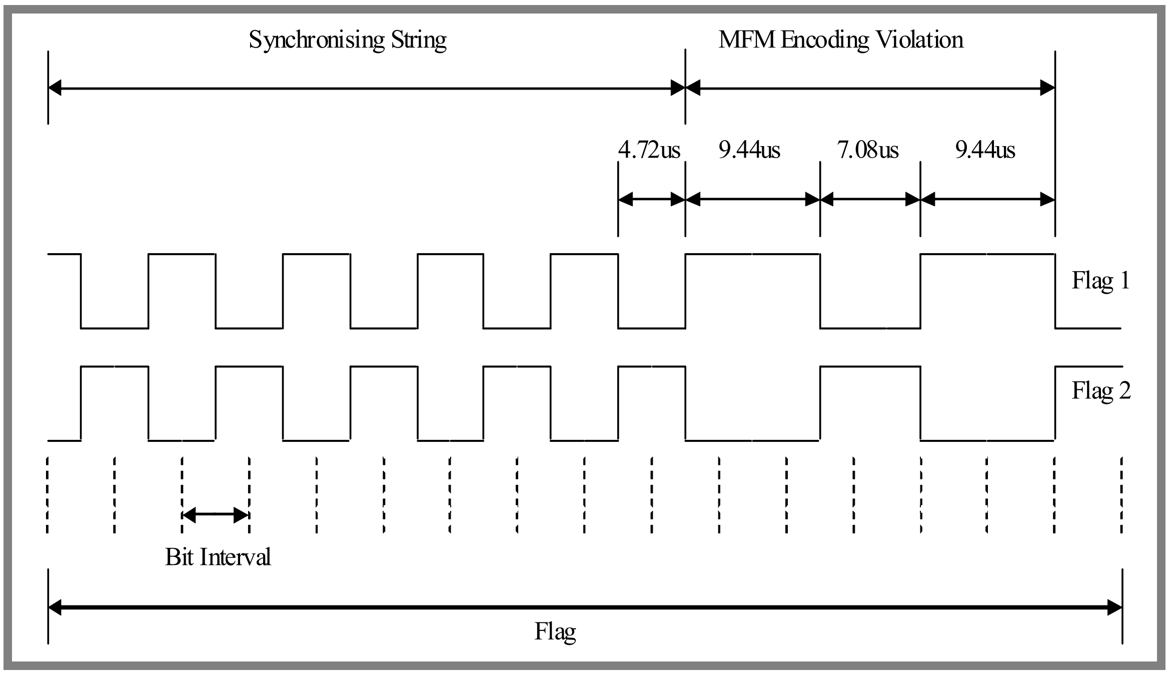


Figure 23 — PJM Method: MFM Encoding and timing for two possible command flags

6.3.3.3.1.3 Tag-to-interrogator (T=>R) communications

The tag communicates with an interrogator using loadmodulation. Loadmodulation is a method of impressing a data signal onto the carrier wave by changing the electrical "load" (impedance) of the tag. A tag is able to transmit data by varying its "load", thus effecting voltage changes at the interrogator antenna. The interrogator can then translate the voltage changes in a binary signal.

Load modulation can be achieved in several ways, e.g. by switching:

- load resistors
- shunt diodes
- the antenna tuning capacitance value or taps on the antenna coil (adjusting the tag tuning). It is possible to consider the switched tuning capacitance as part of the antenna and therefore the load seen by the tag antenna during the reply is fixed. Likewise the tuning capacitor and chip load presented to a tapped coil are constant.

ASK Method: A tag shall loadmodulate using a fixed modulation format, data encoding, and data rate for the duration of an inventory round, where “inventory round” is defined in 6.3.3.4.8. The interrogator sets the encoding and data rate by means of the *BeginRound* command that initiates the round. From this *BeginRound* command, the tag selects the T=>R modulation format.

PJM Method: The reply data rate is 106 kbit/s encoded using MFM and modulated onto the subcarrier as BPSK. Tags can select from one of eight subcarrier frequencies between 969 kHz and 3013 kHz. The subcarrier is derived from division of the interrogator operating frequency. During an inventory round a tag shall loadmodulate using a fixed subcarrier frequency as described in PJM Method sub clause of 6.3.3.1.3.10. The interrogator sets the reply channel by means of the *BeginRound* command that initiates the round.

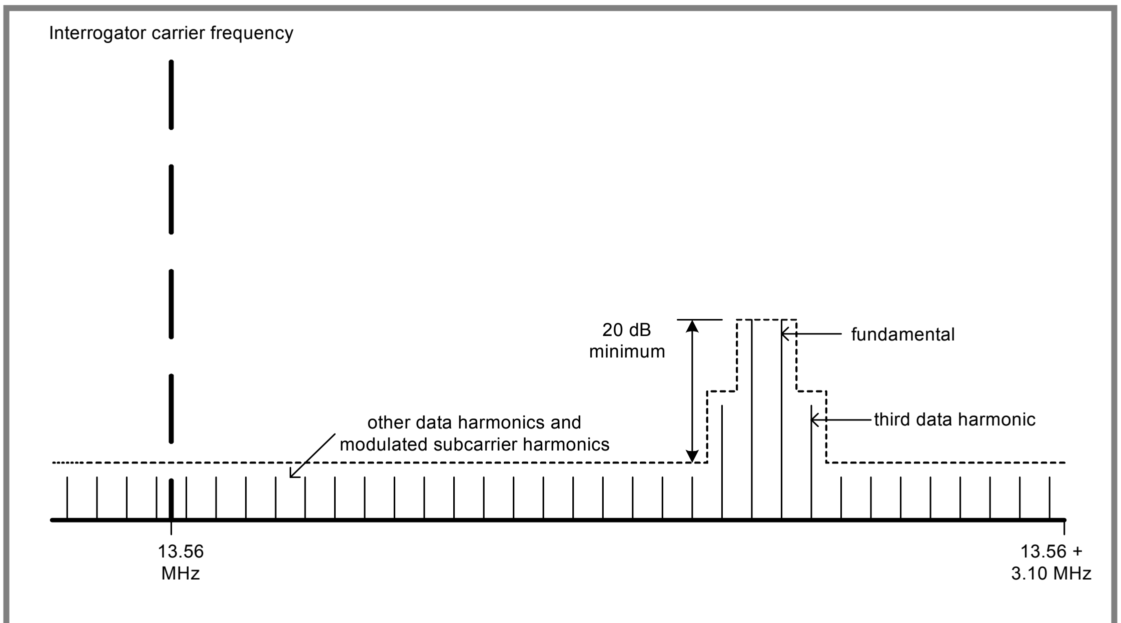
6.3.3.3.1.3.1 Modulation

For subsequent figures, the low values correspond to the tag antenna having a load impedance equal to the load impedance during the CW period immediately before a data transmission, whereas the high values correspond to the tag antenna having a different load impedance to the CW period.

ASK Method: The tag reply waveform may be baseband (for FM0 encoding) or modulated subcarrier (for Miller or Manchester encoding) as specified in clause 6.3.3.3.1.3.2 to 6.3.3.3.1.3.10.

PJM Method: The tag reply waveform is BPSK modulated subcarrier.

In order to ensure that tags replying on different channels are simultaneously received, all tag replies shall be band limited to reduce data and subcarrier harmonic levels as shown in Figure 24 — PJM Method: Tag reply mask. Intermediate load modulation impedances can be used for band limiting purposes


Interrogator carrier frequency
20 dB minimum
fundamental
third data harmonic
other data harmonics and modulated subcarrier harmonics
13.56 MHz
13.56 + 3.10 MHz

Note. This figure shows the upper side band of the Tag reply. The same mask shall be applied to the lower sideband.

Figure 24 — PJM Method: Tag reply mask

6.3.3.3.1.3.2 Data encoding

ASK Method: Tag data shall encode the loadmodulated reply as either FM0 baseband, or as Manchester-encoded subcarrier, or as Miller-encoded subcarrier. The interrogator sets the choice of encoding and data rate by parameters in the *BeginRound* command.

PJM Method: Tag data shall encode the loadmodulated reply as MFM of one of eight subcarriers of different frequencies at a data rate of 106 kbit/s. The selection of subcarrier frequency is either randomly selected by the tag, or directly selected by parameters in the *BeginRound* command.

6.3.3.3.1.3.3 ASK Method: FM0 baseband

Figure 25 — ASK Method: FM0 basis functions and generator state diagram shows basis functions 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.

Figure 26 — ASK Method: FM0 symbols and sequences shows generated baseband FM0 symbols and sequences. The duty cycle of a 00 or 11 sequence, measured at the modulator output, shall be 50%. FM0 encoding has memory; consequently, the choice of FM0 sequences in Figure 26 — ASK Method: FM0 symbols and sequences depends on prior transmissions. FM0 communication shall always end with a dedicated EOF at the end of a transmission, as shown in Figure 28 — ASK Method: Terminating FM0 transmissions EOF.

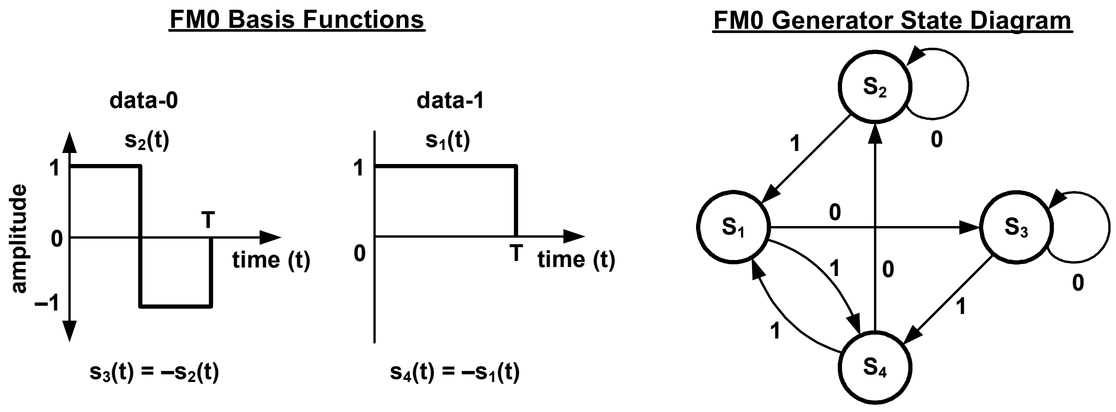


Figure 25 — ASK Method: FM0 basis functions and generator state diagram

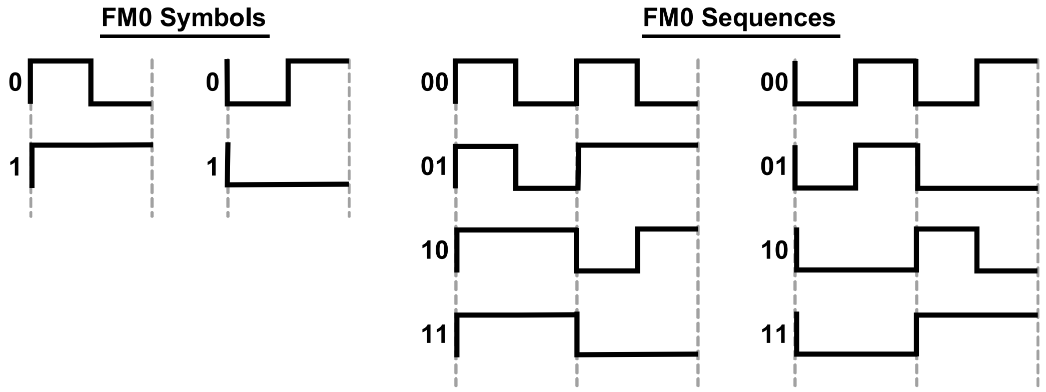


Figure 26 — ASK Method: FM0 symbols and sequences

6.3.3.3.1.3.4 ASK Method: FM0 Preamble SOF and EOF

T=>R FM0 communication shall begin with one of the two preambles shown in Figure 27 — ASK Method: FM0 Preamble SOF. The preamble comprises four data-0 followed by an FM0-violation of one bit duration, and ending with one data-0. The choice depends on the value of the TRext bit specified in the *BeginRound* command that initiated the inventory round. In Figure 27 — ASK Method: FM0 Preamble SOF the pilot tone for TRext=1 consists of additional continuous cycles at the subcarrier frequency for the duration of 12 bits. All tag answers are finalized by an EOF shown in Figure 28 — ASK Method: Terminating FM0 transmissions EOF.

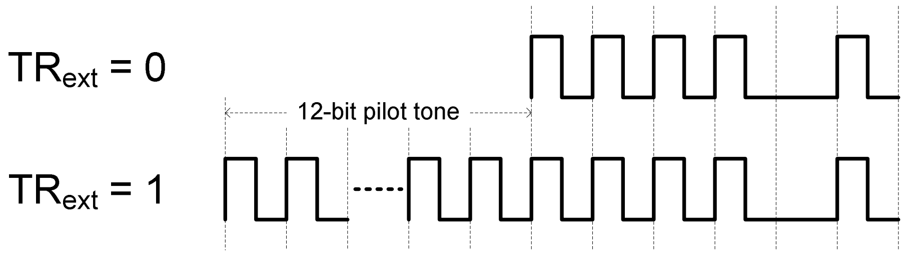


Figure 27 — ASK Method: FM0 Preamble SOF

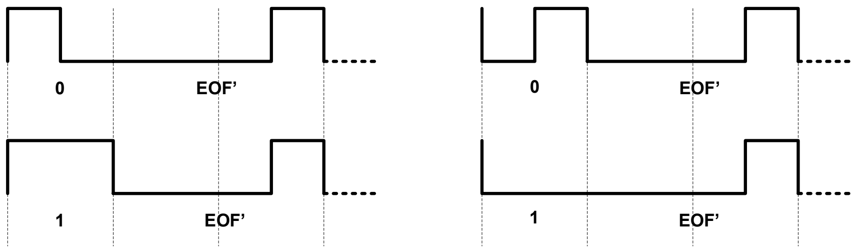


Figure 28 — ASK Method: Terminating FM0 transmissions EOF

6.3.3.3.1.3.5 ASK Method: Manchester-modulated subcarrier

Figure 29 — ASK Method: Manchester basis functions shows basis functions for generating Manchester encoding. Manchester encoding uses a high to low transition for data-0 and a low to high transition for data-1.

Manchester-modulated subcarrier provides subcarrier pulses followed by un-modulated time to encode data-0. Data-1 is encoded by un-modulated time followed by subcarrier pulses.

Figure 30 — ASK Method: Manchester subcarrier sequences shows Manchester-modulated subcarrier sequences; the Manchester sequence shall contain exactly four, or eight cycles per bit, depending on the M value specified in the *BeginRound* command that initiated the inventory round (see Table 36 — ASK Method: Tag-to-interrogator link frequencies). The two possible values of the subcarrier frequency (fc/32 and fc/16) are also defined in the *BeginRound* command by the DR bit.

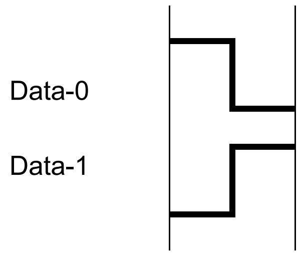


Figure 29 — ASK Method: Manchester basis functions

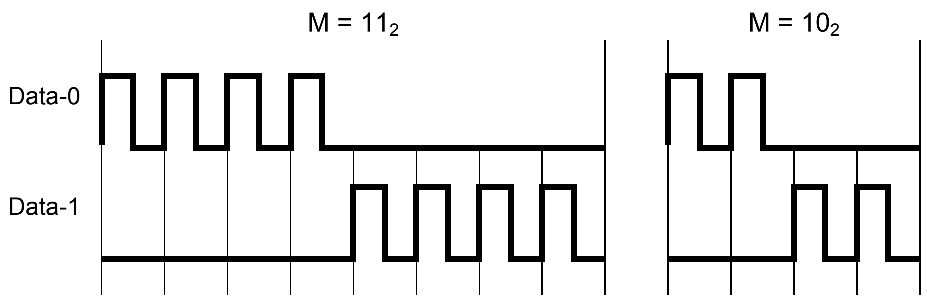


Figure 30 — ASK Method: Manchester subcarrier sequences

6.3.3.3.1.3.6 ASK Method: Manchester subcarrier preamble SOF and EOF

T=>R subcarrier communication shall begin with one of the two preambles shown in Figure 31 — ASK Method: Subcarrier T=>R preamble SOF. They contain exactly six (or twelve) subcarrier pulses followed by an un-modulated time of four (or eight) subcarrier cycles (that corresponds to a Manchester violation) and ended by a bit "0". The choice depends on the value of the TRext bit specified in the *BeginRound* command that initiated the inventory round. The duration of the pilot tone for TRext=1 is 12 bits. This pilot tone consists of continuous cycles at the subcarrier frequency. All tag answers are finalized by an EOF shown in Figure 32 — ASK Method: Terminating subcarrier transmissions EOF. The EOF comprises a bit "1" followed by a Manchester violation as a group of four (or eight) subcarrier pulses.

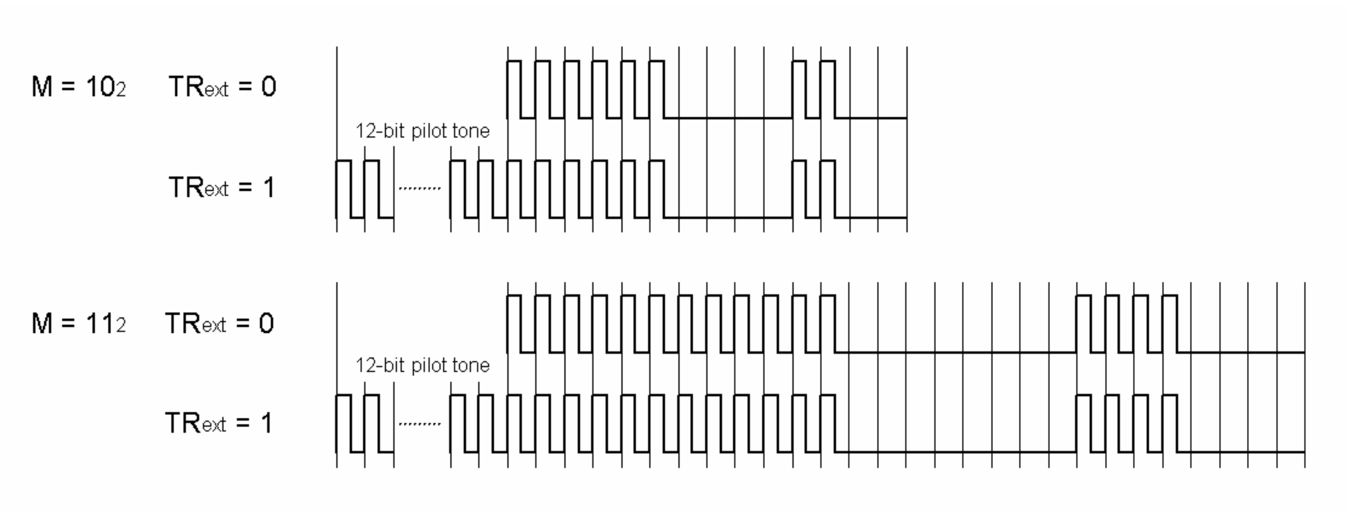


Figure 31 — ASK Method: Subcarrier T=>R preamble SOF

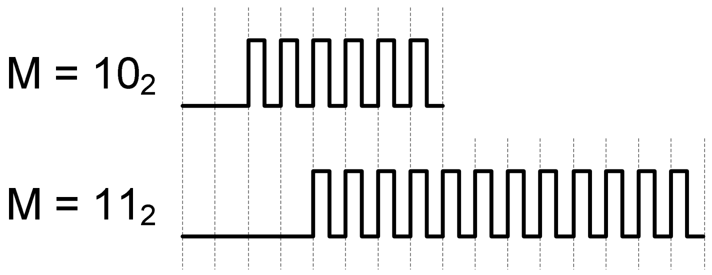


Figure 32 — ASK Method: Terminating subcarrier transmissions EOF

6.3.3.3.1.3.7 ASK Method: Miller-modulated subcarrier

Figure 33 — ASK Method: Miller basis functions and generator state diagram shows basis functions 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 33 — ASK Method: Miller basis functions and generator state diagram 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 8 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 34 — ASK Method: Miller subcarrier sequences shows a Miller-modulated subcarrier sequence; the Miller sequence shall contain exactly eight subcarrier cycles per bit (see Table 36 — ASK Method: Tag-to-interrogator link frequencies). Miller encoding has memory; consequently, the choice of Miller subcarrier sequences as shown in Figure 34 — ASK Method: Miller subcarrier sequences depends on prior transmissions.

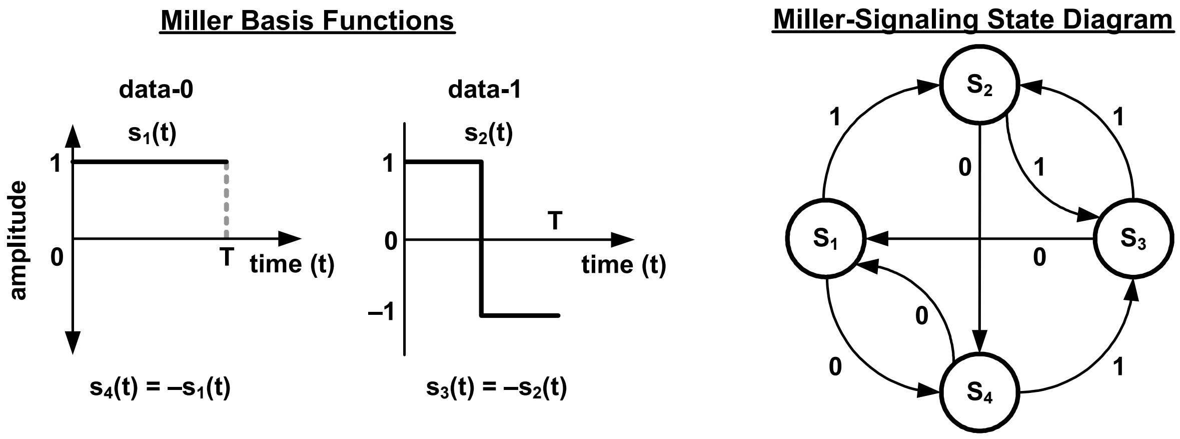


Figure 33 — ASK Method: Miller basis functions and generator state diagram

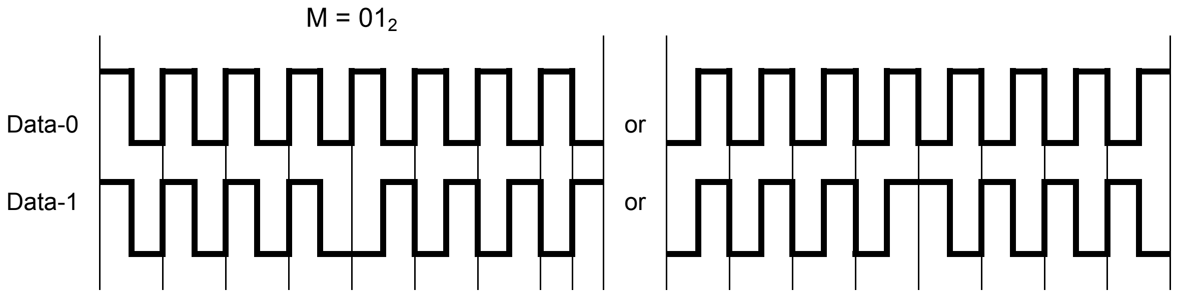


Figure 34 — ASK Method: Miller subcarrier sequences

6.3.3.3.1.3.8 ASK Method: Miller subcarrier preamble SOF and EOF

T=>R subcarrier communication shall begin with one of the two preambles shown in Figure 35 — ASK Method: Miller Subcarrier T=>R preamble SOF. The choice depends on the value of the TRext bit specified in the *BeginRound* command that initiated the inventory round. Figure 35 — ASK Method: Miller Subcarrier T=>R preamble SOF shows the time of a pilot tone for each TRext. This pilot tone consists of continuous cycles at the subcarrier frequency. Miller encoded communication shall always end with dedicated EOF at the end of a transmission, as shown in Figure 36 — ASK Method: Terminating subcarrier transmissions EOF.

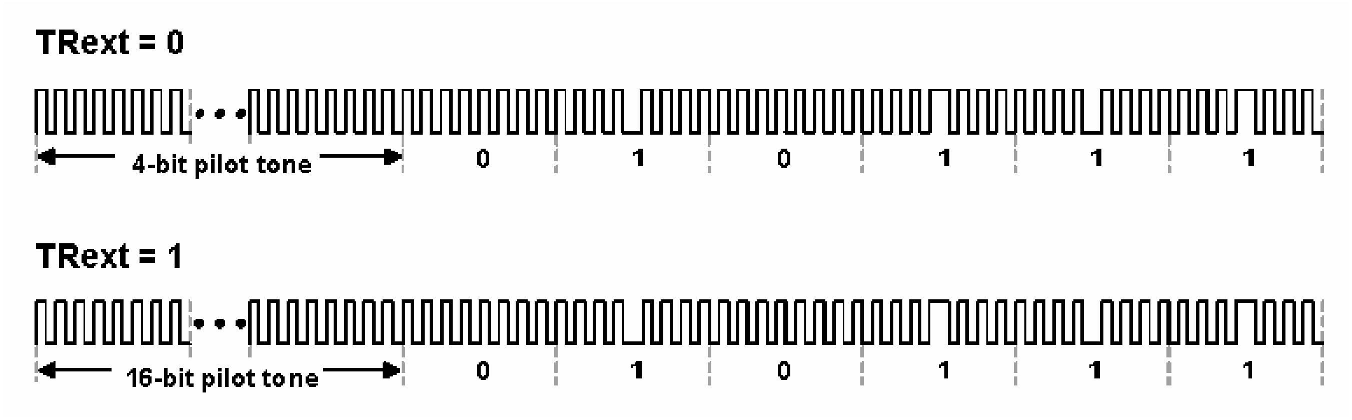


Figure 35 — ASK Method: Miller Subcarrier T=>R preamble SOF

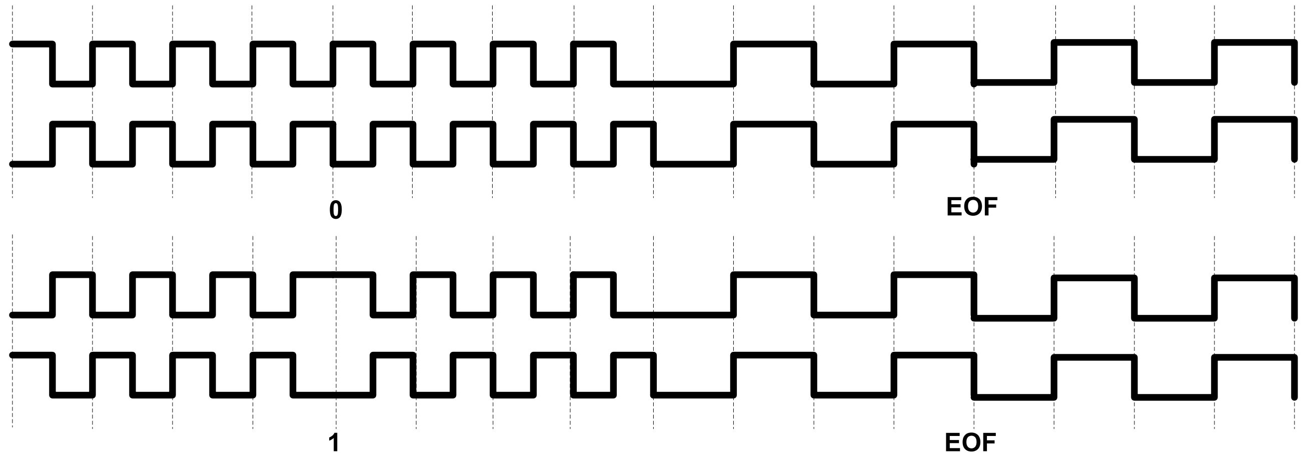


Figure 36 — ASK Method: Terminating subcarrier transmissions EOF

6.3.3.3.1.3.9 PJM Method: MFM modulated subcarrier

Figure 37 — PJM Method: MFM basis functions and generator state diagram shows basis functions for generating MFM encoding. Baseband MFM inverts its phase between two data-0s in sequence. Baseband MFM also places a phase inversion in the middle of a data-1 symbol. The state diagram in Figure 37 — PJM Method: MFM basis functions and generator state diagram maps a logical data sequence to baseband MFM basis functions. The state labels, S_1–S_4, indicate four possible MFM-encoded symbols, represented by the two phases of each of the MFM basis functions. The state labels also represent the baseband MFM waveform that is generated upon entering the state. The transmitted waveform is the baseband waveform multiplied by a square wave at fc divided by n, where n is the division ratio for the 8 subcarriers defined in Table 38 — PJM Method: Subcarrier selection commands. 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 38 — PJM Method: Reply MFM encoding and timing of binary 000100 shows an example for the BPSK phase changes without subcarrier and the corresponding MFM-modulated subcarrier sequence. The phase changes align with MFM state changes shown in Figure 37 — PJM Method: MFM basis functions and generator state diagram. For clarity the subcarrier example shows eight cycles per bit period, however the real subcarrier will have between nine and 28 cycles depending upon the reply channel selected. Also for clarity the phase change shown occurs at the beginning and the middle of a subcarrier cycle; however the phase

change can occur at any point during a subcarrier cycle because the reply data rate and the subcarriers are unsynchronized.

The subcarrier is derived by division of the powering field's frequency and is determined from the DR, M and TRext values specified in the *BeginRound* command that initiated the inventory round (see Table 38 — PJM Method: Subcarrier selection commands). The period T of a bit interval used for encoding command data is 9,44 µs. MFM encoding has memory; consequently, the choice of MFM sequences in Figure 38 — PJM Method: Reply MFM encoding and timing of binary 000100 depends on prior transmissions.

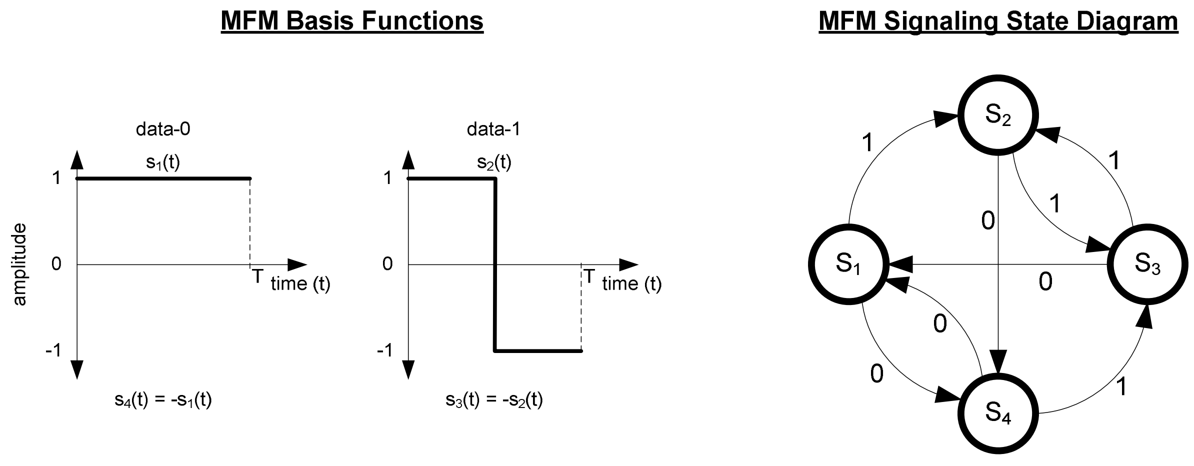


Figure 37 — PJM Method: MFM basis functions and generator state diagram

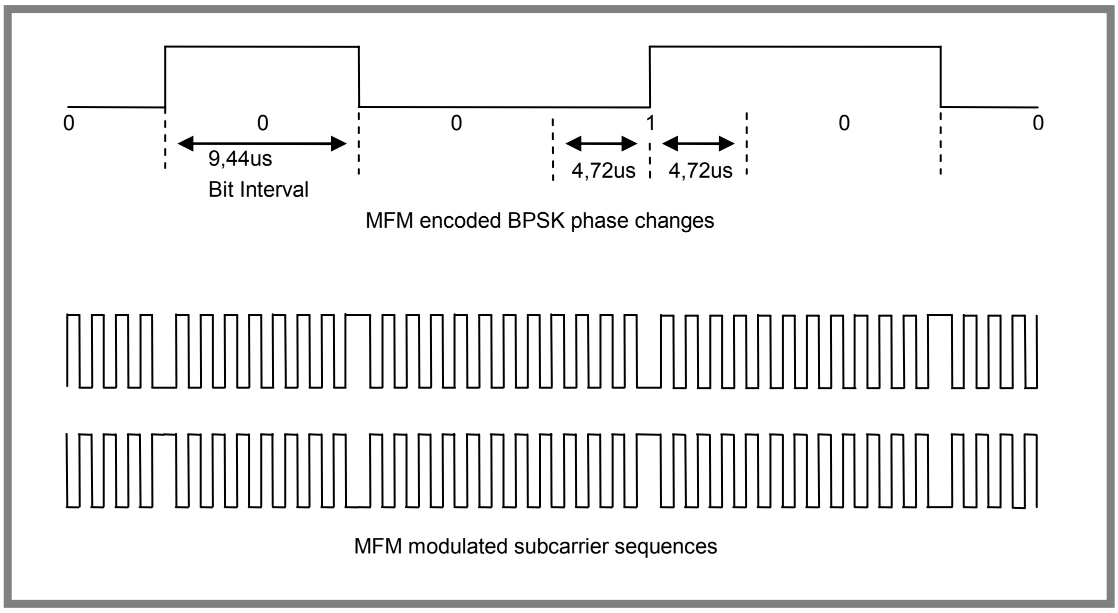

0
0
0
1
0
0
9,44us
Bit Interval
4,72us
4,72us
MFM encoded BPSK phase changes
MFM modulated subcarrier sequences

Figure 38 — PJM Method: Reply MFM encoding and timing of binary 000100

6.3.3.3.1.3.10 PJM Method: MFM subcarrier flag (preamble SOF)

PJM Method: The T=>R subcarrier communication shall begin with either of the two flags shown in Figure 39 —PJM Method: Subcarrier T=>R flags. This figure shows the flag timing for the BPSK subcarrier phase changes.

The flag defines the start of a reply and the bit interval timings. The flag is comprised of three parts:

- A synchronising string of 9 bits of valid MFM encoded data. As an example Figure 39 —PJM Method: Subcarrier T=>R flags shows 9 data bits with the value "1".
- A MFM encoding violation which is not present in normal data. The violation consists of a sequence of 4 state changes separated by a 2 bit interval, a 1,5 bit interval and 2 bit interval. The edge of the fourth transition defines the beginning of a bit interval, and
- A trailing 0.

A synchronising string, encoding violation and a trailing zero for two possible reply flags are illustrated in Figure 39 —PJM Method: Subcarrier T=>R flags.

Note: an EOF is NOT required.

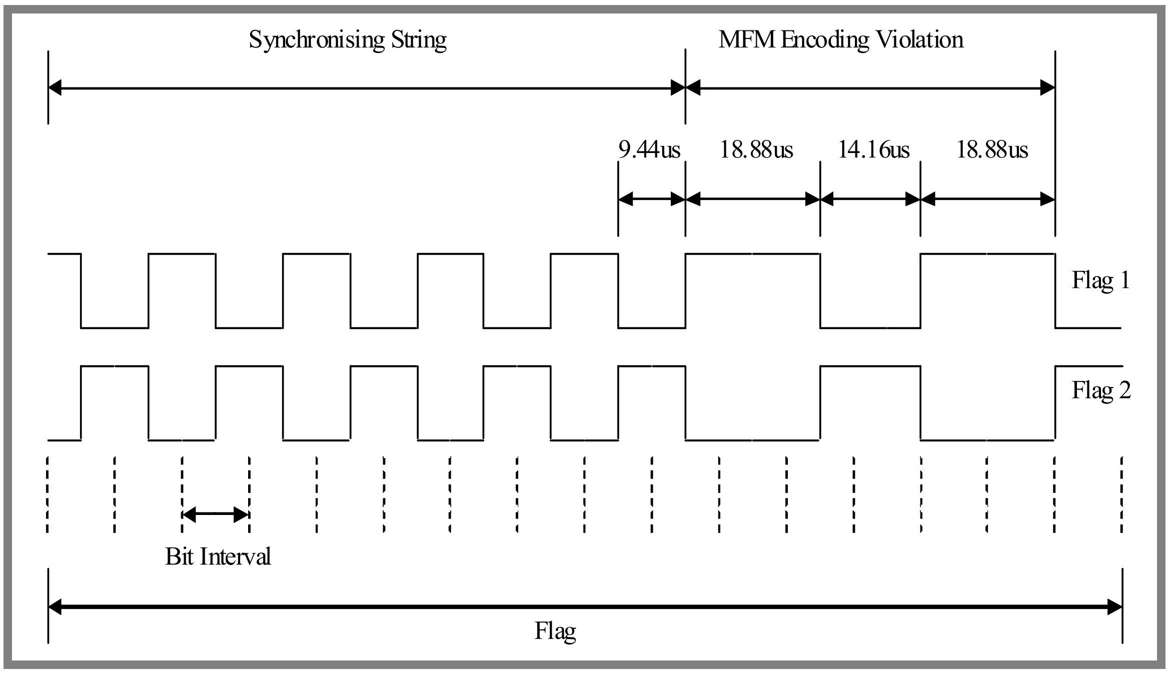


Figure 39 —PJM Method: Subcarrier T=>R flags

6.3.3.3.1.3.11 ASK Method and PJM Method: Selection of modulation type and data rate

- ASK Method: Tags shall support the R=>T Tari values specified in 6.3.3.3.1.2.4, the T=>R link frequencies specified in Table 36 — ASK Method: Tag-to-interrogator link frequencies, and the T=>R data rates specified in Table 37 — ASK Method: Tag-to-interrogator data rates. The *BeginRound* command that initiates an inventory round specifies DR in Table 36 — ASK Method: Tag-to-interrogator link frequencies and M in Table 37 — ASK Method: Tag-to-interrogator data rates.

Table 36 — ASK Method: Tag-to-interrogator link frequencies

M value	Bit modulation	Subcarrier with DR=0 LF = 423kHz (f_c/32)	Subcarrier with DR=1 LF = 847kHz (f_c/16)
11_2	4 subcarrier pulse Manchester	53 kbit/s (f_c/256)	106 kbit/s (f_c/128)
10_2	2 subcarrier pulse Manchester	106 kbit/s (f_c/128)	212 kbit/s (f_c/64)
01_2	8 subcarrier pulse Miller	53 kbit/s (f_c/256)	106 kbit/s (f_c/128)
00_2	FM0	424 kbit/s (f_c/32)	848 kbit/s (f_c/16)

Table 37 — ASK Method: Tag-to-interrogator data rates

M: Number of subcarrier cycles per symbol	Modulation type	Data rate (kbit/s)
M = 00_2: 1 cycles per symbol	FM0 baseband	LF
M = 01_2: 8 cycles per symbol	Miller subcarrier	LF/8
M = 10_2: 4 cycles per symbol	Manchester subcarrier	LF/4
M = 11_2: 8 cycles per symbol	Manchester subcarrier	LF/8

- PJM Method: The T=>R modulation shall be MFM at a data rate of 106 kbit/s on a subcarrier selected by the *BeginRound* command that initiates an inventory round The *BeginRound* command provides the DR, M, and TRext bits which are (for PJM Method commands) decoded by capable tags as shown in Table 38 — PJM Method: Subcarrier selection commands.

Table 38 — PJM Method: Subcarrier selection commands

DR	M	TRext	reply channel selected	division ratio	subcarrier frequency kHz
0	00	0	A	14	969
0	00	1	B	11	1233
0	01	0	C	9	1507
0	01	1	D	7,5	1808
0	10	0	E	6,5	2086
0	10	1	F	5,5	2465
0	11	0	G	5	2712
0	11	1	H	4,5	3013
1	00	0	Subcarrier frequency hopping		
all other states			reserved for future use		

When frequency hopping is selected by the *BeginRound* command tags shall select a reply channel. A tag shall use this reply channel for all T=>R communications. The channel selected by a tag shall be determined from the three least significant bits of the StoredCRC. The three least significant bits of the StoredCRC shall be decoded as for the M and TRext bits in Table 38 — PJM Method: Subcarrier selection commands above. For example, if the three least significant bits of the StoredCRC are 110 then channel G would be selected

6.3.3.3.1.4 Both Methods: Transmission order

The transmission order for all R=>T and T=>R 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.

6.3.3.3.1.5 Both Methods: 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 loadmodulated T=>R replies. This protocol uses two CRC types: (i) a CRC-16, and (ii) a CRC-5. Annex I describes both CRC types.

To generate a CRC-16 a tag or Interrogator shall first generate the CRC-16 precursor shown in Table 39 — CRC 16 precursor and then take the ones-complement of the generated precursor to form the CRC-16.

Table 39 — CRC 16 precursor

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

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

At power-up, a tag calculates and saves into memory a 16-bit StoredCRC — see 6.3.3.4.1.2.1.

Tags shall append a CRC-16 to those replies that use a CRC-16 — see 6.3.3.4.11 for command-specific reply formats.

To generate a CRC-5 an Interrogator shall use the definition in Table 40 — CRC-5 definition. See also Annex I.

Table 40 — CRC-5 definition. See also Annex I

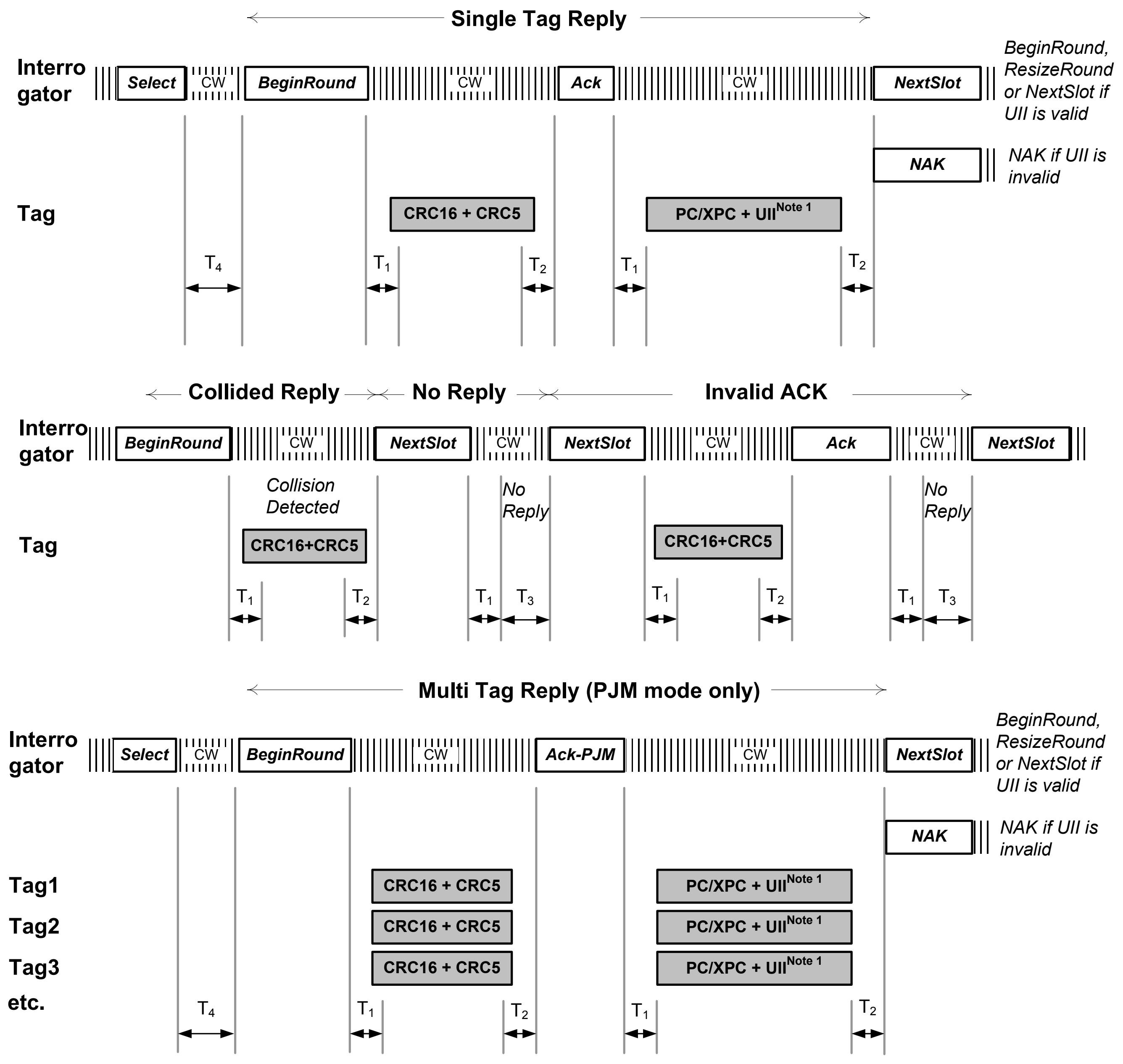
CRC Type	Length	Polynomial	Preset	Residue
—	5 bits	$x^5 + x^3 + 1$	01001_2	00000_2

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

Interrogators shall append the appropriate CRC to R=>T transmissions as specified in Table 46 — Commands.

6.3.3.3.1.6 Link timing – Both Methods

Figure 40 — Link timing – Both Modes illustrates R=>T and T=>R link timing. Figure 40 — Link timing – Both Modes (not drawn to scale) defines interrogator interactions with a tag population. Table 41 — Link timing parameters shows the timing requirements for Figure 40 — Link timing – Both Modes, while 6.3.3.4.11 describes the commands. Tags and interrogators shall meet all timing requirements shown in Table 41 — Link timing parameters. RTcal is defined in 6.3.3.3.1.2.8. As described in 6.3.3.3.1.2.8, 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.



Note 1: In certain cases a PacketCRC shall be added if required. See table 42 for the definition of the cases.

Figure 40 — Link timing – Both Modes

Table 41 — Link timing parameters

Parameter	Minimum	Typical	Maximum	Description
T_1	73,1 µs (1024-32)/f_c	75,5 µs (1024/f_c)	77,9 µs (1024+32)/f_c	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 antenna terminals.
T_2	151 µs (2048/f_c)		1208 µs (16384/f_c)	Interrogator response time required if a tag is to demodulate the Interrogator signal, measured from the end of the tag response to the first falling edge of the Interrogator transmission
T_3	Tsof_tag			Time an interrogator waits, after T_1, before it issues another command
T_4	$T_{1Typ} + T_{3Min}$			Minimum time between interrogator commands

Note 1: A tag may exceed the maximum value of T_1 when responding to commands that write to memory.

Note 2: The maximum value for T_2 shall apply only to tags in the **reply** or **acknowledged** states (see 6.3.3.4.4.3 and 6.3.3.4.4.4). 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 6.3.3.4.4.2);
- 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. “Invalid command” is defined in 6.3.3.4.11.

Note 3: A tag shall be allowed a tolerance of $16384/fc <= T_{2max} <= 16448/fc$ in determining whether T_2 has expired.

6.3.3.4 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.

6.3.3.4.1 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 41 — Logical memory map.

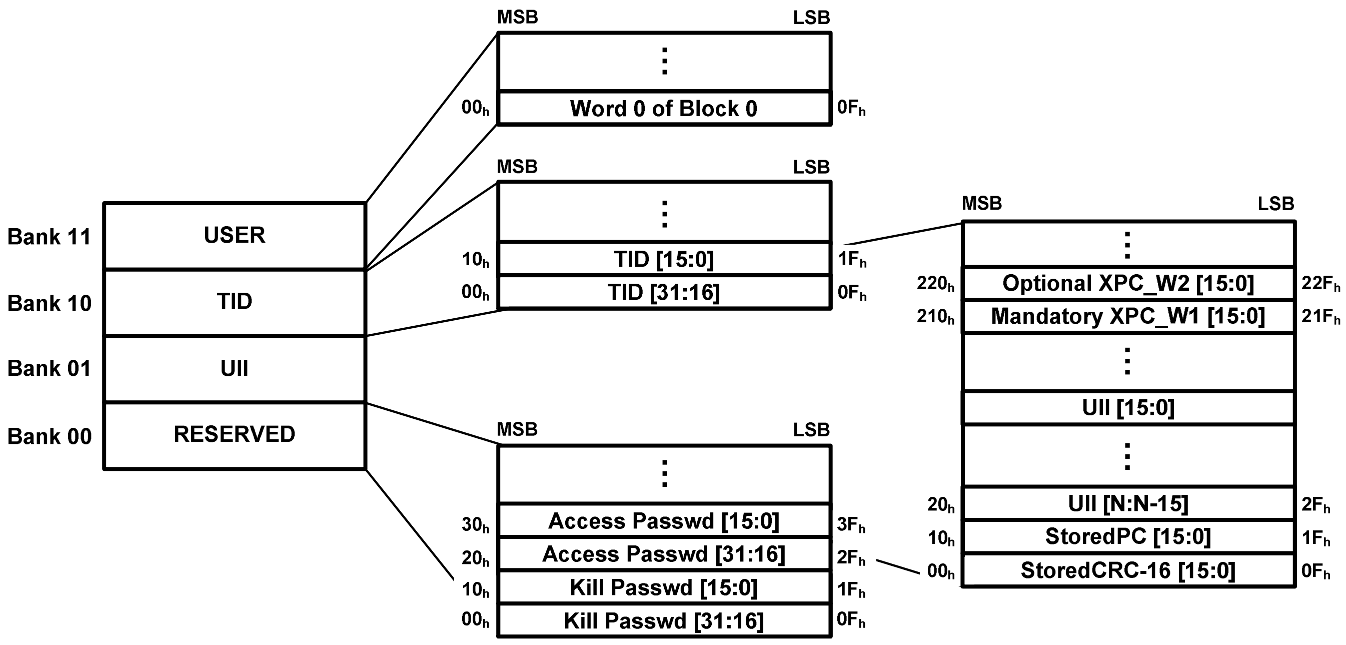


Figure 41 — Logical memory map

The memory banks are:

Reserved memory shall contain the kill and and/or access passwords, if passwords are implemented on the tag. The kill password shall be stored at memory addresses 00h to 1Fh; the access password shall be stored at memory addresses 20h to 3Fh. See 6.3.3.4.1.1

Unique Item Identifier (UII) memory shall contain a CRC-16 at memory addresses 00_h to $0F_h$, a Protocol-Control (PC) word 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 shall be attached beginning at address 20_h, and if the tag implements Extended Protocol Control (XPC) then either one (mandatory) or two (optional) Extended Protocol Control (XPC) word(s) beginning at address 210_h. See 6.3.3.4.1.2.

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 6.3.3.4.1.3.

User memory is optional. See 6.3.3.4.1.4.

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, an address length parameter and an address parameter to select a particular memory location within that bank. The address parameter has a length of 8, 16, 24 or 32 bits as defined in the address length parameter. Interrogators shall use the shortest address length possible to encode the memory location. When Tags loadmodulate memory contents, this modulation shall fall on word boundaries (except in the case of a truncated reply – see 6.3.3.4.11.1.1).

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 6.3.3.4.9, involve the transfer of 16-bit words from Interrogator to Tag. A *Write* command writes 16 bits (*i.e.* one word) at a time, optionally 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 killed tag's permanently non-responsive status regardless of the memory 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 6.3.3.4.9 for a description of memory locking and unlocking, and 6.3.3.4.10 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.

6.3.3.4.1.1 Reserved Memory

Reserved memory contains the kill (see 6.3.3.4.1.1.1) and/or access (see 6.3.3.4.1.1.2) 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 6.3.3.4.11.3.5), and the corresponding physical memory locations in Reserved memory need not exist.

6.3.3.4.1.1.1 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.

6.3.3.4.1.1.2 Access password

The access password is a 32-bit value stored in Reserved memory 20_h to $3F_h$, MSB first. The default (unpro-grammed) 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

6.3.3.4.1.2 UII Memory

UII memory contains a StoredCRC at memory addresses 00_h to $0F_h$, a StoredPC at 10_h to $1F_h$, an UII beginning at 20_h, and a first XPC word (XPC_W1) at 210_h to $21F_h$ and an optional second XPC word (XPC_W2) at 220_h to $22F_h$. 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 is described in 6.3.3.4.1.2.1.

The StoredPC, as described in 6.3.3.4.1.2.2, is subdivided into an 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 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 6.3.3.4.1.2.3; the UII for ISO Applications is described in 6.3.3.4.1.2.4. 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 loadmodulate its PC word, XPC (if XI is asserted), UII, and a PacketCRC if required. Under certain circumstances the tag may truncate its reply (see 6.3.2.11.1.1). An Interrogator may issue a *Read* command to read all or part of the UII.

6.3.3.4.1.2.1 CRC-16 (StoredCRC and PacketCRC)

All Tags shall implement a StoredCRC. Tags shall also implement a PacketCRC.

At power-up a tag shall calculate a CRC-16 over (a) the StoredPC and (b) the UII specified by the UII length field in the StoredPC (see 6.3.4.1.2.1) and shall map the calculated CRC-16 into UII memory 00_h to $0F_h$, MSB first. This CRC is denoted the StoredCRC. Because the StoredPC and UII comprise an integer number of UII-memory words, a tag calculates this StoredCRC on word boundaries. Although Tags include the XI value in the StoredPC in their StoredCRC calculation, regardless of the XI value a tag shall omit XPC_W1 and XPC_W2 from the calculation. Tags shall finish this CRC-16 computation and memory mapping by the end of interval T_s in Figure 21 — Interrogator power-up and power-down RF envelope. Interrogators may issue a *Select* command that includes all or part of the StoredCRC in Mask. Interrogators may issue a *Read* command to instruct a tag to loadmodulate its StoredCRC. Tags shall not recalculate this StoredCRC for a truncated reply (see 6.3.3.4.11.1.1).

In response to an *ACK* command a tag loadmodulates a protocol-control (PC) word (either StoredPC or PacketPC – see 6.3.3.4.1.2.2), a first XPC word (XPC_W1) and an optional second XPC word (XPC_W2) depending on XI (see 6.3.3.4.1.2.5), UII (see 6.3.3.4.1.2.3 and 6.3.3.4.1.2.4), and a CRC-16 only in case of a PackedCRC is required according to table 42 below. The CRC-16 in this case shall be a PacketCRC that the tag shall calculate dynamically over the loadmodulated PC word, XPC word or words (if supported), and UII. Whether a tag omits the CRC16 or loadmodulates its PacketCRC, shall be as defined in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command, depending on the tag's XI value and whether truncation (see 6.3.3.4.11.1.1) is asserted or deasserted.

Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an *ACK* command

XI	XEB	Truncation	Tag Loadmodulation			
			PC	XPC	UII	CRC-16
0	0	Deasserted	StoredPC	None	Full	no CRC16 in reply
0	0	Asserted	00000_2	None	Truncated	no CRC16 in reply
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	no CRC16 in reply
1	1	Deasserted	PacketPC	Both XPC_W1 and XPC_W2	Full	PacketCRC
1	1	Asserted	00000_2	None	Truncated	no CRC16 in reply

Note 1: XI is the bitwise logical OR of the 16 bits of XPC_W1, and XEB is the MSB (bit Fh) of XPC_W1, so if XEB=1 then XI=1

Note 2: The StoredCRC may be different from the Packet CRC if the tag memory has been programmed and the tag has not undergone a power-on-reset.

A tag shall loadmodulate 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 6.3.3.3.1.5 an Interrogator shall verify, using a tag's loadmodulated CRC-16, the integrity of a received PC word, first XPC word (XPC_W1), optional second XPC word (XPC_W2), and UII.

6.3.3.4.1.2.2 Protocol-control (PC) word

All Tags shall implement a StoredPC whose fields, comprising UII length, a UMI, an XI, and an NSI, shall be as defined below. Tags 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 loadmodulates in response to an *ACK* shall be as defined in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command

The StoredPC shall be located in UII memory at addresses 10h to 1Fh with bit values defined as follows:

- Bits 10_h – 14_h: The length of the UII that a tag loadmodulates, in words:
  - 00000_2: Zero words.
  - 00001_2: One word (addresses 20_h to $2F_h$ in UII memory).
  - 00010_2: Two words (addresses 20_h to $3F_h$ in UII memory).
  -
  -
  -
  - 11101_2: 29 words (addresses 20_h to $1EF_h$ in UII memory).

  The maximum value of the UII length field in the StoredPC shall be 11101_2 (allows a 464-bit UII). A tag shall ignore a *Write* or *BlockWrite* command to the StoredPC if the UII length field exceeds 11101_2, and shall instead loadmodulate an error code (see Annex E).

- 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. In case that only the mandatory recommisioning option with asserted Recom bit 3SB is supported and no block permalocking is supported also Method 2 may be used instead of Method 1.
  - Method 1: The tag computes the UMI. At power-up, and prior to computing the StoredCRC, 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 the StoredCRC calculated at power-up (see 6.3.3.4.1.2.1). 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 6.3.3.4.10) 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 — if 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 the 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.

- Bit 16_h: An XPC_W1 indicator (XI). If bit 16_h is deasserted, then the XPC_W1 is zero-valued, in which case the tag shall loadmodulate its StoredPC but not an XPC_W1 during inventory (see 6.3.3.4.1.2). If bit 16_h is asserted then one or more bits of XPC_W1 have nonzero values, indicating that the tag has been previously re-commissioned (see 6.3.3.4.1.2.5).
  In this latter case the tag shall loadmodulate its XPC_W1 immediately after the PacketPC, and before the UII, during inventory.

  At power-up, and prior to computing the StoredCRC the tag shall compute the bitwise logical OR of the XPC_W1, and map the computed value into bit 16_h (i.e. into the XI). The tag shall include the computed XI value in its StoredCRC calculation. If an Interrogator recommissions the tag (see 6.3.3.4.10) then the tag shall recompute and remap its XI into bit 16_h. After remapping 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 PC word, the tag shall ignore the data value that the Interrogator provides for bit 16_h.

- 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 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 bits 18_h – $1F_h$ shall contain the entire AFI defined in ISO/IEC 15961 parts 2 and 3. The default value for bits 18_h – $1F_h$ is 00000000_2.

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

If an Interrogator changes the UII length (via a memory write operation), and if it wishes the tag to subsequently loadmodulate the new UII length, then it must write the new UII length field into the first 5 bits of the tag's StoredPC.

Note: After changing the UII length field an Interrogator should power-cycle the tag to ensure a correct StoredCRC.

A tag shall loadmodulate an error code (see Annex E) if an Interrogator attempts to write an UII length field, that is not supported by the tag, to the first 5 bits of the tag's PC word.

The PacketPC differs from the StoredPC in its UII length field, which a tag shall adjust to match the length of the loadmodulated data that follow the PC word. Specifically, if XI is asserted but XEB is not asserted then the tag loadmodulates 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 loadmodulates 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 loadmodulated PacketPC but leaves the memory contents unaltered.

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.

During truncated replies a tag substitutes 00000_2 for the PC word — see 6.3.3.4.11.1.1

6.3.3.4.1.2.3 UII for an EPCglobal™ Application

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

6.3.3.4.1.2.4 UII for an ISO Application

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

6.3.3.4.1.2.5 Extended Protocol Control (XPC) word (mandatory) or words (optional)

A tag shall implement an XPC_W1 logically located at addresses 210_h to $21F_h$ of UII memory. A tag may additionally implement an XPC_W2 logically located at address 220_h to $22F_h$ of UII memory. These XPC words shall be exactly 16 bits in length and are stored MSB first. If a tag does not support XPC_W2 then the specified memory locations need not exist.

A tag shall not implement any non-XPC memory element at UII memory locations 210_h to $22F_h$, inclusive. This requirement shall apply both to Tags that support an XPC_W2 word and to those that do not.

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

The remainder of this section 6.3.3.4.1.2.5 assumes that a tag implements an XPC_W1 (mandatory) and an XPC_W2 (optional).

When this document refers to the 3 least-significant-bits (LSBs) of XPC_W1 it specifically means locations $21D_h$, $21E_h$ and $21F_h$ of UII memory. The 3 LSBs of XPC_W1 indicate whether and how a tag was recommissioned.

For virgin tags the 3 LSBs of XPC_W1 shall be zero-valued. A tag writes a nonzero value to one or more of these 3 LSBs during Tag recommissioning (see 6.3.3.4.10). All the other bits in XPC_W1, namely UII memory locations 210_h to $21C_h$, inclusive, as well as all the bits in XPC_W2, shall be RFU and zero-valued. Tag vendors and end users shall not use these RFU bits for proprietary purposes.

The 3 LSBs of the 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 6.3.3.4.11.3.4). A tag shall not write to the 3 LSBs of the XPC_W1 except during Tag recommissioning.

If a tag receives a *Write*, *BlockWrite*, or *BlockErase* that attempts to write the XPC_W1, it responds with an error code (see Annex E).

An Interrogator may issue a *Select* command (see 6.3.3.4.11.1) with a <u>Mask</u> that covers all or part of the XPC_W1 and or XPC_W2. For example, <u>Mask</u> may have the value 000_2 for the 3 LSBs of the XPC_W1, in which case recommissioned Tags shall be non-matching.

An Interrogator may read a tag's XPC_W1 and XPC_W2 using a *Read* command (see 6.3.3.4.11.3.2).

Bit E_h of XPC_W1 (UII memory location 211_h) is reserved for use as a protocol functionality indicator.

The following encoding provides a general mapping between the 3 XPC_W1 LSBs and a tag's recommissioned status. Table 43 — XPC LSBs and a tag's recommissioned status provides a detailed mapping between these 3 LSBs and a tag's recommissioned status:

- **An asserted LSB** (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 LSB also indicates that the *BlockPermalock* command has been disabled. If a tag did not implement block permalocking prior to recommissioning then block permalocking shall remain disabled.
- **An asserted 2SB** (UII memory location $21E_h$) indicates that User memory has been rendered inaccessible. The 2SB has precedence over the LSB — if both are asserted then User memory is inaccessible.
- **An asserted 3SB** (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 left the read lock status of the kill and access passwords in the state prior to the recommissioning. If an Interrogator subsequently attempts to read the tag's kill or access passwords, the tag loadmodulates an error code (see Annex E) if the kill or access password was unreadable prior to the recommisioning. 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.

Table 43 — XPC LSBs and a tag's recommissioned status

LSBs of XPC Word	Tag Status	Notes
000_2	1. The tag has not been recommissioned	1. The tag's XI bit is deasserted
001_2	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 (6.3.3.4.11.3.5)
010_2	1. The User memory bank has been rendered inaccessible 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 inaccessible, so block permalocking and the *BlockPermalock* command are disabled
011_2	1. The User memory bank has been rendered inaccessible 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 inaccessible, so block permalocking and the *BlockPermalock* command are disabled
100_2	1. The UII, TID, and User memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The Read/Lock status of the kill and access passwords shall be the same state as prior to the recommissioning	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 loadmodulating an error code (see Annex E) if the kill or access password was unreadable prior to the recommissioning 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_2	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 Read/Lock status of the kill and access passwords shall be the same state as prior to the recommissioning	1. The lock bits are the sole determinant of the User memory bank's lock status (see 6.3.3.4.11.3.5) 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 loadmodulating an error code (see Annex E) if the kill or access password was unreadable prior to the recommissioning 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_2	1. The UII and TID memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The Read/Lock status of the kill and access passwords shall be the same state as prior to the recommissioning 4. The User memory bank has been rendered inaccessible 5. A tag whose XPC_W1 LSBs are 110_2 acts identically to one whose XPC_W1 LSBs are 111_2	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 loadmodulating an error code (Annex E) if the kill or access password was unreadable prior to the recommisioning 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 inaccessible, so block permalocking and the *BlockPermalock* command are disabled

LSBs of XPC Word	Tag Status	Notes
111_2	1. The UII and TID memory banks have been unlocked 2. The kill and access passwords have been write-unlocked 3. The <u>Read/Lock</u> status of the kill and access passwords shall be the same state as prior to the recommissioning 4. The User memory bank has been rendered inaccessible 5. A tag whose XPC_W1 LSBs are 111_2 acts identically to one whose XPC_W1 LSBs are 110_2	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 loadmodulating an error code (see Annex E) if the kill or access password was unreadable prior to the recommisioning 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 inaccessible, so block permalocking and the *BlockPermalock* command are disabled

6.3.3.4.1.3 TID Memory

TID memory locations 00_h to 07_h shall contain one of two ISO/IEC 15963 class-identifier values — either $E0_h$ or $E2_h$. 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. $E0_h$ or $E2_h$), 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 15693, and TID memory is permalocked at the time of manufacture. If the class identifier is $E2_h$, TID memory location 08_h contains an XTID; TID memory locations 09_h to 13_h contain an 11-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 TID memory above $1F_h$ is defined in version 1.3 and above of the EPCglobal™ Tag Data Standards.

6.3.3.4.1.4 User Memory

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 byte of User memory (i.e. memory addresses 03_h to 07_h) shall have the default value 00000_2.

During recommissioning an Interrogator may instruct a tag to render its User memory inaccessible, causing the entire memory bank to become unreadable, unwriteable, and unselectable. A tag with inaccessible User memory shall function as though its User memory bank no longer exists.

6.3.3.4.1.4.1 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 (version 1.3 and above).

6.3.3.4.1.4.2 User memory for an ISO Application

If User memory is included on a tag, then its encoding shall be as defined in ISO/IEC 15961 and ISO/IEC 15962.

6.3.3.4.2 ASK and PJM Method: Sessions and inventoried flags

Note: An interrogator chooses one of two sessions and inventories tags within that session. The interrogator and associated tag population operate in one and only one session for the duration of an inventory round (defined above). For each session, tags maintain a corresponding inventoried flag. Sessions allow tags to keep track of their inventoried status separately for each of two possible time-interleaved inventory processes, using an independent inventoried flag for each process.

Interrogators shall support and tags shall provide two mandatory sessions (denoted S0 and S2) and two optional sessions (denoted S1 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 session's concept is illustrated in Figure 42 — Session diagram.

Tags shall maintain an independent **inventoried** flag for each session. Each of the two mandatory **inventoried** flags has two values, denoted *A* and *B.* 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.

Sessions allow tags to associate a separate and independent **inventoried** flag to each of several interrogators.

After singulating a tag an interrogator may issue a command that causes the tag to set its **inventoried** flag for that session ($A \rightarrow B$).

Only tags with an inventoried Flag of A respond in an inventory round.

Note: A change from $B \rightarrow A$ can be achieved with a select command. (Reset)

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 S0 to *B*,
  - It powers off.
- Interrogator #2 powers-on, then
  - It initiates an inventory round during which it singulates *A* tags in session S2 to *B*,
  - It powers off.

This process repeats until interrogator #1 has placed all tags in session S0 into *B*. Similarly, interrogator #2 places all tags in session S2 into *B*.

Note: Tags maintain a separate inventoried flag for each of two mandatory sessions.

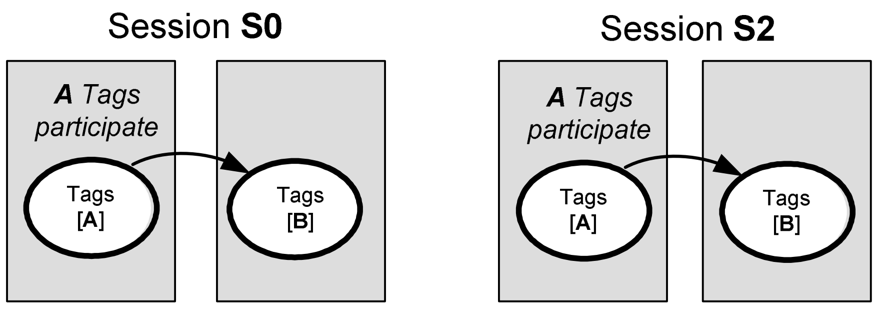


Figure 42 — Session diagram

A tag's **inventoried** flags shall have the persistence times shown in Table 44 — Tag flags and persistence values. 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 (optional) 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 (optional) 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 (optional) 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 (optional) set to *A*.

A tag shall be capable of setting any of its **inventoried** flags to either *A* or *B* in 2ms 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 44 — Tag flags and persistence values.

Table 44 — Tag flags and persistence values

Flag	**Required persistence**
S0 **inventoried** flag (mandatory)	Tag energized: Indefinite Tag not energized: None
S1 **inventoried** flag (optional)	Tag energized: Nominal temperature range: 500ms < persistence < 5s Extended temperature range: Not specified Tag not energized: Nominal temperature range: 500ms < persistence < 5s Extended temperature range: Not specified
S2 **inventoried** flag (mandatory)	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified
S3 **inventoried** flag (optional)	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified
Selected (**SL**) flag	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified

Note 1: Nominal temperature range is –25 °C to +40 °C (see extended temperature range)

Note 2: Extended temperature range is –40 °C to +65 °C (see nominal temperature range)

6.3.3.4.3 ASK and PJM Method: Selected flag

Tags shall implement a selected flag, **SL,** which an interrogator may assert or deassert using a *Select* command. The *Select* parameter in the *BeginRound* command allows an interrogator to inventory tags that have **SL** either asserted or deasserted (*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** applies to all tags regardless of session.

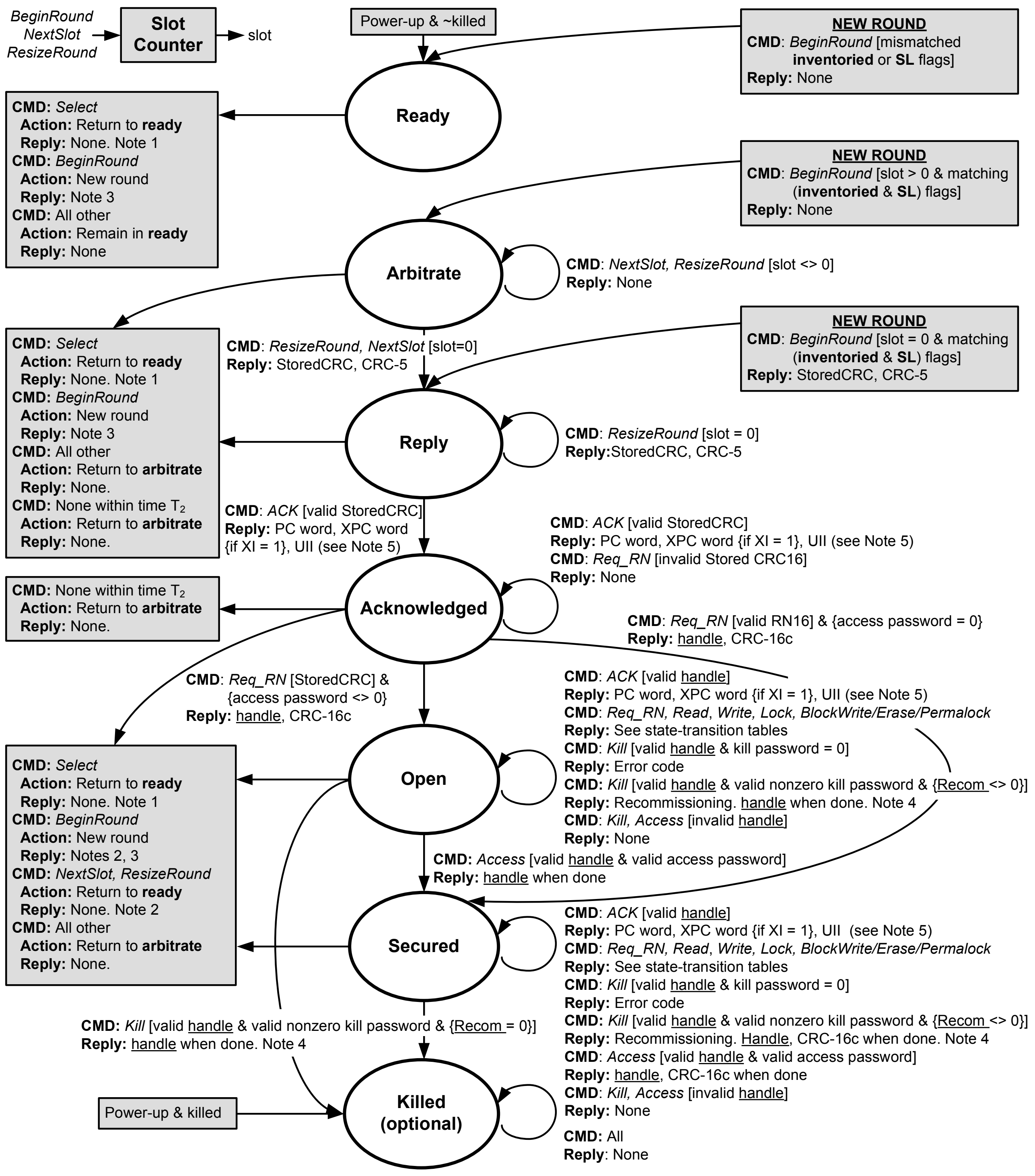
A tag's **SL** flag shall have the persistence times shown in Table 44 — Tag flags and persistence values. A tag shall power-up with its **SL** flag either asserted or deasserted, depending on the stored value, unless the tag has lost power for a time greater than the **SL** persistence time, in which case the tag shall power-up with its **SL** flag deasserted (set to **~SL**). A tag shall be capable of asserting or deasserting its **SL** flag in 2ms 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 44 — Tag flags and persistence values.

6.3.3.4.4 ASK and PJM Method: Tag states and slot counter

Tags shall implement the states and the slot counter shown in Figure 43 — Tag state diagram. Annex C shows the associated state transition tables; Annex D shows the associated command-response tables.

Note: The tag collision arbitration protocol uses the popular and industry proven slotted Aloha algorithm. Slotted ALOHA is a refinement over the pure Aloha. It requires that time be segmented into slots that are separated by slot markers. The probability that a tag chooses a timeslot is equal to 1/n (where "n" is the number of timeslots and equals the maximum value of the described Q-bit random number). This algorithm ensures that each tag, if selected, replies once and only once during an inventory round.

Example: for Q=3 the maximum number of slots is 8, Q is a 3 bit random number, ranging from 0-7, in each slot the probability of a specific tag reply is 1/8.


BeginRound
NextSlot
ResizeRound
Slot Counter
slot
Power-up & ~killed
NEW ROUND
CMD: BeginRound [mismatched inventoried or SL flags]
Reply: None
Ready
CMD: Select
Action: Return to ready
Reply: None. Note 1
CMD: BeginRound
Action: New round
Reply: Note 3
CMD: All other
Action: Remain in ready
Reply: None
NEW ROUND
CMD: BeginRound [slot > 0 & matching (inventoried & SL) flags]
Reply: None
Arbitrate
CMD: NextSlot, ResizeRound [slot <> 0]
Reply: None
CMD: Select
Action: Return to ready
Reply: None. Note 1
CMD: BeginRound
Action: New round
Reply: Note 3
CMD: All other
Action: Return to arbitrate
Reply: None.
CMD: None within time T_2
Action: Return to arbitrate
Reply: None.
CMD: ResizeRound, NextSlot [slot=0]
Reply: StoredCRC, CRC-5
NEW ROUND
CMD: BeginRound [slot = 0 & matching (inventoried & SL) flags]
Reply: StoredCRC, CRC-5
Reply
CMD: ResizeRound [slot = 0]
Reply:StoredCRC, CRC-5
CMD: ACK [valid StoredCRC]
Reply: PC word, XPC word {if XI = 1}, UII (see Note 5)
CMD: ACK [valid StoredCRC]
Reply: PC word, XPC word {if XI = 1}, UII (see Note 5)
CMD: Req_RN [invalid Stored CRC16]
Reply: None
CMD: None within time T_2
Action: Return to arbitrate
Reply: None.
Acknowledged
CMD: Req_RN [valid RN16] & {access password = 0}
Reply: handle, CRC-16c
CMD: Req_RN [StoredCRC] & {access password <> 0}
Reply: handle, CRC-16c
CMD: ACK [valid handle]
Reply: PC word, XPC word {if XI = 1}, UII (see Note 5)
CMD: Req_RN, Read, Write, Lock, BlockWrite/Erase/Permalock
Reply: See state-transition tables
CMD: Kill [valid handle & kill password = 0]
Reply: Error code
CMD: Kill [valid handle & valid nonzero kill password & {Recom <> 0}]
Reply: Recommissioning. handle when done. Note 4
CMD: Kill, Access [invalid handle]
Reply: None
CMD: Select
Action: Return to ready
Reply: None. Note 1
CMD: BeginRound
Action: New round
Reply: Notes 2, 3
CMD: NextSlot, ResizeRound
Action: Return to ready
Reply: None. Note 2
CMD: All other
Action: Return to arbitrate
Reply: None.
Open
CMD: Access [valid handle & valid access password]
Reply: handle when done
Secured
CMD: ACK [valid handle]
Reply: PC word, XPC word {if XI = 1}, UII (see Note 5)
CMD: Req_RN, Read, Write, Lock, BlockWrite/Erase/Permalock
Reply: See state-transition tables
CMD: Kill [valid handle & kill password = 0]
Reply: Error code
CMD: Kill [valid handle & valid nonzero kill password & {Recom <> 0}]
Reply: Recommissioning. Handle, CRC-16c when done. Note 4
CMD: Access [valid handle & valid access password]
Reply: handle, CRC-16c when done
CMD: Kill, Access [invalid handle]
Reply: None
CMD: Kill [valid handle & valid nonzero kill password & {Recom = 0}]
Reply: handle when done. Note 4
Power-up & killed
Killed (optional)
CMD: All
Reply: None

NOTES 1. *Select*: Assert/deassert **SL** or set **inventoried** to *A* or *B*.

2. *BeginRound*: $A \rightarrow B$ if the new session matches the prior session; otherwise no change to the **inventoried** flag.
NextSlot/ResizeRound: $A \rightarrow B$ if the session matches the prior *BeginRound*; otherwise, the command is invalid and ignored by the Tag.
3. *BeginRound* starts a new round and may change the session. Tags may go to **ready**, **arbitrate**, or **reply**.
4. If a Tag does not implement Recom Bits LSB and 2SB then the Tag treats nonzero Recom bits as though Recom = 0.
Two part reply see *Kill*
5. In certain cases a PacketCRC should be added if required. See table 42 for the definition of the cases.

Figure 43 — Tag state diagram

6.3.3.4.4.1 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 *BeginRound* command (see 6.3.3.4.11.2.1) whose <u>inventoried</u> parameter (for the <u>session</u> specified in the *BeginRound)* and *Select* parameter match its current flag values. Matching tags shall draw a *Q*-bit number from their RNG (see 6.3.3.4.5), load this number into their slot counter, and transition to the **arbitrate** state if the number is non-zero, 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.

6.3.3.4.4.2 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 6.3.3.4.4.8) hold non-zero values. A tag in **arbitrate** shall decrement its slot counter every time it receives a *NextSlot* command (see 6.3.3.4.11.2.3) whose <u>session</u> parameter matches the session for the inventory round currently in progress, and it shall transition to the **reply** state 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 *NextSlot* (with matching <u>session</u>) and, because their slot value is now non-zero, shall remain in **arbitrate.**

6.3.3.4.4.3 Reply state

Tags shall implement a **reply** state. Upon entering **reply** a tag shall loadmodulate the StoredCRC. If the tag receives a valid acknowledgement (*ACK*), it shall transition to the **acknowledged** state, loadmodulates its PC word, XPC word (if XI is asserted), UII and PacketCRC if required (see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command). If the tag fails to receive an *ACK*, or receives an invalid *ACK*, it shall return to **arbitrate**. Tag and interrogator shall meet all timing requirements specified in Table 41 — Link timing parameters.

6.3.3.4.4.4 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 43 — Tag state diagram). If a tag in the **acknowledged** state receives a valid *ACK* containing the correct StoredCRC it shall re-loadmodulate the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command. 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 41 — Link timing parameters.

6.3.3.4.4.5 Open state

Tags shall implement an **open** state. A tag in the **acknowledged** state whose access password is non-zero shall transition to **open** upon receiving a *Req_RN* command, loadmodulating an RN16 (denoted <u>handle</u>) that the interrogator shall use in subsequent commands and that 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 43 — Tag state diagram). If a tag in the **open** state receives a valid *ACK* containing the correct <u>handle</u>, it shall re-loadmodulate the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command. Tag and interrogator shall meet all timing requirements specified in Table 41 — Link timing parameters except $T_{2(max)}$; in the **open** state the maximum delay between tag response and interrogator transmission is unrestricted.

6.3.3.4.4.6 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, loadmodulate an RN16 (denoted <u>handle</u>) 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 non-zero shall transition to **secured** upon receiving a valid *Access* command, maintaining the same handle that it previously transmitted 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 state** may transition to any state except **open** or **acknowledged,** depending on the received command (see Figure 43 — Tag state diagram). If a tag in the **secured** state receives a valid *ACK* containing the correct handle, it shall re-loadmodulate the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command. Tag and interrogator shall meet all timing requirements specified in Table 41 — Link timing parameters except $T_{2(max)}$; in the **secured** state the maximum delay between tag response and interrogator transmission is unrestricted.

6.3.3.4.4.7 Killed state (optional)

Tags may implement a **killed** state. If the killed state is supported, a tag in either the **open** or **secured** states shall enter the **killed** state upon receiving a valid *Kill* command (see 6.3.3.4.11.3.4) with a valid non-zero kill password, zero-valued Recom bits (see 6.3.3.4.10), and valid handle. If a tag does not implement Recom bits LSB and 2SB, it then 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 the killed state upon subsequent power-ups. Killing a tag is not reversible.

6.3.3.4.4.8 Slot counter

Tags shall implement a 15-bit slot counter. Upon receiving a *BeginRound* or *ResizeRound* command, a tag shall preload a value between 0 and $2^Q–1$, drawn from the tag RNG (see 6.3.3.4.5), into its slot counter. *Q* is an integer in the range (0, 15). A *BeginRound* specifies *Q*; a *ResizeRound* may modify *Q* from the prior *BeginRound*. Upon receiving a *NextSlot* command a tag shall decrement its slot counter. The slot counter shall be capable of continuous counting: meaning that, after the slot counter decrements to 0000_h, it shall roll over and begin counting down from $7FFF_h$. See also Annex F.

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 slot counter

Tags in the **arbitrate** state decrement their slot counter every time they receive a *NextSlot* with matching session, transitioning to the **reply** state and loadmodulating the StoredCRC 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 *BeginRound* 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 *NextSlot*. 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 F.

Note: An interrogator commands tags in an inventory round to load a Q-bit random (or pseudo-random) number into their slot counter; the interrogator may also command tags to decrement their slot counter. Tags reply when the value in their slot counter (i.e. their slot – see below) is zero. Q is an integer in the range (0, 15); the corresponding tag response probabilities range from $2^0 = 1$ to $2^{-15} = 0,000031$.

6.3.3.4.5 ASK and PJM Method: 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 the PC word, XPC word, UII, and CRC-16). The RNG is used to produce pseudo-random numbers for RN16, Handle, and Slot Counter. Tags that support cover-coding 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 47 — *Kill* procedure or Figure 49 — *Access* procedure).

- **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 ten thousand 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 RNG 10ms after the end of T_r in Figure 19 — ASK Method: Interrogator-to-tag RF envelope 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.

6.3.3.4.6 ASK and PJM Method: Managing tag populations

Interrogators manage tag populations using the three basic operations shown in Figure 44 — Interrogator/tag operations and tag state. Each of these operations comprises one or more commands. The operations are defined as follows:

a. **Select (ASK and PJM Method):** 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.

b. **Inventory ASK Method:** The process by which an interrogator identifies tags. An interrogator begins an inventory round by transmitting a *BeginRound* command in one of two sessions. One or more tags may reply. The interrogator detects a single tag reply and requests the PC word XPC_W1 if XI is set and optionally XPC_W2, UII, and PacketCRC if required (see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command) from the tag. An inventory round operates in one and only one session at a time. Annex H shows an example of an interrogator inventorying and accessing a single tag.

   Inventory PJM Method: The process by which an interrogator identifies tags. An interrogator begins an inventory round by transmitting a *BeginRound* command in one of two sessions. One or more tags may reply. The interrogator detects one or more tag replies and requests the PC word, optional XPC word or words, UII, and CRC-16 from the tag or tags. An inventory round operates in one and only one session at a time. Annex H shows an example of an interrogator inventorying and accessing a single tag or multiple tags.

c. **Access ASK Method:** 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 may employ one-time-pad based cover-coding of the R=>T link.

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

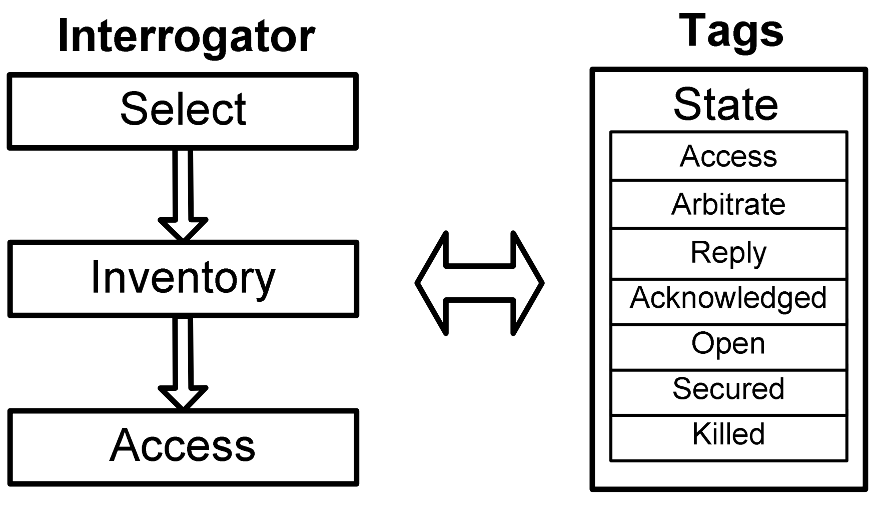


Figure 44 — Interrogator/tag operations and tag state

6.3.3.4.7 ASK and PJM Method: 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 deassert 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.
- Pointerlength, 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 has a length of 8, 16, 24 or 32 bits as defined in Pointerlength. 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.
- Truncate specifies whether a tag loadmodulates its entire UII, or only that portion of the UII immediately following Mask. Truncated replies are always followed by the tag´s StoredCRC; a tag does not recalculate this CRC for a truncated reply.

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 *BeginRound* 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.

6.3.3.4.8 ASK and PJM Method: Inventorying tag populations

The inventory command set includes *BeginRound, ResizeRound, NextSlot, ACK,* and *NAK. BeginRound* initiates an inventory round and decides which tags participate in the round (where “inventory round” is defined as the period between successive *BeginRound* commands).

BeginRound contains a slot-count parameter *Q.* Upon receiving a *BeginRound* participating tags shall pick a random value in the range (0, 2^Q–1), inclusive, and shall load this value into their slot counter. tags that pick a zero shall transition to the **reply** state and reply immediately. tags that pick a non-zero value shall transition to the **arbitrate** state and await a *ResizeRound* or a *NextSlot* command. Assuming that a single tag replies, the *BeginRound*-response algorithm proceeds as follows:

a. The tag loadmodulates the StoredCRC as it enters **reply**,

b. The interrogator acknowledges the tag with an *ACK* containing this same StoredCRC,

c. The acknowledged tag transitions to the **acknowledged** state, loadmodulates the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command

d. The interrogator issues a *ResizeRound* or *NextSlot,* causing the identified tag to set its **inventoried** flag ($A \rightarrow B$) and transition to **ready,** and potentially causing another tag to initiate a *BeginRound*/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 40 — Link timing – Both Modes and Figure 43 — Tag state diagram), or receives the *ACK* with an erroneous StoredCRC, it shall return to **arbitrate.**

If multiple tags reply in step (a) but the interrogator, by detecting and resolving collisions at the waveform level, can resolve an StoredCRC from one of the tags, the interrogator can ACK the resolved tag. Unresolved tags receive erroneous StoredCRCs and return to **arbitrate** without loadmodulating the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command.

If the interrogator sends a valid *ACK (i.e.* an *ACK* containing the correct StoredCRC to the tag in the **acknowledged** state the tag shall re-loadmodulate the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command.

At any point the interrogator may issue a *NAK,* in response to which all tags in the inventory round shall return to **arbitrate** without changing their **inventoried** flag.

After issuing a *BeginRound* to initiate an inventory round, the interrogator typically issues one or more *ResizeRound* or *NextSlot* commands. *ResizeRound* repeats a previous *BeginRound* and may increment or decrement *Q,* but does not introduce new tags into the round. *NextSlot* repeats a previous *BeginRound* without changing any parameters and without introducing new tags into the round. An inventory round can contain multiple *ResizeRound* or *NextSlot* commands. At some point the interrogator shall issue a new *BeginRound,* thereby starting a new inventory round.

Tags in the **arbitrate** or **reply** states that receive a *ResizeRound* 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 shall transition to the **reply** state and reply immediately. Tags that pick a non-zero value shall transition to the **arbitrate** state and await a *ResizeRound* or a *NextSlot* command.

Tags in the **arbitrate** state decrement their slot counter every time they receive a *NextSlot,* transitioning to the **reply** state and loadmodulate the StoredCRC 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 *BeginRound* 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 *NextSlot*, thereby effectively preventing subsequent replies until the tag loads a new random value into its slot counter. Tags shall reply at least once in 2^Q–1 *NextSlot* commands.

Annex G describes an exemplary interrogator algorithm for choosing *Q.*

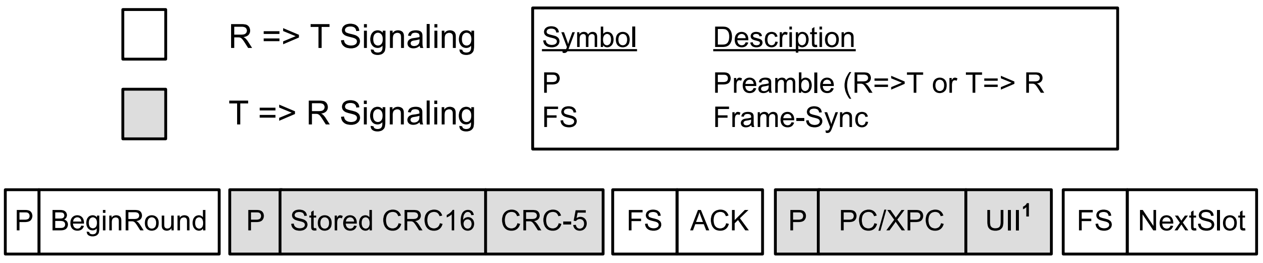
The scenario outlined above assumed a single interrogator operating in a single session. However, as described in 6.3.3.4.2, an interrogator can inventory a tag population in one of two sessions. Furthermore, as described in 6.3.3.4.11.2, the *BeginRound, ResizeRound,* and *NextSlot* commands, each contain a session parameter. How a tag responds to these commands varies with the command, session parameter, and tag state, as follows:

- *BeginRound*: A *BeginRound* command starts an inventory round and chooses the session for the round. Tags in any state except **killed** shall execute a *BeginRound*, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure 43 — Tag state diagram).
    - If a tag in the **acknowledged**, **open**, or **secured** states receives a *BeginRound* whose session parameter matches the prior session it shall set its **inventoried** flag ($A \rightarrow B$) 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 *BeginRound* whose session parameter does not match the prior session it shall leave its **inventoried** flag for the prior session unchanged as it evaluates whether to transition to **ready**, **arbitrate**, or **reply**.
- *ResizeRound*, *NextSlot*: Tags in any state except **ready** or **killed** shall execute a *ResizeRound* or *NextSlot* command if, and only if, the session parameter in the command matches the session parameter in the *BeginRound* that started the round. Tags shall ignore a *ResizeRound* or *NextSlot* with mismatched session.
    - If a tag in the **acknowledged**, **open**, or **secured** states receives a *ResizeRound* or *NextSlot* whose session parameter matches the session parameter in the prior *BeginRound*, it shall set its **inventoried** flag ($A \rightarrow B$) for the current session then transition 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 *BeginRound* specifying **(SL,** *Q* = 4, S0*).* 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 *BeginRound* has 3 possible outcomes:

1) **No tags reply:** The interrogator may issue another *BeginRound,* or it may issue a *ResizeRound* or *NextSlot.*

2) **One tag replies** (see Figure 45 — One tag reply): The tag transitions to the **reply** state and loadmodulates a StoredCRC. The interrogator acknowledges the tag by sending an *ACK.* If the tag receives the *ACK* with a correct StoredCRC it loadmodulates the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command and transitions to the **acknowledged** state. The diagram in assumes that XI is deasserted. If the tag receives the *ACK* with an incorrect StoredCRC, it transitions to **arbitrate.** Assuming a successful *ACK,* the interrogator may either access the acknowledged tag or issue a *ResizeRound* or *NextSlot* to set the tag's **inventoried** flag from *A→B* and send the tag to **ready** (a *BeginRound* with matching prior-round session parameter shall also set the **inventoried** flag from *A→B).*



Note 1: In certain cases a PacketCRC shall be added if required. See table 42 for the definition of the cases.

Figure 45 — One tag reply

3) **Multiple tags reply:** The interrogator observes a loadmodulated waveform comprising multiple StoredCRCs. It may try to resolve the collision and issue an *ACK;* not resolve the collision and issue a *ResizeRound, NextSlot,* or *NAK;* or quickly identify the collision and issue a *ResizeRound* or *NextSlot* before the collided tags have finished loadmodulation. In the latter case the collided tags, not observing a valid reply within T_2 (see Figure 40 — Link timing – Both Modes), shall return to **arbitrate** and await the next *BeginRound* or *ResizeRound* command.

PJM Method: Interrogators may receive multiple tag replies on multiple channels in response to parameters in the *BeginRound, NextSlot or ResizeRound* command. It is possible that such an interrogator may receive (without any clashing error) up to eight tags per *BeginRound, NextSlot or ResizeRound* command. However, with a correctly set Q, compared with the unknown tag population, the average reception rate for an interrogator equipped with eight reply channels would be as follows, per *BeginRound, NextSlot or ResizeRound* command:

- three channels with one tag in each (therefore three tags correctly received, for example, tag 1 on channel A, tag 2 on channel B and tag 3 on channel E),
- one channel with two tags clashing,
- one channel with three tags clashing, and
- three empty channels.

Channel selection is determined by the StoredCRC (see clause 6.3.3.3.1.3.11)

If three tags are correctly received the interrogator can acknowledge all three tags by sending a single *ACK-PJM*, see 6.3.3.4.11.2.4. In the current example, the *ACK-PJM* includes the three tag StoredCRC that have just been received. At the completion of the *ACK-PJM,* if a tag has received a correct StoredCRC it loadmodulates its PC, UII and PacketCRC if required (see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command) using the same reply channel used to respond to the previous *BeginRound, NextSlot* or *ResizeRound* command. The tag then transitions to the **acknowledged** state. In this case the three tags shall simultaneously reply their PC, UII and PacketCRC if required (see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command) each on a different channel (tag 1 on channel A, tag 2 on channel B and tag 3 on channel E). If required, the interrogator can correlate the tag StoredCRC to the UII codes, because each tag uses the same channel for replies to the *BeginRound, NextSlot* or *ResizeRound* commands and the subsequent *ACK-PJM*.

If a tag receives the *ACK-PJM* where all StoredCRCs are incorrect, it transitions to **arbitrate**.

Assuming a successful *ACK-PJM* the interrogator may either access the acknowledged tags or issue a *ResizeRound* or *NextSlot* to set the tag's **inventoried** flags from $A \rightarrow B$ and send the tags to **ready** (a *BeginRound* with prior-round <u>session</u> parameter shall also set the **inventoried** flags from $A \rightarrow B$).

The tags referred to above are the types that are equipped to receive PJM commands.

6.3.3.4.9 ASK and PJM Method: 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, BlockErase* and *BlockPermalock.* Tags execute *Req_RN* from the **acknowledged, open,** or **secured** states. Tags execute *Read, Write, Kill, Access, BlockWrite,* and *BlockErase* from the **open** or **secured** states. Tags execute *Lock* only from the **secured** state.

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 an RN16 (denoted <u>handle</u>), loadmodulates the <u>handle</u>, and transitions to the open state if its access password is non-zero, or transitions 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 <u>handle</u> as a parameter in the command. When in either of these two states, tags shall verify that the <u>handle</u> is correct prior to executing an access command, and shall ignore access commands with an incorrect <u>handle</u>. The <u>handle</u> value is fixed for the entire duration of an access sequence.

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 response to an access command includes, at a minimum, the tag <u>handle;</u> 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 loadmodulate the reply shown in Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command

Interrogator and tag can communicate indefinitely in the **open** or **secured** states. The interrogator may terminate the communications at any time by issuing a *BeginRound, ResizeRound, NextSlot,* or a *NAK.* The tag response to a *BeginRound, ResizeRound,* or *NextSlot* is described in 6.3.3.4.8. 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 could optionally 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 loadmodulating a new RN16. The interrogator then generates a 16-bit cover-coded text 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 cover-coded text string as a parameter.

Step 2. The tag decrypts the received cover-coded text string by performing a bit-wise EXOR of the received 16-bit cover-coded text string with the original RN16.

An interrogator shall not use handle for cover-coding purposes.

An interrogator shall not re-use an RN16 for cover-coding, if it is used. If an interrogator reissues a command that contained cover-coded data, then the interrogator shall reissue the command unchanged. If the interrogator changes the data and uses cover-coding, then it shall first issue a Req_RN to obtain a new RN16 and shall use this new RN16 for cover-coding.

If no RN16 is requested the following 16 bit words (either data or half-passwords) shall be transmitted in plain text without cover-coding. The interrogator shall decide at the beginning of an access sequence whether to use cover-coding or not. If a tag does not support cover-coding it shall respond with 0000_h to each Req_RN command which requests a RN16 for cover-coding.

Table 45 — Access commands and tag states in which they are permitted

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

The *BlockWrite* command (see Table 75 — *BlockWrite* command) 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 to or erases memory (*i.e. Write*, *Kill*, *Lock*, *BlockWrite*, *BlockErase*, and *BlockPermalock* with Read/Lock=1 (see 6.3.3.4.11.3.9)) by loadmodulating its handle, indicating that the operation was successful, or by loadmodulating an error code (see Annex E), indicating that the operation was unsuccessful. The tag reply uses the preamble shown in Figure 27 — ASK Method: FM0 Preamble SOF, Figure 31 — ASK Method: Subcarrier T=>R preamble SOF, and Figure 35 — ASK Method: Miller Subcarrier T=>R preamble SOF, as appropriate. See 6.3.3.4.11.3 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 6.3.3.4.11.3.6 and outlined in Figure 49 — *Access* procedure.

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 6.3.3.4.10, 6.3.3.4.11.3.5, 6.3.3.4.11.3.9, Table 70 — *Lock* command, and Table 80 — *BlockPermalock* command for a detailed description of memory lock, permalock, recommissioning, and the tag state required to modify memory.

Killing a tag is a multi-step procedure, described in 6.3.3.4.11.3.4 and outlined in Figure 47 — *Kill* procedure.

Issuing an access password to a tag is a multi-step procedure, described in 6.3.3.4.11.3.6 and outlined in Figure 49 — *Access* procedure.

Interrogator and tag shall transmit all strings MSB first.

Interrogators shall not power-off while a tag is in the **reply, acknowledged, open** or **secured** states. Interrogators should end their dialog with a tag before powering off, leaving the tag in either the **ready** or **arbitrate** state.

PJM Method: Interrogators can communicate to and receive replies from multiple tags simultaneously.

Note: This is specific to PJM multi-channel method. For *Kill* and *Access* commands the interrogator shall communicate with only one tag at a time. However for *Req_RN, Read, Write, Lock, BlockWrite, BlockErase* and *BlockPermalock* commands the interrogator can extend the RN field to include multiple StoredCRCs or handles, see 6.3.3.4.11, such that the interrogator can communicate with multiple tags at a time. A tag shall reply to such commands using the same channel that was used for its reply to the previous *BeginRound*, *NextSlot* or *ResizeRound* command.

6.3.3.4.10 Killing (optional) or recommissioning a tag

Killing or recommissioning a tag is a multi-step procedure, described in 6.3.3.4.11.3.4 and shown in Figure 47 — *Kill* procedure, 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 re-commission itself without first receiving the correct kill password by the procedure shown in Figure 47 — *Kill* procedure.

If a tag does not implement Recom bits LSB and 2SB then it shall ignore these recommissioning bits and treat them as though they were zero; If the tag receives a properly formatted *Kill* command sequence with the correct kill password and all three Recom bits set to zero and it does not support this command as it is optional it shall interpret it in the same way as with 3SB of the Recom bits is set to one. The remainder of this section 6.3.3.4.10 assumes that a tag implements recommissioning supporting also Recom bits LSB and 2SB.

Upon receiving a properly formatted *Kill* command sequence with the correct kill password and one or more non-zero 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 3SB of its XPC_W1). 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 XPC_W1, a subsequent interrogator can know how the tag was re-commissioned (see Table 43 — XPC LSBs and a tag's recommissioned status). A tag shall perform the following operations based on the recommissioning bit values it receives (see also 6.3.3.4.1.2.5):

- **Asserted LSB:** 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 did not 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 6.3.3.4.11.3.5).
- **Asserted 2SB:** The tag shall render its User memory inaccessible, causing the entire memory bank to become unreadable, unwriteable, and unselectable (*i.e.* the tag functions as though its User memory bank no longer exists).The 2SB has precedence over the LSB — if both are asserted then User memory is inaccessible.
- **Asserted 3SB:** 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 LSB is also asserted, the tag shall unlock its permalocked blocks. The tag shall write-unlock its kill and access passwords. The Read/Lock state of the kill and access passwords shall be the same as prior to the recommissioning. If an Interrogator subsequently attempts to read the tag's kill or access passwords the tag shall loadmodulate an error code (see Annex E) if the kill or access password was unreadable prior to the recommisioning. 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.

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 shall not allow a subsequent recommissioning
- A previously recommissioned tag that receives a properly formatted *Kill* command sequence with the correct kill password and Recom = 000_2 shall be killed
- 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_W1 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 LSBs of its current XPC_W1 and the recommissioning bits, and store the resulting value into its XPC_W1. For example, if a tag whose XPC_W1 bits are 100_2 receives a *Kill* command sequence whose recommissioning bits are 010_2, the tag shall render its User bank inaccessible and store 110_2 into its XPC_W1.

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 Recom bits LSB and 2SB, 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 47 — *Kill* procedure) to a *Kill* command sequence regardless of the RFU or recommissioning bit settings.

A tag shall accept all eight possible combinations of the three recommissioning bits if Recom bits LSB and 2SB are supported, 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 as follows:

- 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, and
- a tag in which a portion of TID memory is factory locked cannot unlock this portion.

6.3.3.4.11 ASK and PJM Method: Interrogator commands and tag replies

Interrogator-to-tag commands shall have the format shown in Table 46 — Commands.
- *ACK* has a 2-bit command code 01_2
- *BeginRound, NextSlot*, *ResizeRound*, and *Select* have 4-bit command codes (see Table 46 — Commands
- 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
- *NextSlot*, *ACK*, *BeginRound*, *ResizeRound*, and *NAK* have the unique command lengths shown in.Table 46 — Commands. No other commands shall have these lengths. If a tag receives one of these commands with an incorrect length it shall ignore the command.
- *BeginRound, ResizeRound,* and *NextSlot* contain a session parameter
- *BeginRound* is protected by a CRC-5, shown in Table 49 — BeginRound command and detailed in Annex I
- *Select*, *Req_RN*, *Read, Write, Kill, Lock, Access, BlockWrite, BlockErase* and BlockPermalock are protected by a CRC-16c, defined in 6.3.3.4.1.4 and detailed in Annex I
- R=>T commands begin with either a preamble or a frame-sync, as described in 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8. The command code lengths specified in Table 46 — Commands 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, or (6) 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 C and Annex E.

Table 46 — Commands

Command	Code	Length (bits)	Mandatory?	Protection
ACK (ASK Method)	01	18	Yes	Unique command length
ACK (PJM Method)	01	>/= 23	Yes	CRC-5
NextSlot	0000	6	Yes	Unique command length
BeginRound	1000	22	Yes	Unique command length and a CRC-5
ResizeRound	1001	9	Yes	Unique command length
Select	1010	> 46	Yes	CRC-16c
Reserved for future use	1011	–	–	–
NAK	11000000	8	Yes	Unique command length
Req_RN	11000001	40	Yes	CRC-16c
Read	11000010	> 59	Yes	CRC-16c
Write	11000011	> 60	Yes	CRC-16c
Kill[1]	11000100	59	Yes	CRC-16c
Lock	11000101	60	Yes	CRC-16c
Access	11000110	56	No	CRC-16c
BlockWrite	11000111	> 59	No	CRC-16c
BlockErase	11001000	> 59	No	CRC-16c
BlockPermalock	11001001	> 68	No	CRC-16c
Reserved for future use	11001010 … 11011111	–	–	–
Reserved for custom commands	11100000 00000000 … 11100000 11111111	–	–	Manufacturer specified
Reserved for proprietary commands	11100001 00000000 … 11100001 11111111	–	–	Manufacturer specified
Reserved for future use	11100010 00000000 … 11101111 11111111	–	–	–

Note 1: For the *Kill* command only the support of asserted Recom bit 3SB is mandatory. See section 6.3.3.4.10

PJM Method: *ACK-PJM*, *Req_RN*, *Read*, *Write*, *Lock*, *BlockWrite*, *BlockErase* and *BlockPermalock* commands can include multiple StoredCRCs, or handles (within the RN field). PJM Method *Kill* and *Access* commands only include a single handle.

Therefore for PJM Method the *ACK-PJM*, *Req_RN*, *Read*, *Write*, *Lock*, *BlockWrite*, *BlockErase* and *BlockPermalock* commands may have different lengths than those shown in Table 46 — Commands. The length of these commands shown in the table only include a single StoredCRC, or handle. This length (or minimum length) shall be increased by 16 bits for each additional StoredCRC or handle.

R=>T commands begin with either a flag (preamble or a frame-sync), as described in 6.3.3.3.1.3.10. The command code lengths specified in Table 46 — Commands do not include the flag.

6.3.3.4.11.1 Select commands

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

6.3.3.4.11.1.1 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 deassert a tag **SL** flag, which applies across all four sessions, or it can set a tag **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 47 — *Select* command. Target shall indicate whether the *Select* modifies a tag **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. Action shall elicit the tag response shown in Table 48 — Tag response to Action parameter. 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 transmitted reply shall be truncated to include only those UII and StoredCRC bits following Mask. *Select* passes the following parameters from interrogator to tags:

- Target indicates whether the *Select* command modifies a tag **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 deassert **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 Re-served 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).
- Pointerlength, 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 has a length of 8, 16, 24 or 32 bits as defined in Pointerlength. 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, in which case the tag shall consider the *Select* to be non-matching.
- Truncate: If an interrogator asserts Truncate, and if a subsequent *BeginRound* specifies Sel=10 or Sel=11, then a matching tag shall truncate its reply to an *ACK* to that portion of the UII immediately following Mask, if remainder of UII length > 0.

  Interrogators shall assert Truncate:

  - in the last (and only in the last) *Select* that the interrogator issues prior to sending a *BeginRound*,
  - 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 assert Truncate only in the last *Select,* and only if this last *Select* targets the **SL** flag. Tags shall power-up with Truncate deasserted.

  Tags shall decide whether to truncate their transmitted 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 loadmodulate SOF followed by 00000_2.and EOF. Truncated replies never include XPC_W1 or an XPC_W2, because Mask must end in the UII. 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 loadmodulate 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 behavior. Specifically, a tag's loadmodulated reply may appear to not match Mask even though the tag's behavior 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 loadmodulate this incremented value in the PacketPC. The tag was matching, but the loadmodulated UII length field appears to be non-matching.

Interrogators shall prefix a *Select* with a frame-sync (see 6.3.3.4.3). The CRC-16c that protects a select is calculated over the first command-code bit to the Truncate bit.

Tags shall not reply to a *Select.*

Note: The interrogator shall not transmit Target 001 and 011.

Table 47 — *Select* command

	Command	**Target**	**Action**	**Mem Bank**	**Pointer length**	**Pointer**	**Length**	**Mask**	**Truncate**	**CR 16**
# of bits	4	3	3	2	2	8, 16, 24 or 32	8	Variable	1	16
Description	1010	000: **Inventoried** (S0) LSB always 0 001: Not Permitted 010: **Inventoried** (S2) LSB always 0 011: Not Permitted 100: **SL** 101: RFU 110: RFU 111: RFU	See Table 48 — Tag response to Action parameter	00: RFU 01: UII 10: TID 11: User	Length of pointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Starting Mask address	Mask length (bits)	Mask value	0: Disable truncation 1: Enable truncation	

Table 48 — Tag response to Action parameter

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

6.3.3.4.11.2 Inventory commands

The inventory command set comprises *BeginRound, ResizeRound, NextSlot, ACK,* and *NAK.*

6.3.3.4.11.2.1 *BeginRound* (mandatory)

Interrogators and tags shall implement the *BeginRound* command shown in Table 49 — *BeginRound* command. *BeginRound* initiates and specifies an inventory round. *BeginRound* includes the following fields:

- DR (divide ratio) sets the T=>R link frequency as described in 6.3.3.3.1.3.9 and Table 36 — ASK Method: Tag-to-interrogator link frequencies,
- M (cycles per symbol) sets the T=>R data rate and modulation format as shown in Table 37 — ASK Method: Tag-to-interrogator data rates,
- TRext chooses whether the T=>R preamble is prefixed with a pilot tone as described in 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8,
- Sel chooses which tags respond to the *BeginRound* (see 6.3.3.4.11.1.1 and 6.3.3.4.8),
- Session chooses a session for the inventory round (see 6.3.3.4.8),
- Q sets the number of slots in the round (see 6.3.3.4.8).

Interrogators shall prefix a *BeginRound* with a preamble.

The CRC-5 is calculated over the first command-code bit to the last Q bit. If a tag receives a *BeginRound* with a CRC-5 error, it shall ignore the command.

Upon receiving a *BeginRound,* tags with matching *Select* and Target 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 *BeginRound,* loads its slot counter with zero, then its reply to a *BeginRound* shall be as shown in Table 50 — Tag reply to a *BeginRound* command; otherwise the tag shall remain silent.

A *BeginRound* 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 *BeginRound* whose session parameter matches the prior session, it shall set its **inventoried** flag (*A*→*B)* for the session. If a tag in the **acknowledged, open,** or **secured** states receives a *BeginRound* 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.

The tags shall support all DR, M and Trext parameters specified in Table 36 — ASK Method: Tag-to-interrogator link frequencies and Table 37 — ASK Method: Tag-to-interrogator data rates, respectively.

PJM Method: The tags shall interpret DR, M, and TRext parameters as specified in Table 38 — PJM Method: Subcarrier selection commands.

Tags in any state other than **killed** shall execute a *BeginRound* command; tags in the **killed** state shall ignore a *BeginRound.*

Table 49 — *BeginRound* command

	Com mand	DR	M	TRext	Sel	Session	RFU	Q	CRC-5
# of bits	4	1	2	1	2	2	1	4	5
description	1000	0: FL=423KHz (fc/32) 1: FL=847KHz (fc/16)	00: FM0 01: Miller 8 10: Manchester 2 11: Manchester 4	0: No pilot tone 1: Use pilot tone	00: All 01: All 10: ~**SL** 11: **SL**	00: S0 01: not permitted 10: S2 11: not permitted	0: A 1: *RFU*	0–15	

Table 50 — Tag reply to a *BeginRound* command

	Reply	CRC-5
# of bits	16	5
description	StoredCRC	

PJM Method: Tags equipped to receive PJM Method shall decode the DR, M, and TRext bits as per Table 38 — PJM Method: Subcarrier selection commands.

6.3.3.4.11.2.2 *ResizeRound* (mandatory)

Interrogators and tags shall implement the *ResizeRound* command shown in Table 51 — *ResizeRound* command. *ResizeRound* adjusts *Q (i.e.* the number of slots in an inventory round – see 6.3.3.4.8) without changing any other round parameters.

ResizeRound includes the following fields:

- Session corroborates the session number for the inventory round (see 6.3.3.4.8 and 6.3.3.4.11.2.1). If a tag receives a *ResizeRound* whose session number is different from the session number in the *BeginRound* 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 *ResizeRound* with an UpDn value different from those specified above it shall ignore the command. If a tag whose *Q* value is 15 receives a *ResizeRound* 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 *ResizeRound* 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 *BeginRound* command that started the inventory round; one or more subsequent *ResizeRound* commands may modify Q.

A *ResizeRound* shall be prefixed with a frame-sync (see 6.3.3.4.8).

Upon receiving a *ResizeRound* 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 *ResizeRound,* loads its slot counter with zero, then its reply to a *ResizeRound* shall be shown in Table 52 — Tag reply to a *ResizeRound* command otherwise, the tag shall remain silent. Tags shall respond to a *ResizeRound* only if they received a prior *BeginRound.*

Tags in the **acknowledged, open,** or **secured** states that receive a *ResizeRound* set their **inventoried** flag (*A*→*B*) for the current session and transition to ready.

Table 51 — *ResizeRound* command

	Command	Session	UpDn
# of bits	4	2	3
description	1001	00: S0 01: Not permitted 10: S2 11: Not permitted	110: Q = Q + 1 000: No change to Q 011: Q = Q – 1

Table 52 — Tag reply to a *ResizeRound* command

	Reply	CRC-5
# of bits	16	5
description	StoredCRC	

6.3.3.4.11.2.3 *NextSlot* (mandatory)

Interrogators and tags shall implement the *NextSlot* command shown in Table 53 — NextSlot command. *NextSlot* instructs tags to decrement their slot counters and, if slot = 0 after decrementing, to loadmodulate the StoredCRC +CRC-5 to the interrogator.

NextSlot includes the following field.

- Session corroborates the session number for the inventory round (see 6.3.3.4.8 and Table 53 — NextSlot command). If a tag receives a *NextSlot* whose session number is different from the session number in the *BeginRound* that initiated the round, it shall ignore the command.

A *NextSlot* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

If a tag, in response to the *NextSlot,* decrements its slot counter and the decremented slot value is zero, then its reply to a *NextSlot* shall be as shown in Table 54 — Tag reply to a *NextSlot* command; otherwise the tag shall remain silent. Tags shall respond to a *NextSlot* only if they received a prior *BeginRound.*

Tags in the **acknowledged, open,** or **secured** states that receive a *NextSlot* set their **inventoried** flag (*A*→*B*) for the current session and transition to **ready.**

Table 53 — NextSlot command

	Command	Session
# of bits	4	2
description	0000	00: S0 01: Not permitted 10: S2 11: Not permitted

Table 54 — Tag reply to a *NextSlot* command

	Reply	CRC-5
# of bits	16	5
description	StoredCRC	

6.3.3.4.11.2.4 *ACK* (mandatory)

Interrogators and tags shall implement the *ACK* command shown in Table 55 — *ACK* command. An interrogator sends an *ACK* to acknowledge a single tag. *ACK* echoes the StoredCRC if the tag is in the reply or in the acknowledged state or the tag handle if the tag is in the **open** or **secured** state.

An *ACK* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

The tag reply to a successful *ACK* shall be as shown in Table 56 — Tag reply to a successful *ACK* command. As described in 6.3.3.4.11.1.1, the reply may be truncated. A tag that receives an *ACK* with an incorrect StoredCRC 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.

Table 55 — *ACK* command

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

Table 56 — Tag reply to a successful *ACK* command

	Reply
# of bits	5 to 528
description	See Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.8 and 6.3.3.4.11.

Interrogators can simultaneously receive replies from up to eight tags at a time. This mode allows for these multiply received tags to all be addressed in certain commands by extending the RN field to include multiple StoredCRCs or handles. To acknowledge a tag or tags, this mode uses a *ACK-PJM* shown in Figure 40 — Link timing – Both Modes. This command includes one or more echoed StoredCRCs or handles. The selection of StoredCRC or handle shall be as described above for ASK Method.

One or more tags shall reply to a successful *ACK-PJM* as shown in Table 56 — Tag reply to a successful *ACK* command. The tag or tags response to the *ACK-PJM* shall be as described above for ASK Method.

Table 57 — *ACK*-PJM command

	Command	RN	CRC-5
# of bits	2	Number of tags to be acknowledged multiplied by 16	5
description	01	Echoed StoredCRCs or handles	

6.3.3.4.11.2.5 *NAK* (mandatory)

Interrogators and tags shall implement the *NAK* command shown in Table 58 — NAK command. *NAK* shall return all tags to the **arbitrate** state unless they are in **ready** or **killed,** in which case they shall ignore the *NAK* and remain in their current state.

A *NAK* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

Tags shall not reply to a *NAK.*

Table 58 — NAK command

	Command
# of bits	8
description	11000000

6.3.3.4.11.3 Access commands

The set of access commands comprises *Req_RN, Read, Write, Kill, Lock, Access, BlockWrite,* and *BlockErase*, and *BlockPermalock*. As described in 6.3.3.4.9, 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 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 duration of an access sequence.

A tag 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.

After an interrogator writes to a tag´s StoredPC or UII, or changes bits 03_h to 07_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 first powers-down and then re-powers-up its energizing RF field, because the tag calculates its StoredCRC at power-up.

If an Interrogator attempts to write to UII memory locations 00_h to $0F_h$ (*i.e.* to the StoredCRC) using a *Write*, *BlockWrite*, or *BlockErase* command the tag shall ignore the command and respond with an error code (see Annex E 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_W1 or XPC_W2) of a tag using a *Write*, *BlockWrite*, or *BlockErase* command the tag shall ignore the command and respond with an error code (see Annex E 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, 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 E for error-code definitions and for the reply format).

Req_RN, Read, Write, *Kill* (only with asserted Recom bit 3SB), and *Lock* are required commands; *Access*, *BlockWrite*, *BlockErase*, and *BlockPermalock* are optional. Tags shall ignore optional access commands that they do 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.

6.3.3.4.11.3.1 *Req_RN* (mandatory)

Interrogators and tags shall implement the *Req_RN* command shown in Table 59 — Req_RN command. *Req_RN* instructs a tag to loadmodulate a new RN16. Both the interrogator's command, and the tag's response, depends 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 StoredCRC as a parameter in the *Req_RN*. The *Req_RN* command is protected by a CRC-16c calculated over the command bits and the StoredCRC. If the tag receives the *Req_RN* with a valid CRC-16c and a valid StoredCRC it shall generate and store an RN16 (denoted handle), loadmodulate 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-16c but an invalid StoredCRC 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 handle as a parameter in the *Req_RN.* The *Req_RN* command is protected by a CRC-16c calculated over the command bits and the handle. If the tag receives the *Req_RN* with a valid CRC-16c and a valid handle, it shall generate and loadmodulate a new RN16. If the tag receives the *Req_RN* with a valid CRC-16c 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 loadmodulate 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 transmitted RN16 shall be denoted the MSB; the last bit shall be denoted the LSB.

A *Req_RN* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

The tag reply to a Req_RN shall be as shown in Table 60 — Tag reply to a Req_RN command. The StoredCRC and handle are protected by a CRC-16c.

Table 59 — Req_RN command

	Command	RN	CRC-16c
# of bits	8	16	16
description	11000001	StoredCRC or handle	

Table 60 — Tag reply to a Req_RN command

	RN	CRC-16c
# of bits	16	16
description	New RN16 or handle	

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11, As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple StoredCRCs or handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.2 *Read* (mandatory)

Interrogators and tags shall implement the *Read* command shown in Table 62 — *Read* command. *Read* allows an interrogator to read part or all of a tag 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.
- WordPtrlength, 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 has a length of 8, 16, 24 or 32 bits as defined in WordPtrlength.
- WordCount specifies the number of 16-bit words to be read. If WordCount = 00_h the tag shall loadmodulate the contents of the chosen memory bank starting at WordPtr and ending at the end of the bank, unless MemBank = 01_2, in which case the tag shall loadmodulate the UII memory contents specified in Table 61 — Tag loadmodulation when WordCount=00h and MemBank=012.

Table 61 — Tag loadmodulation when WordCount=00_h and MemBank=01_2

WordPtr Memory Address	Tag Implements XPC_W2?	What the tag Loadmodulates
Within the StoredCRC, StoredPC, or the UII specified by bits 10_h–14_h of the StoredPC	Do not 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	UII memory starting at WordPtr and ending at the end of physical UII memory, including 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	UII memory starting at WordPtr and ending at the end of physical UII memory, including 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 address 220_h
210_h. Above physical UII memory.	No	XPC_W1
210_h. Above physical UII memory.	Yes	XPC_W1 and XPC_W2
220_h. Above physical UII memory.	No	Error code
220_h. Above physical UII memory.	Yes	XPC_W2
Not 210_h or 220_h. Above physical UII memory.	Do not care	Error code

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

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

A *Read* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

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 63 — Tag reply to a successful *Read* command. The tag responds by loadmodulating a header (a 0-bit), the requested memory words, and its handle. The reply includes a CRC-16c calculated over the 0-bit, memory words, and handle.

If a one or more of the memory words specified in the *Read* command either do not exist or are read-locked, the tag shall loadmodulate an error code, within time T_1 in Table 41 — Link timing parameters, rather than the reply shown in Table 63 — Tag reply to a successful *Read* command (see Annex E for error-code definitions and for the reply format).

Table 62 — *Read* command

	Command	**MemBank**	**WordPtrLength**	**WordPtr**	**WordCount**	**RN**	**CRC-16c**
# of bits	8	2	2	8, 16, 24 or 32	8	16	16
description	11000010	00: Reserved 01: UII 10: TID 11: User	Length of Wordpointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Starting address pointer	Number of words to read	handle	

Table 63 — Tag reply to a successful *Read* command

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

Optionally the interrogators shall communicate using PJM.

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.3 Write (mandatory)

Interrogators and tags shall implement the *Write* command shown in Table 64 — *Write* command. *Write* allows an interrogator to write a word in a tag Reserved, UII, TID, or User memory. *Write* has the following fields:

- MemBank specifies whether the *Write* occurs in Reserved, UII, TID, or User memory.
- WordPtrlength, 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 has a length of 8, 16, 24 or 32 bits as defined in the WordPtrlength.

- Data contains a 16-bit word to be written. Before each and every *Write* the interrogator may optionally first issue a *Req_RN* command; the tag responds by loadmodulating a new RN16. The interrogator shall cover-code the data by EXORing it with this new RN16 prior to transmission if a *Req_RN* command was issued.

The *Write* command also includes the tag handle and a CRC-16c. The CRC-16c 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-16c but an invalid handle, shall ignore the *Write* and remain in its current state. If it receives a *Write* where the immediately preceding command was not a *Req_RN,* it shall interpret the data field as plain data.

A *Write* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8). 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 transmitted reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 μs so that T_{REPLY} = n*1024/fc +/- 32/fc. An interrogator may observe several possible outcomes from a *Write,* depending on the success or failure of the tag memory-write operation:

- **The *Write* succeeds:** After completing the *Write,* a tag shall loadmodulate the reply shown in Table 65 — Tag reply to a successful *Write* command and Figure 46 — Successful *Write* sequence comprising a header (a 0-bit), the tag handle, and a CRC-16c calculated over the 0-bit and handle. If the interrogator observes this reply within 20ms then the *Write* completed successfully.
- **The tag encounters an error:** The tag shall loadmodulate an error code during the CW period rather than the reply shown in Figure 46 — Successful *Write* sequence (see Annex E 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 handle) to verify that the tag is still in the interrogator's field if cover-coding is used, and may reissue the *Write* command regardless the use of cover-coding.

Upon receiving a valid *Write* command, a tag shall write the commanded Data into memory. The tag reply to a successful *Write* shall use the preamble as specified in the *BeginRound* command that initiated the round.

Table 64 — *Write* command

	Command	MemBank	WordPtrLength	WordPtr	Data	RN	CRC-16c
# of bits	8	2	2	8, 16, 24 or 32	16	16	16
description	11000011	00: Reserved 01: UII 10: TID 11: User	Length of Wordpointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Address pointer	RN16 ⊗ word to be written or word to be written	handle	

Table 65 — Tag reply to a successful *Write* command

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

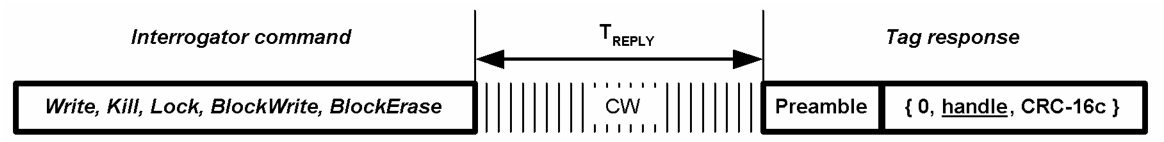


Figure 46 — Successful *Write* sequence

Optionally the interrogators shall communicate using PJM.

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.4 *Kill* (only support of asserted Recom bit 3SB is mandatory)

ASK Method and PJM Method: Interrogators and tags shall implement the *Kill* command (only support of asserted Recom bit 3SB is mandatory) shown in Table 66 — First *Kill* command and Figure 47 — *Kill* procedure. *Kill* allows an interrogator to permanently disable a tag. *Kill* also allows an Interrogator to recommission tags.

To kill or recommission a tag, an interrogator shall follow the multi-step procedure outlined in Figure 47 — *Kill* procedure. Briefly, an interrogator issues two *Kill* commands, the first containing the 16 MSBs of the tag kill password optionally EXORed with an RN16, and the second containing the 16 LSBs of the tag kill password optionally 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 if cover-coding is used, otherwise no *Req_RN* command shall be issued.

Tags shall incorporate the necessary logic to successively accept two 16-bit sub-portions 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 command other than *Req_RN* before the second *Kill,* it shall return to **arbitrate,** unless the intervening command is a *BeginRound,* in which case the tag shall execute the *BeginRound* (setting its **inventoried** flag if the session parameter in the *BeginRound* matches the prior session).

Kill contains 3 RFU/Recom bits. In the first *Kill* command these bits are RFU. Interrogators shall set the 3 RFU bits to 000_2 when communicating with the tags, and tags shall ignore those bits. As described in 6.3.3.4.10 in the second *Kill* command, the 3 RFU 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 (optional) and are nonzero when recommissioning (support of Recom bit 3SB is mandatory) a tag. 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 6.3.3.4.10.

If a tag does not implement Recom bits LSB and 2SB (see 6.3.3.4.10), then the tag ignores the LSB and 2SB recommissioning bits and treats them as though they were zero, If the tag receives a properly formatted *Kill* command sequence with the correct kill password and all 3 Recom bits set to zero, and the tag does not support the second *Kill* command with all 3 Recom bits set to zero - because it is optional, the tag shall interpret this *Kill* command sequence in the same way as if the 3SB of the Recom bits is set to one. If the tag does not support killing, then the tag recommissions itself regardless of the values of the recommissioning bits if recommissioning is permitted.

A tag whose kill password is zero, does not execute a kill or a recommissioning operation; if such a tag receives a *Kill* command, it ignores the command and loadmodulates an error code (see Figure 47 — *Kill* procedure).

The tag reply to the first *Kill command* shall be as shown in Table 68 — Tag reply to the first *Kill* command. The reply shall use the TRext value specified in the *BeginRound* 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 transmitted reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4µs so that T_{REPLY} = n*1024/f_c +/- 32/f_c. An interrogator may observe several possible outcomes from a *Kill* command sequence*,* depending on the success or failure of the tag kill or recommissioning operation:

- **The *Kill* or recommissioning succeeds:** After completing the operation the tag shall loadmodulate the reply shown in Table 69 — Tag reply to a successful *Kill* procedure and Figure 47 — *Kill* procedure comprising a header (a 0-bit), the tag handle, and a CRC-16c calculated over the 0-bit and handle. If the Interrogator observes this reply within 20ms then the operation completed successfully. If the tag is killed, 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 loadmodulate an error code during the CW period rather than the reply shown in Table 69 — Tag reply to a successful *Kill* procedure (see Annex E 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 20 ms then the operation did not complete successfully. The interrogator may issue a *Req_RN* command (containing the tag handle) to verify that the tag is still in the interrogator's field if cover-coding is used, and may reinitiate the multi-step kill procedure outlined in Figure 47 — *Kill* procedure regardless the use of cover-coding.

A *Kill* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

Upon receiving a valid *Kill* command sequence a tag shall render itself killed or recommissioned as appropriate. The tag reply to the second *kill* command shall use the preamble as specified in the *BeginRound* command that initiated the round.

Table 66 — First *Kill* command

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

Table 67 — Second *Kill* command

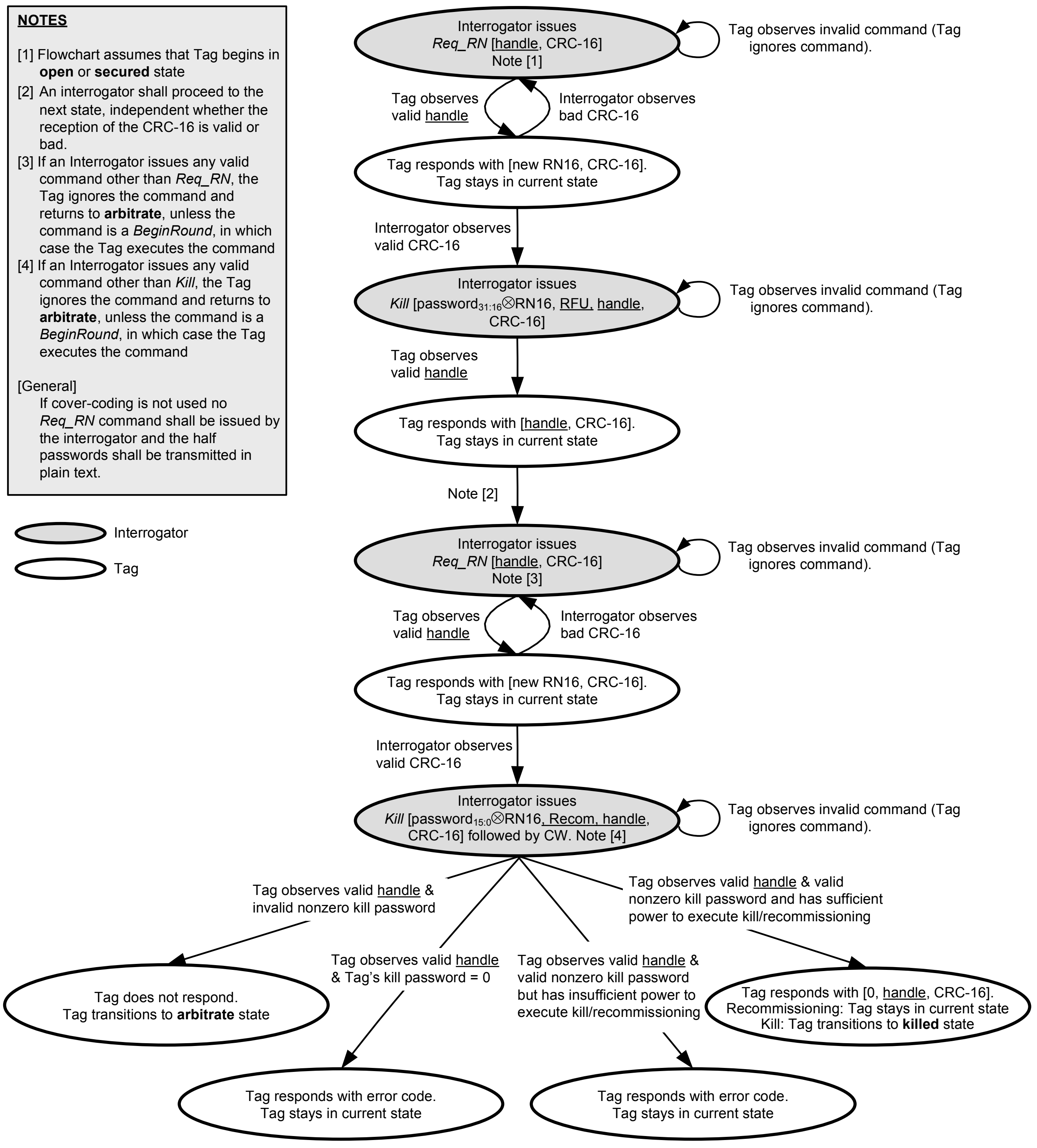
	Command	Password	RFU/Recom	RN	CRC-16c
# of bits	8	16	3	16	16
description	11000100	(½ kill password) ⊗ RN16 or plain	Recommissioning bits (see 6.3.3.4.10)	handle	

Table 68 — Tag reply to the first *Kill* command

	RN	CRC-16c
# of bits	16	16
description	handle	

Table 69 — Tag reply to a successful *Kill* procedure

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


NOTES

[1] Flowchart assumes that Tag begins in **open** or **secured** state

[2] An interrogator shall proceed to the next state, independent whether the reception of the CRC-16 is valid or bad.

[3] If an Interrogator issues any valid command other than *Req_RN*, the Tag ignores the command and returns to **arbitrate**, unless the command is a *BeginRound*, in which case the Tag executes the command

[4] If an Interrogator issues any valid command other than *Kill*, the Tag ignores the command and returns to **arbitrate**, unless the command is a *BeginRound*, in which case the Tag executes the command

[General]
If cover-coding is not used no *Req_RN* command shall be issued by the interrogator and the half passwords shall be transmitted in plain text.

Interrogator
Tag
Interrogator issues *Req_RN* [handle, CRC-16] Note [1]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Interrogator observes bad CRC-16
Tag responds with [new RN16, CRC-16]. Tag stays in current state
Interrogator observes valid CRC-16
Interrogator issues *Kill* [password$_{31:16} \otimes$RN16, RFU, handle, CRC-16]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Tag responds with [handle, CRC-16]. Tag stays in current state
Note [2]
Interrogator issues *Req_RN* [handle, CRC-16] Note [3]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Interrogator observes bad CRC-16
Tag responds with [new RN16, CRC-16]. Tag stays in current state
Interrogator observes valid CRC-16
Interrogator issues *Kill* [password$_{15:0} \otimes$RN16, Recom, handle, CRC-16] followed by CW. Note [4]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle & invalid nonzero kill password
Tag observes valid handle & valid nonzero kill password and has sufficient power to execute kill/recommissioning
Tag observes valid handle & Tag's kill password = 0
Tag observes valid handle & valid nonzero kill password but has insufficient power to execute kill/recommissioning
Tag does not respond. Tag transitions to **arbitrate** state
Tag responds with [0, handle, CRC-16]. Recommissioning: Tag stays in current state Kill: Tag transitions to **killed** state
Tag responds with error code. Tag stays in current state
Tag responds with error code. Tag stays in current state

Figure 47 — *Kill* procedure

6.3.3.4.11.3.5 *Lock* (mandatory)

Interrogators and tags shall implement the *Lock* command shown in Table 70 — *Lock* command and Figure 48 — *Lock* payload and usage. 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: Deassert 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 72 — Lock *Action-field* functionality.

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 loadmodulate an error code (see Annex E).

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 79 — Precedence for *Lock* and *BlockPermalock* commands 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 deasserted except by recommissioning the tag (see 6.3.3.4.10). If a tag receives a *Lock* whose payload attempts to deassert a previously asserted permalock bit, the tag shall ignore the *Lock* and loadmodulate an error code (see Annex E). 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, unless the tag has been recommissioned and the corresponding memory location is no longer permalocked, in which case the tag shall reassert the permalock bit.

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 48 — *Lock* payload and usage, 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 loadmodulate an error code (see Annex E). 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 loadmodulate an error code for any payload other than $FFFFF_h$.

A *Lock* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

After issuing a *Lock* an interrogator shall transmit CW for the lesser of T_{REPLY} or 20 ms, where T_{REPLY} is the time between the interrogator's *Lock* command and the tag transmitted reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 µs so that T_{REPLY} = n*1024/fc +/- 32/fc. An interrogator may observe several possible outcomes from a *Lock,* depending on the success or failure of the tag memory-write operation:

- **The *Lock* succeeds:** After completing the *Lock* the tag shall loadmodulate the reply shown in Table 71 — Tag reply to a *Lock* command and Figure 46 — Successful *Write* sequence comprising a header (a 0-bit), the tag handle and a CRC-16c 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 loadmodulate an error code during the CW period rather than the reply shown in Table 71 — Tag reply to a *Lock* command (see Annex E 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 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 reply to a Lock shall use the preamble as specified in the *BeginRound* command that initiated the round.

Table 70 — *Lock* command

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

Table 71 — Tag reply to a *Lock* command

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

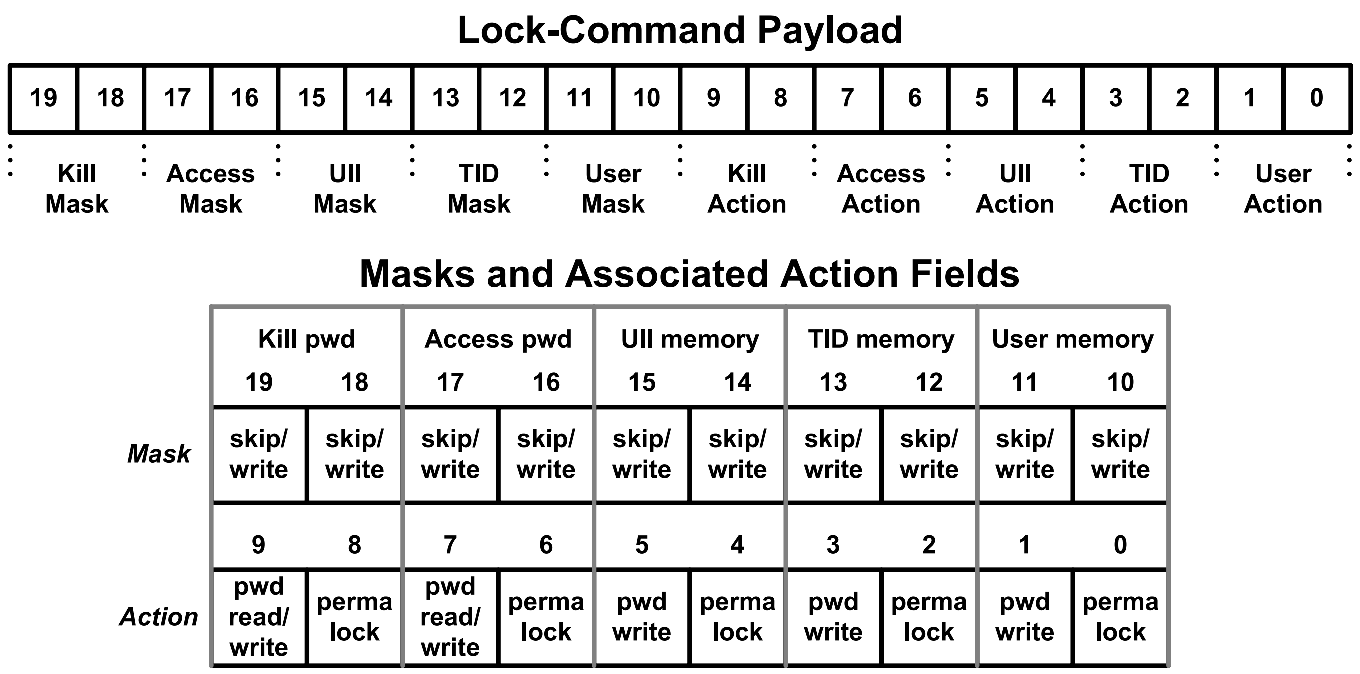


Figure 48 — *Lock* payload and usage

Table 72 — 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 **open** or **secured** 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.

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.6 *Access* (optional)

ASK Method and PJM Method: Interrogators and tags may implement an Access command; if they do, the command shall be as shown in Table 73 — Access command. *Access* causes a tag with a non-zero-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 43 — Tag state diagram) or, if the tag is already in the **secured** state, to remain **secured.**

To access a tag, an interrogator shall follow the multi-step procedure outlined in Figure 49 — *Access* procedure. Briefly, an interrogator issues two Access commands, the first containing the 16 MSBs of the tag access password optionally EXORed with an RN16, and the second containing the 16 LSBs of the tag access password optionally 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 if cover-coding is used otherwise no Req_RN command shall be issued.

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

An *Access* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

The tag reply to an *Access* command shall be as shown in Table 74 — Tag reply to an *Access* command. If the *Access* is the first in the sequence, then the tag loadmodulates 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 loadmodulates its handle to acknowledge that it has executed the command successfully and has transitioned to the **secured** state; otherwise the tag does not reply. The reply includes a CRC-16c calculated over the handle.

Table 73 — Access command

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

Table 74 — Tag reply to an *Access* command

	RN	CRC-16c
# of bits	16	16
description	handle	

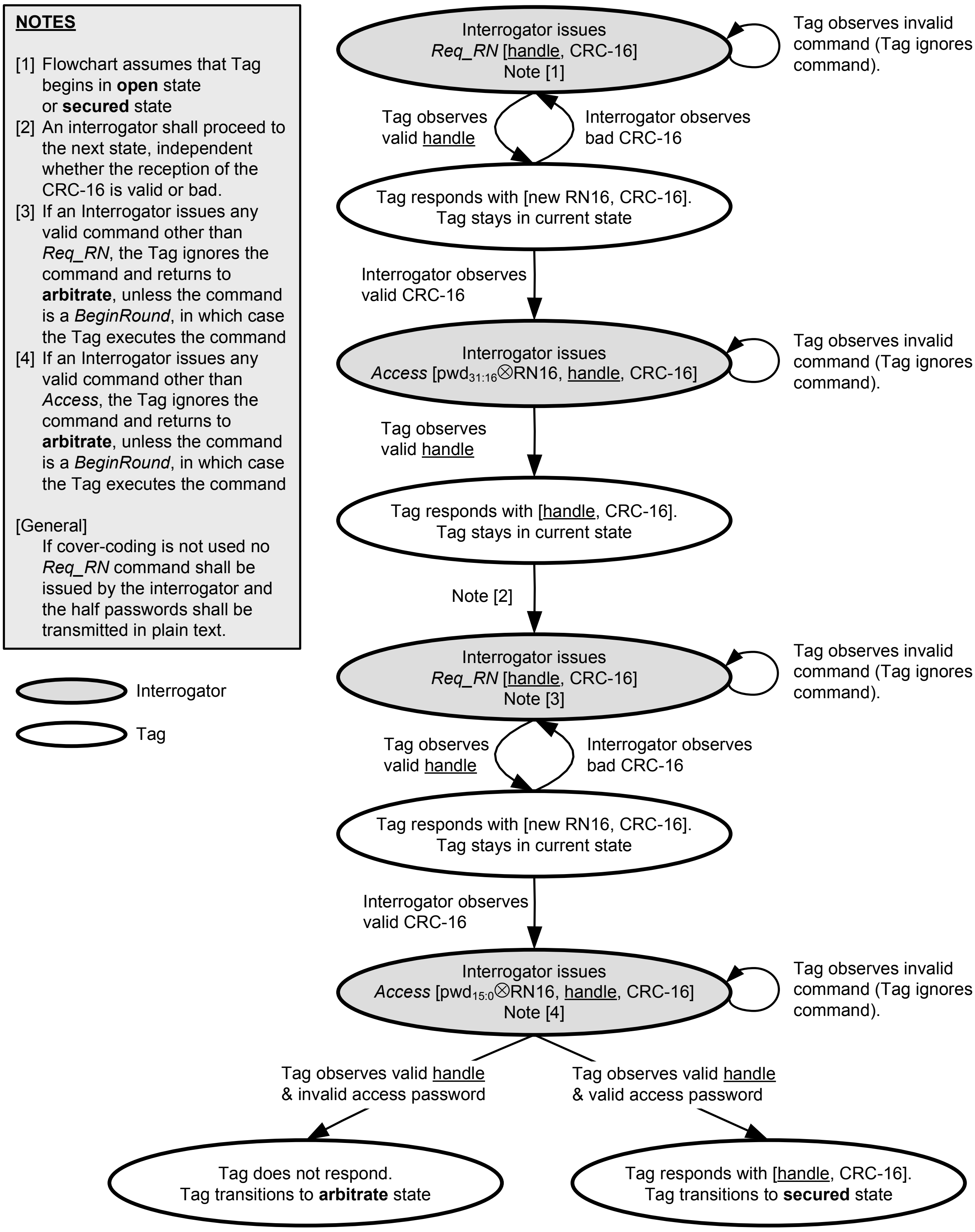

NOTES
[1] Flowchart assumes that Tag begins in open state or secured state
[2] An interrogator shall proceed to the next state, independent whether the reception of the CRC-16 is valid or bad.
[3] If an Interrogator issues any valid command other than Req_RN, the Tag ignores the command and returns to arbitrate, unless the command is a BeginRound, in which case the Tag executes the command
[4] If an Interrogator issues any valid command other than Access, the Tag ignores the command and returns to arbitrate, unless the command is a BeginRound, in which case the Tag executes the command
[General] If cover-coding is not used no Req_RN command shall be issued by the interrogator and the half passwords shall be transmitted in plain text.
Interrogator
Tag
Interrogator issues Req_RN [handle, CRC-16] Note [1]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Interrogator observes bad CRC-16
Tag responds with [new RN16, CRC-16]. Tag stays in current state
Interrogator observes valid CRC-16
Interrogator issues Access [pwd31:16⊗RN16, handle, CRC-16]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Tag responds with [handle, CRC-16]. Tag stays in current state
Note [2]
Interrogator issues Req_RN [handle, CRC-16] Note [3]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle
Interrogator observes bad CRC-16
Tag responds with [new RN16, CRC-16]. Tag stays in current state
Interrogator observes valid CRC-16
Interrogator issues Access [pwd15:0⊗RN16, handle, CRC-16] Note [4]
Tag observes invalid command (Tag ignores command).
Tag observes valid handle & invalid access password
Tag observes valid handle & valid access password
Tag does not respond. Tag transitions to arbitrate state
Tag responds with [handle, CRC-16]. Tag transitions to secured state

Figure 49 — *Access* procedure

6.3.3.4.11.3.7 BlockWrite (optional)

Interrogators and tags may implement a *BlockWrite* command; if they do, they shall implement it as shown in Table 75 — *BlockWrite* command. *BlockWrite* allows an interrogator to write multiple words in a tag 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.
- WordPtrlength, 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 has a length of 8, 16, 24 or 32 bits as defined in WordPtrlength.
- 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 handle and a CRC-16c. The CRC-16c is calculated over the first command-code bit to the last handle bit.

If a tag receives a *BlockWrite* with a valid CRC-16c but an invalid handle, it shall ignore the *BlockWrite* and remain in its current state **(open** or **secured,** as appropriate). A *BlockWrite* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

After issuing a *BlockWrite*, an interrogator shall transmit CW for the lesser of T_{REPLY} or 20 ms, where T_{REPLY} is the time between the interrogator's *BlockWrite* command and the tag transmitted reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 µs so that T_{REPLY} = n*1024/fc +/- 32/fc. An interrogator may observe several possible outcomes from a *BlockWrite,* depending on the success or failure of the tag memory write operation:

- **The *BlockWrite* succeeds:** After completing the *BlockWrite* a tag shall loadmodulate the reply shown in Table 76 — Tag reply to a successful *BlockWrite* command and Figure 46 — Successful *Write* sequence comprising a header (a 0-bit), the tag handle and a CRC-16c 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 loadmodulate an error code during the CW period rather than the reply shown in Table 76 — Tag reply to a successful *BlockWrite* command (see Annex E 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 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 reply to a *BlockWrite* shall use the preamble as specified in the *BeginRound* command that initiated the round.

Table 75 — *BlockWrite* command

	Command	MemBank	WordPtrLength	WordPtr	WordCount	Data	RN	CRC-16c
# of bits	8	2	2	8, 16, 24 or 32	8	Variable	16	16
Descrip tion	11000111	00: Reserved 01: UII 10: TID 11: User	Length of Wordpointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Starting address pointer	Number of words to write	Data to be written	hand le	

Table 76 — Tag reply to a successful *BlockWrite* command

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

Optionally the interrogators shall communicate using PJM

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.8 *BlockErase* (optional)

Interrogators and tags may implement a *BlockErase* command; if they do, they shall implement it as shown in Table 77 — BlockErase command. *BlockErase* allows an interrogator to erase multiple words in a tag 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.
- WordPtrlength, 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 has a length of 8, 16, 24 or 32 bits as defined in WordPtrlength.
- 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 handle and a CRC-16c. The CRC-16c is calculated over the first command-code bit to the last handle bit.

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

A *BlockErase* shall be prefixed with a frame-sync (see 6.3.3.3.1.3.4, 6.3.3.3.1.3.6 and 6.3.3.3.1.3.8).

After issuing a *BlockErase* an interrogator shall transmit CW for the lesser of T_{REPLY} or 20 ms, where T_{REPLY} is the time between the interrogator's *BlockErase* command and the tag transmitted reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 μs so that T_{REPLY} = n*1024/fc +/- 32/fc. An interrogator may observe several possible outcomes from a *BlockErase,* depending on the success or failure of the tag memory erase operation:

- **The *BlockErase* succeeds:** After completing the *BlockErase* a tag shall loadmodulate the reply shown in Table 65 — Tag reply to a successful *Write* command and Figure 46 — Successful *Write* sequence comprising a header (a 0-bit), the tag handle and a CRC-16c 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 loadmodulate an error code during the CW period rather than the reply shown in Figure 46 — Successful *Write* sequence (see Annex C for error-code definitions and for the reply format).
- **The *BlockErase* does not succeed:** If the interrogator does not observe a reply within 20 ms then the *BlockErase* did not complete successfully. The interrogator may issue a *Req_RN* command (containing the tag 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 reply to a *BlockErase* shall use the preamble as specified in the *BeginRound* command that initiated the round.

Table 77 — BlockErase command

	Command	MemBank	WordPtrLength	WordPtr	WordCount	RN	CRC-16c
# of bits	8	2	2	8, 16, 24 or 32	8	16	16
description	11001000	00: Reserved 01: UII 10: TID 11: User	Length of Wordpointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Starting address Pointer	Number of words to write	handle	

Table 78 — Tag reply to a successful BlockErase command

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

Optionally the interrogators shall communicate using PJM.

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

6.3.3.4.11.3.9 *BlockPermalock* (optional)

Interrogators and tags may implement a *BlockPermalock* command; if they do, they shall implement it as shown in Table 80 — *BlockPermalock* command. *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. 6.3.3.4.11.3.9 specifies how a tag shall react to a *BlockPermalock* command (with Read/Lock = 1) that follows a prior *Lock* command, or vice versa.

Table 79 — 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 6.3.3.4.11.3.9
	0	1				Reject the *BlockPermalock*; respond with an error code
	1	0				Permalock the blocks indicated by Mask; respond as described in this section 6.3.3.4.11.3.9
	1	1				Permalock the blocks indicated by Mask; respond as described in this section 6.3.3.4.11.3.9
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 6.3.3.4.11.3.5
				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 6.3.3.4.11.3.5
				1	1	Implement the *Lock*; respond as described in 6.3.3.4.11.3.5

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. Tags shall only execute a *BlockPermalock* command if MemBank = 11_2 (User memory); if a tag receives a *BlockPermalock* with MemBank<>11_2 it shall ignore the command and instead loadmodulation an error code (see Annex E), remaining in the **secured** state.
- Read/Lock specifies whether a tag loadmodulations 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 loadmodulation 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 loadmodulation 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.
- BlockPtrlength, BlockPtr specifies the starting address for Mask, in units of 16 blocks. For example, BlockPtr = 00_h indicates block 0, BlockPtr = 01_h indicates block 16, and BlockPtr = 02_h indicates block 32. BlockPtr has a length of 8, 16, 24 or 32 bits as defined in BlockPtrlength.
- BlockRange specifies the range of Mask, starting at BlockPtr and ending (16×BlockRange)–1 blocks later. If BlockRange=00_h, then the tag shall ignore the *BlockPermalock* command and instead backscatter an error code (see Annex E), 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 6.3.3.4.10).

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 00_h when communicating with 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 E), remaining in the **secured** state.

The *BlockPermalock* command also includes the tag's handle and a CRC-16c. The CRC-16c is calculated over the first command-code bit to the last handle bit. If a tag receives a *BlockPermalock* with a valid CRC-16c 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 2SB of a tag's XPC word is/are asserted (see Table 38), or in which one of the asserted Mask bits references a non-existent block, then the tag shall ignore the *BlockPermalock* command and instead loadmodulate an error code (see Annex E), 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 6.3.3.4.11 for the definition of an "invalid" command).

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 loadmodulation an error code (see Annex E), 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 6.3.3.4.1.3), Interrogators shall 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 6.3.3.3.1.2.8).

After issuing a *BlockPermalock* 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 *BlockPermalock* and the tag's loadmodulated reply. The time T_{REPLY} shall be a multiple of T_1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 µs so that T_{REPLY} = n*1024/f_c +/- 32/f_c. 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 loadmodulate the reply shown inTable 81 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 0, within time T_1 in Table 41 — Link timing parameters, comprising a header (a 0-bit), the requested permalock bits, the tag's handle, and a CRC-16c calculated over the 0-bit, permalock bits, and handle. The tag's reply shall use the preamble specified by the TRext value in the *BeginRound* that initiated the round.
- **Read/Lock = 0 and the tag is unable to execute the command:** the tag shall loadmodulate an error code, within time T_1 in Table 41 — Link timing parameters; rather than the reply shown in Table 81 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 0 (see Annex E for error-code definitions and for the reply format). The tag's reply shall use the preamble specified by the TRext value in the *BeginRound* that initiated the round.
- **Read/Lock = 1 and The *BlockPermalock* succeeds:** After completing the *BlockPermalock* the tag shall loadmodulate the reply shown in Table 82 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 1comprising a header (a 0-bit), the tag's handle, and a CRC-16c calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockPermalock* completed

successfully The tag reply shall use the preamble as specified in the *BeginRound* command that initiated the round.

- **Read/Lock = 1 and the *BlockPermalock* does not succeed:** If the Interrogator does not observe a reply within 20 ms 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.*
- **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 82 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 1 (see Annex E for error-code definitions and for the reply format). The tag reply shall use the preamble as specified in the *BeginRound* command that initiated the round. The time T_{REPLY} shall be a multiple of T1 typical (see Table 41 — Link timing parameters) with a tolerance of +/- 2.4 µs so that $T_{REPLY} = n*1024/f_c +/- 32/f_c$.

Table 80 — *BlockPermalock* command

	Command	RFU	Read/ Lock	Mem Bank	BlockPtr Length	BlockPtr	BlockRange	Mask	RN	CRC-16c
# of bits	8	8	1	2	2	8, 16, 24 or 32	8	Variable	16	16
Description	11001001	00_h	0: Read 1: Perma-lock	00: RFU 01: UII 10: TID 11: User	Length of Wordpointer 00: 8 bit 01: 16 bit 10: 24 bit 11: 32 bit	Mask starting address, specified in units of 16 blocks	Mask range, specified in units of 16 blocks	0: Retain current permalock setting 1: Assert perma-lock	handle	

Table 81 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 0

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

Table 82 — Tag reply to a successful *BlockPermalock* command with Read/Lock = 1

	Header	RN	CRC-16c
# 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.

PJM Method: Refer to the PJM Method part of clauses 6.3.3.4.9 and 6.3.3.4.11. As for the *ACK-PJM*, see 6.3.3.4.11.2.4 and Table 57 — *ACK-PJM* command, this command can include multiple handles in the RN field in order to address multiple tags. Interrogators and tags shall operate (when using this command) as described above for ASK Method except that multiple tags can be addressed by this command and multiple tags can reply to this command.

7 Marking of equipment

All interrogators/readers (or the associated user manuals) shall be clearly and permanently marked stating with which National Regulations they comply.

All interrogators/readers (or the associated user manuals) shall be clearly permanently marked to show which Modes of ISO/IEC 18000 they support.

Annex A
(informative)

MODE 1: mandatory and optional commands required to support the ISO/IEC 15962 data protocol

ISO/IEC 15962 is an integral part of an RFID for item management system and defines the encoding rules for data for both open and closed system applications. To enable application data to be encoded onto the RFID tag and read from the RFID a number of the optional commands defined in ISO/IEC 15693-3 are required for an ISO/IEC 18000-3 Mode 1 tag to fulfil the functions necessary for RFID for item management. Table A.1 — ISO/IEC 15693 commands required to support the ISO/IEC 15962 data protocol defines these requirements and other conditions. The commands themselves remain exactly as specified in ISO/IEC 15693-3.

Table A.1 — ISO/IEC 15693 commands required to support the ISO/IEC 15962 data protocol

Command Code	Function	ISO/IEC 15693-3 Status	Status to support the ISO/IEC 15962 data protocol
01_h	Inventory	Mandatory	Required
02_h	Stay quiet	Mandatory	Required for the air interface
20_h	Read single block	Optional	One of the two read commands is required
23_h	Read multiple blocks	Optional	
21_h	Write single block	Optional	One of the two write commands is required
24_h	Write multiple blocks	Optional	
22_h	Lock block	Optional	Required
25_h	Select	Optional	Required for the air interface
26_h	Reset to ready	Optional	Required for the air interface
27_h	Write AFI	Optional	Required
28_h	Lock AFI	Optional	Desirable for full support
29_h	Write DSFID	Optional	Desirable for full support (see Note)
$2A_h$	Lock DSFID	Optional	Desirable for full support
$2B_h$	Get system information	Optional	Required
$2C_h$	Get multiple block security status	Optional	Required

Note 1: Support for the DSFID is a required part of ISO/IEC 15962. For ISO/IEC 18000-3 Mode 1 tags that do not support the Write DSFID air interface command, ISO/IEC 15962 defines a method for encoding this within user memory.

Annex B
(informative)

MODE 2 and MODE 3: Phase Jitter Modulation

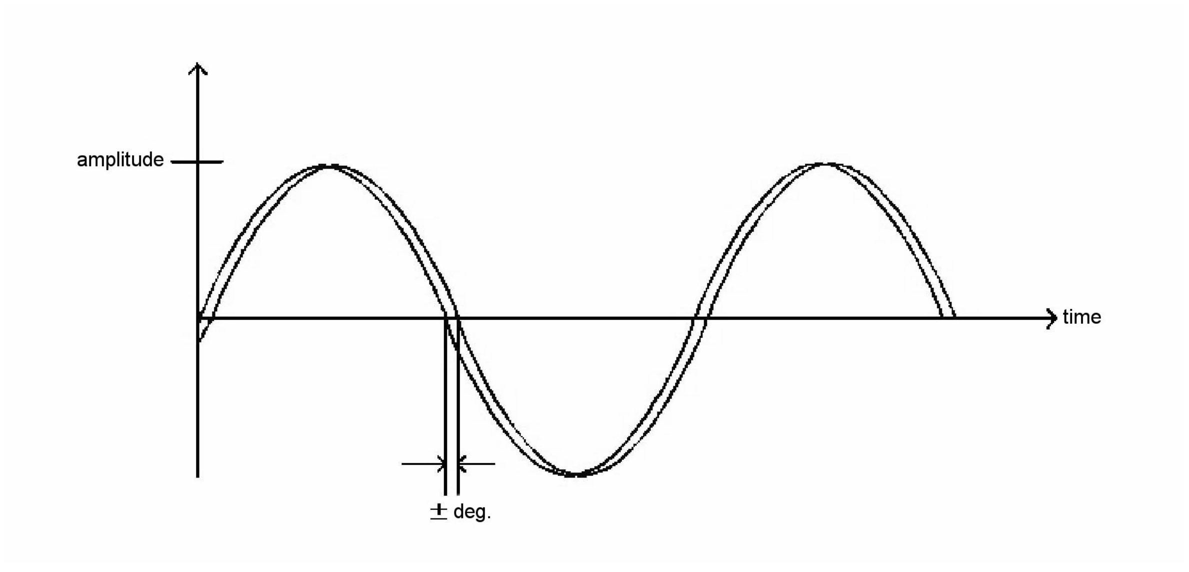


Figure B.1 — Phase jitter modulation

PJM consists of two components:

1) An in-phase (0°) powering signal I.
2) A low level quadrature (90°) data signal ±Q.

The PJM waveform is the sum of these two signals. In phasor notation, these can be represented as shown in Figure B.2 — Frequency spectrum.

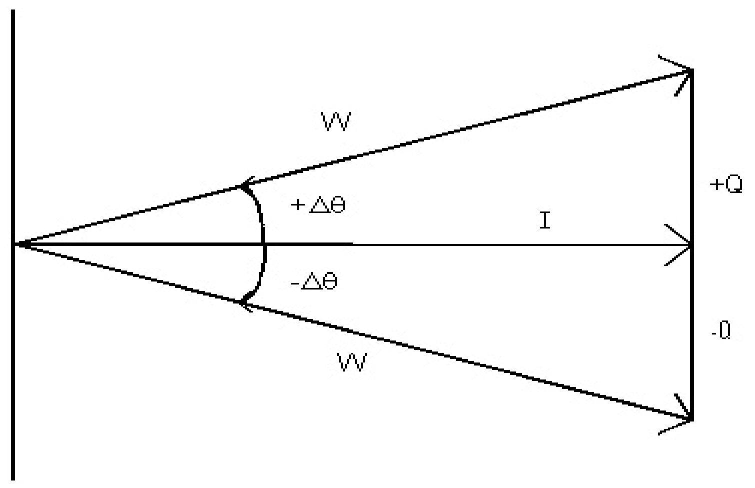


Figure B.2 — Frequency spectrum

The frequency spectrum of the phasor components are shown in Figure B.3 — Generation of PJM:

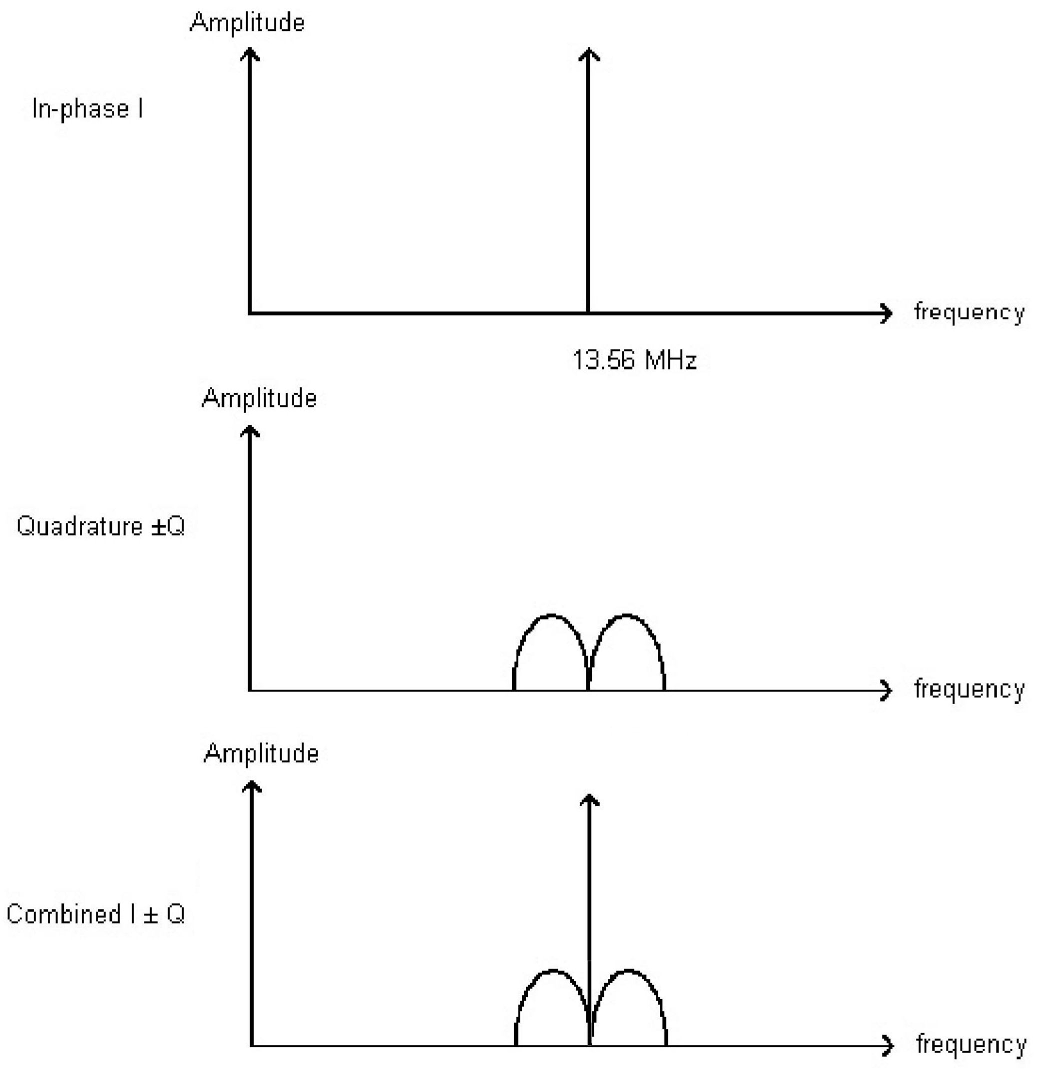


Figure B.3 — Generation of PJM

In MODE 2 of this part of ISO/IEC 18000:

- the interrogator command bit rate is 423,75 kbit/s, see M2-Int:9
- the phase change details are defined in M2-Int: 7 and 6.2.3.2.1

In MODE 3 of this part of ISO/IEC 18000:

- the interrogator command bit rate is 212 kbit/s, see M3-Int:9
- the phase change details are defined in M3-Int: 7 and 6.3.3.3.1.2.5

Features of PJM are:

- Constant amplitude signal with constant power transfer
- Sideband levels independent of data rate and can be adjusted to suit regulations
- Very high speed data can be transmitted because PJM bandwidth is no wider than the original doublesided data bandwidth
- Narrow bandwidth antennas do not limit high speed PJM signals. PJM can be pre-compensated to cancel for the effect of antenna bandwidth.

Example implementations of PJM are given in Figure B.4 — Example implementation - A simple circuit for providing a data controlled variable phase delay for generating PJM and Figure B.5 — Example implementation - circuit for the generation of PJM showing the various elements of a PJM signal:

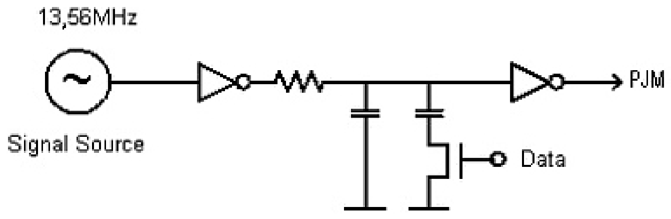


Figure B.4 — Example implementation - A simple circuit for providing a data controlled variable phase delay for generating PJM

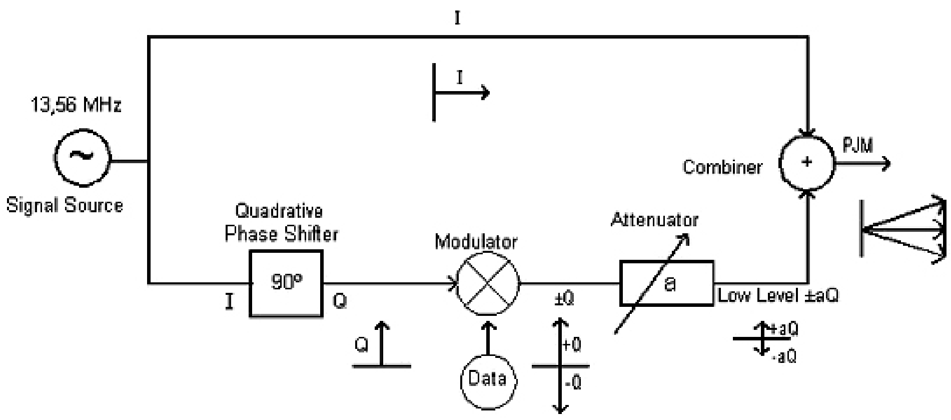


Figure B.5 — Example implementation - circuit for the generation of PJM showing the various elements of a PJM signal

Annex C
(normative)

MODE3: State transition tables

State-transition tables Table C.1 — Ready state-transition table to Table C.7 — Killed state-transition table shall define a tag response to interrogator commands. The term handle used in the state-transition tables is defined in 6.3.3.4.4; error codes are defined in Table E.2 — Tag error codes; "slot" is the slot-counter output shown in Figure 43 — Tag state diagram and detailed in Annex F "–" in the "Action" column means that a tag neither modifies its **SL** or **inventoried** flags nor loadmodulates a reply.

C.1 Present state: Ready

Table C.1 — Ready state-transition table

Command	Condition	Action	Next State
BeginRound [1]	slot=0; matching **inventoried** & **SL** flags	loadmodulate StoredCRC	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**ready**
NextSlot	All	–	**ready**
ResizeRound	All	–	**ready**
ACK	All	–	**ready**
NAK	All	–	**ready**
Req_RN	All	–	**ready**
Select	All	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	All	–	**ready**
Write	All	–	**ready**
Kill	All	–	**ready**
Lock	All	–	**ready**
Access	All	–	**ready**
BlockWrite	All	–	**ready**
BlockErase	All	–	**ready**
BlockPermalock	All	–	**ready**
Invalid [2]	All	–	**ready**

Note 1: *BeginRound* starts a new round and may change the session. *BeginRound* also instructs a tag to load a new random value into its slot counter.

Note 2: "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.2 Present state: Arbitrate

Table C.2 — Arbitrate state-transition table

Command	Condition	Action	Next State
BeginRound [1,2]	slot=0; matching **inventoried** & **SL** flags	loadmodulate StoredCRC	**Reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	Otherwise	–	**Ready**
NextSlot	slot=0 after decrementing slot counter	loadmodulate StoredCRC	**Reply**
	slot<>0after decrementing slot counter	–	**arbitrate**
ResizeRound [2]	slot=0	loadmodulate StoredCRC	**Reply**
	slot<>0	–	**arbitrate**
ACK	All	–	**arbitrate**
NAK	All	–	**arbitrate**
Req_RN	All	–	**arbitrate**
Select	All	assert or deassert **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**
Invalid [3]	all	–	**arbitrate**

Note 1: *BeginRound* starts a new round and may change the session.
Note 2: *BeginRound* and *ResizeRound* instruct a tag to load a new random value into its slot counter.
Note 3: "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 *BeginRound*) 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.

C.3 Present state: Reply

Table C.3 — Reply state-transition table

Command	Condition	Action	Next State
BeginRound [1,2]	slot=0; matching **inventoried** & **SL** flags	loadmodulate StoredCRC	**Reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**Arbitrate**
	otherwise	–	**Ready**
NextSlot	all	–	**Arbitrate**
ResizeRound [2]	slot=0	loadmodulate StoredCRC	**Reply**
	slot<>0	–	**Arbitrate**
ACK	valid StoredCRC	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**Acknowledged**
	invalid StoredCRC	–	**Arbitrate**
NAK	all	–	**Arbitrate**
Req_RN	all	–	**Arbitrate**
Select	all	assert or deassert **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 40 — Link timing – Both Modes and Table 41 — Link timing parameters	–	**Arbitrate**
Invalid [3]	all	–	**Reply**

Note 1: *BeginRound* starts a new round and may change the session.
Note 2: *BeginRound* and *ResizeRound* instruct a tag to load a new random value into its slot counter.
Note 3: “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 *BeginRound*) 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.

C.4 Present state: Acknowledged

Table C.4 — Acknowledged state-transition table

Command	Condition	Action	Next State
BeginRound [1]	slot=0; matching **inventoried** [2] & **SL** flags	loadmodulate StoredCRC; transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried** [2] & **SL** flags	transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* if and only if new session matches prior session	**ready**
NextSlot	all	transition **inventoried** from *A*→*B*	**ready**
ResizeRound	all	transition **inventoried** from *A*→*B*	**ready**
ACK	valid StoredCRC	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**acknowledged**
	invalid StoredCRC	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid StoredCRC & access password<>0	loadmodulate handle	**open**
	valid StoredCRC & access password=0	loadmodulate handle	**secured**
	invalid StoredCRC	–	**acknowledged**
Select	all	assert or deassert **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 40 — Link timing – Both Modes and Table 41 — Link timing parameters	–	**arbitrate**
Invalid [3]	all	–	**acknowledged**

Note 1: *BeginRound* starts a new round and may change the session. *BeginRound* also instructs a tag to load a new random value into its slot counter.

Note 2: As described in 6.3.3.4.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

Note 3: “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 *BeginRound*) 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.

C.5 Present state: Open

Table C.5 — Open state-transition table

Command	Condition	Action	Next State
BeginRound [1]	slot=0; matching **inventoried**[2] & **SL** flags	loadmodulate StoredCRC; transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**Reply**
	slot<>0; matching **inventoried**[2] & **SL** flags	transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**Arbitrate**
	otherwise	transition **inventoried** from *A*→*B* if and only if new session matches prior session	**Ready**
NextSlot	all	transition **inventoried** from *A*→*B*	**Ready**
ResizeRound	all	transition **inventoried** from *A*→*B*	**Ready**
ACK	valid handle	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**Open**
	invalid handle	–	**Arbitrate**
NAK	all	–	**Arbitrate**
Req_RN	valid handle	loadmodulate new RN16	**Open**
	invalid handle	–	**Open**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**Ready**
Read	valid handle & valid memory access	loadmodulate data and handle	**Open**
	valid handle & invalid memory access	loadmodulate error code	**Open**
	invalid handle	–	**Open**
Write	valid handle & valid memory access	loadmodulate handle when done	**Open**
	valid handle & invalid memory access	loadmodulate error code	**Open**
	invalid handle	–	**Open**
Kill [3] (see also 6.3.3.4.11.3.4)	valid handle & valid nonzero kill password & Recom = 0	loadmodulate handle when done	**Killed**
	valid handle & valid nonzero kill password & Recom <> 0	loadmodulate handle when done	**Open**
	valid handle & invalid nonzero kill password	–	**Arbitrate**
	valid handle & kill password=0	loadmodulate error code	**Open**
	invalid handle	–	**Open**
Lock	all	–	**Open**
Access (see also Figure 49 — *Access* procedure*)*	valid handle & valid access password	loadmodulate handle	**Secured**
	valid handle & invalid access password	–	**Arbitrate**
	invalid handle	–	**Open**
BlockWrite	valid handle & valid memory access	loadmodulate handle when done	**Open**
	valid handle & invalid memory access	loadmodulate error code	**Open**
	invalid handle	–	**Open**
BlockErase	valid handle & valid memory access	loadmodulate handle when done	**Open**
	valid handle & invalid memory access	loadmodulate error code	**Open**
	invalid handle	–	**Open**
BlockPermalock	all	–	**Open**

Command	Condition	Action	Next State
Invalid [4]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**open**
	otherwise valid commands, except Req_RN or BeginRound, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 47 — Kill procedure and Figure 49 — Access procedure).	–	**Arbitrate**

Note 1: *BeginRound* starts a new round and may change the session. *BeginRound* also instructs a tag to load a new random value into its slot counter.
Note 2: As described in 6.3.3.4.8, a tag transitions its **inventoried** flag prior to evaluating the condition.
Note 3: As described in 6.3.3.4.11.3.4, if a tag does not implement Recom bits LSB and 2SB it has to ignore their value and treat them as if their value were zero.
Note 4: "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 *BeginRound*) 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.

C.6 Present state: Secured

Table C.6 — Secured state-transition table

Command	Condition	Action	Next State
BeginRound [1]	slot=0; matching **inventoried** [2] & **SL** flags	loadmodulate StoredCRC; transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**Reply**
	slot<>0; matching **inventoried** [2] & **SL** flags	transition **inventoried**[2] from *A*→*B* if and only if new session matches prior session	**Arbitrate**
	otherwise	transition **inventoried** from *A*→*B* if and only if new session matches prior session	**Ready**
NextSlot	all	transition **inventoried** from *A*→*B*	**Ready**
ResizeRound	all	transition **inventoried** from *A*→*B*	**Ready**
ACK	valid handle	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**Secured**
	invalid handle	–	**Arbitrate**
NAK	all	–	**Arbitrate**
Req_RN	valid handle	loadmodulate new RN16	**Secured**
	invalid handle	–	**Secured**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**Ready**
Read	valid handle & valid memory access	loadmodulate data and handle	**Secured**
	valid handle & invalid memory access	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
Write	valid handle & valid memory access	loadmodulate handle when done	**Secured**
	valid handle & invalid memory access	loadmodulate error code	**Secured**

Command	Condition	Action	Next State
	invalid handle	–	**Secured**
Kill[3] (see also 6.3.3.4.11.3.4)	valid handle & valid nonzero kill password & Recom = 0	loadmodulate handle when done	**Killed**
	valid handle & valid nonzero kill password & Recom <> 0	loadmodulate handle when done	**Secured**
	valid handle & invalid nonzero kill password	–	**Arbitrate**
	valid handle & kill password=0	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
Lock	valid handle & valid lock payload	loadmodulate handle when done	**Secured**
	valid handle & invalid lock payload	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
Access (see also Figure 49 — Access procedure)	valid handle & valid access password	loadmodulate handle	**Secured**
	valid handle & invalid access password	–	**Arbitrate**
	invalid handle	–	**Secured**
BlockWrite	valid handle & valid memory access	loadmodulate handle when done	**Secured**
	valid handle & invalid memory access	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
BlockErase	valid handle & valid memory access	loadmodulate handle when done	**Secured**
	valid handle & invalid memory access	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
BlockPermalock	valid handle, valid payload, & Read/Lock = 0	loadmodulate permalock bits and handle	**Secured**
	valid handle, invalid payload, & Read/Lock = 0	loadmodulate error code	**Secured**
	valid handle, valid payload, & Read/Lock = 1	loadmodulate handle when done	**Secured**
	valid handle, invalid payload, & Read/Lock = 1	loadmodulate error code	**Secured**
	invalid handle	–	**Secured**
Invalid[4]	all, excluding valid commands interspersed between successive Kill or Access commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**Secured**
	otherwise valid commands, except Req_RN or BeginRound, interspersed between successive Kill or Access commands in a kill or access sequence, respectively(see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**Arbitrate**

Note 1: *BeginRound* starts a new round and may change the session. *BeginRound* also instructs a tag to load a new random value into its slot counter.

Note 2: As described in 6.3.3.4.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

Note 3: As described in 6.3.3.4.11.3.4, if a tag does not implement Recom bits LSB and 2SB it has to ignore their value and treat them as if their value were zero.

Note 4: “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 *BeginRound*) 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.

C.7 Present state: Killed (optional)

Table C.7 — Killed state-transition table

Command	Condition	Action	Next State
BeginRound	all	–	**killed**
NextSlot	all	–	**killed**
ResizeRound	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[1]	all	–	**killed**

Note 1: "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 D
(normative)

MODE 3: Command Response Tables

Table D.1 — Power-up command-response table to Table D.17 — T2 timeout command-response table shall define a tag response to interrogator commands. The term "handle" used in the state-transition tables is defined in 6.3.3.4.4; error codes are defined in Table E.2 — Tag error codes; "slot" is the slot counter output shown in Figure 43 — Tag state diagram and detailed in Annex F; "–" in the "Response" column means that a tag neither modifies its **SL** or **inventoried** flags nor loadmodulates a reply.

D.1 Command response: Power-up

Table D.1 — Power-up command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	power-up	–	**ready**
killed	all	–	**killed**

D.2 Command response: *BeginRound*

Table D.2 — *BeginRound* [1] command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply	slot=0; matching **inventoried** & **SL flags**	loadmodulate StoredCRC	reply
	slot<>0; matching **inventoried** & **SL flags**	–	arbitrate
	otherwise	–	ready
acknowledged, open, secured	slot=0; matching **inventoried** [2] & **SL** flags	loadmodulate StoredCRC; transition **inventoried**[2] from $A \rightarrow B$ if and only if new session matches prior session	reply
	slot<>0; matching **inventoried** [2] & **SL** flags	transition **inventoried**[2] from $A \rightarrow B$ if and only if new session matches prior session	arbitrate
	otherwise	transition **inventoried** from $A \rightarrow B$ if and only if new session matches prior session	ready
killed	all	–	killed

Note 1: *BeginRound* (in any other state than **killed**) starts a new round and may change the session. *BeginRound* also instructs a tag to load a new random value into its slot counter.

Note 2: As described in 6.3.3.4.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

D.3 Command response: *NextSlot*

Table D.3 — *NextSlot* command-response table [1]

Starting State	Condition	Response	Next State
ready	All	–	**ready**
arbitrate	slot<>0 after decrementing slot counter	–	**arbitrate**
	slot=0 after decrementing slot counter	loadmodulate StoredCRC	**reply**
reply	All	–	**arbitrate**
acknowledged, open, secured	All	transition **inventoried** from *A→B*	**ready**
killed	all	–	**killed**

Note 1: See Table D.17 — T2 timeout command-response table for the tag response to a *NextSlot* whose session parameter does not match that of the current inventory round.

D.4 Command response: *ResizeRound*

Table D.4 — *ResizeRound* [1] command-response table [2]

Starting State	Condition	Response	Next State
ready	All	–	**ready**
arbitrate, reply	slot<>0	–	**arbitrate**
	slot=0	loadmodulate StoredCRC	**reply**
acknowledged, open, secured	All	transition **inventoried** from *A→B*	**ready**
killed	All	–	**killed**

Note 1: *ResizeRound,* in the **arbitrate** or **reply** states, instructs a tag to load a new random value into its slot counter.

Note 2: See Table Table D.17 — T2 timeout command-response table for the tag response to a *ResizeRound* whose session parameter does not match that of the current inventory round.

D.5 Command response: *ACK*

Table D.5 — *ACK* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	all	–	**arbitrate**
reply	valid StoredCRC	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**acknowledged**
	invalid StoredCRC	–	**arbitrate**
acknowledged	valid StoredCRC	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**acknowledged**
	invalid StoredCRC	–	**arbitrate**
open	valid handle	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**open**
	invalid handle	–	**arbitrate**
secured	valid handle	see Table 42 — Tag data and, if required, PacketCRC loadmodulated in response to an ACK command	**secured**
	invalid handle	–	**arbitrate**
killed	all	–	**killed**

D.6 Command response: *NAK*

Table D.6 — *NAK* command-response table

Starting State	Condition	Response	Next State
ready	All	–	**ready**
arbitrate, reply, acknowledged, open, secured	All	–	**arbitrate**
killed	All	–	**killed**

D.7 Command response: *Req_RN*

Table D.7 — *Req_RN* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply	all	–	**arbitrate**
acknowledged	valid StoredCRC & access password<>0	loadmodulate handle	**open**
	valid StoredCRC & access password=0	loadmodulate handle	**secured**
	invalid StoredCRC	–	**acknowledged**
open	valid handle	loadmodulate new RN16	**open**
	invalid handle	–	**open**
secured	valid handle	loadmodulate new RN16	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

D.8 Command response: *Select*

Table D.8 — *Select* command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	All	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
killed	All	–	**killed**

D.9 Command response: *Read*

Table D.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	loadmodulate error code	**open**
	valid handle & valid memory access	loadmodulate data and handle	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	loadmodulate error code	**secured**
	valid handle & valid memory access	loadmodulate data and handle	**secured**
	invalid handle	–	**secured**
killed	All	–	**killed**

D.10 Command response: *Write*

Table D.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	loadmodulate error code	**open**
	valid handle & valid memory access	loadmodulate handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	loadmodulate error code	**secured**
	valid handle & valid memory access	loadmodulate handle when done	**secured**
	invalid handle	–	**secured**
killed	All	–	**killed**

D.11 Command response: *Kill*

Table D.11 — *Kill*[1] command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open[2]	valid handle & kill password=0	loadmodulate error code	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password & Recom=0	loadmodulate handle when done	**killed**
	valid handle & valid nonzero kill password & Recom<>0	loadmodulate handle when done	**open**
	invalid handle	–	**open**
secured[2]	valid handle & kill password=0	loadmodulate error code	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password & Recom=0	loadmodulate handle when done	**killed**
	valid handle & valid nonzero kill password & Recom<>0	loadmodulate handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

Note 1: See also Figure 47 — *Kill* procedure.

Note 2: As described in 6.3.3.4.11.3.4, if a tag does not implement Recom bits LSB and 2SB it has to ignore their value and treat them as if their value were zero.

D.12 Command response: *Lock*

Table D.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	loadmodulate error code	**secured**
	valid handle & valid lock payload	loadmodulate handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

D.13 Command response: *Access*

Table D.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	loadmodulate handle	**secured**
	invalid handle	–	**open**
secured	valid handle & invalid access password	–	**arbitrate**
	valid handle & valid access password	loadmodulate handle	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

1: See also Figure 49 — Access procedure.

D.14 Command response: *BlockWrite*

Table D.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	loadmodulate error code	**open**
	valid handle & valid memory access	loadmodulate handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	loadmodulate error code	**secured**
	valid handle & valid memory access	loadmodulate handle when done	**secured**
	invalid handle	–	**secured**
killed	All	–	**killed**

D.15 Command response: *BlockErase*

Table D.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	loadmodulate error code	**open**
	valid handle & valid memory access	loadmodulate handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	loadmodulate error code	**secured**
	valid handle & valid memory access	loadmodulate handle when done	**secured**
	invalid handle	–	**secured**
killed	All	–	**killed**

D.16 Command response: *BlockPermalock*

Table D.16 — *BlockPermalock* command and 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	loadmodulate permalock bits and handle	**secured**
	valid handle, invalid payload, & **Read/Lock** = 0	loadmodulate error code	**secured**
	valid handle, valid payload, & **Read/Lock** = 1	loadmodulate handle when done	**secured**
	valid handle, invalid payload, & **Read/Lock** = 1	loadmodulate error code	**secured**
	invalid handle	–	**secured**
killed	All	–	**killed**

D.17 Command response: T_2 timeout

Table D.17 — T_2 timeout command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	all	–	**arbitrate**
reply, acknowledged	See Figure 40 — Link timing – Both Modes and Table 41 — Link timing parameters	–	**arbitrate**
open	all	–	**open**
secured	all	–	**secured**
killed	all	–	**killed**

D.18 Command response: Invalid command

Table D.18 — Invalid command-response table

Starting State	Condition	Response	Next State
ready[1]	All	–	**ready**
arbitrate[2]	All	–	**arbitrate**
reply[2]	All	–	**reply**
acknowledged[2]	All	–	**acknowledged**
open[2]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**open**
	otherwise valid commands, except *Req_RN* or *BeginRound*, interspersed between successive Kill or Access commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**arbitrate**
secured[2]	all, excluding valid commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**secured**
	otherwise valid commands, except *Req_RN* or *BeginRound*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see 6.3.3.4.11.3.4 and 6.3.3.4.11.3.6).	–	**arbitrate**
killed[1]	All	–	**killed**

1: "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.

2: "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 *BeginRound)* 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.

Annex E
(normative)

MODE 3: Error codes

If a tag encounters an error when executing an access command that reads from or writes to memory, and if the command is a handle-based command (i.e. *Read*, *Write*, *Kill*, *Lock*, *BlockWrite*, or *BlockErase*, or *BlockPermalock*), then the tag shall loadmodulate an error code as shown in Table E.1 — Tag-error reply format instead of its normal reply.

- If the tag supports error-specific codes, it shall use the error-specific codes shown in Table E.2 — Tag error codes.
- If the tag does not support error-specific codes, it shall loadmodulate error code 00001111_2 (indicating a non-specific error) as shown in Table E.2 — Tag error codes.
- Tags shall loadmodulate error codes only from the **open** or **secured** states.
- A tag shall not loadmodulate an error code if it receives an invalid access command; instead, it shall ignore the command.
- If an error is described by more than one error code, the more specific error code shall take precedence and shall be the code that the tag loadmodulates.
- The header for an error code is a 1-bit, unlike the header for a normal tag response, which is a 0-bit.

Table E.1 — Tag-error reply format

	Header	Error Code	RN	CRC-16c
# of bits	1	8	16	16
description	1	Error code	handle	

Table E.2 — Tag error codes

Error-Code Support	Error Code	Error-Code Name	Error Description
Error-specific	00000000_2	Other error	"Catch-all" for errors not covered by other codes
	00000011_2	Memory overrun	The specified memory location does not exist or the UII length field is not supported by the tag
	00000100_2	Memory locked	The specified memory location is locked and/or permalocked and is either not writeable or not readable
	00001011_2	Insufficient power	The tag has insufficient power to perform the memory-write operation
Non-specific	00001111_2	Non-specific error	The tag does not support error-specific codes

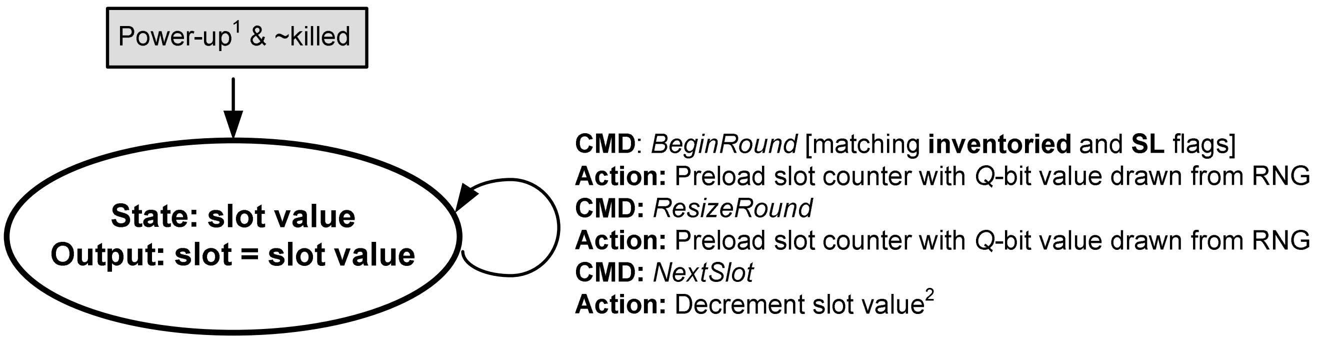
Annex F
(normative)

MODE 3: Slot counter

F.1 Slot-counter operation

As described in 6.3.3.4.4.8, tags implement a 15-bit slot counter. As described in 6.3.3.4.9, interrogators use the slot counter to regulate the probability of a tag responding to a *BeginRound*, *ResizeRound*, or *NextSlot* command. Upon receiving a *BeginRound* or *ResizeRound* a tag preloads a *Q*-bit value, drawn from the tag RNG (see 6.3.3.4.4.8), into its slot counter. *Q* is an integer in the range (0,15). A *BeginRound* specifies *Q*; a *ResizeRound* may modify *Q* from the prior *BeginRound*. Upon receiving a *NextSlot*, a tag decrements its slot value. Tags transition to the **reply** state when their slot value reaches 0000_h. The slot counter implements continuous counting, meaning that, after the slot value decrements to 0000_h, the next *NextSlot* causes it to roll over and begin counting down from $7FFF_h$. Tags that return to **arbitrate** (for example, from the **reply** state) with a slot value of 0000_h decrement their slot value from 0000_h to $7FFF_h$ at the next *NextSlot* (with matching session) and, because their slot value is now non-zero, remain in **arbitrate**.

Annex C and Annex D contain tables describing a tag response to interrogator commands; "slot" is a parameter in these tables.


Power-up[1] & ~killed
State: slot value
Output: slot = slot value
CMD: BeginRound [matching inventoried and SL flags]
Action: Preload slot counter with Q-bit value drawn from RNG
CMD: ResizeRound
Action: Preload slot counter with Q-bit value drawn from RNG
CMD: NextSlot
Action: Decrement slot value[2]

Notes
1. The slot counter may assume an arbitrary value at Tag power-up
2. If slot value = 0000_h, then decrementing the slot value causes it to roll over to $7FFF_h$

Figure F.1 — Slot-counter state diagram

Annex G
(informative)

MODE 3: Example slot-count (Q) selection algorithm

G.1 Example algorithm an interrogator might use to choose *Q*

Figure G.1 — Example algorithm for choosing the slot count parameter *Q* shows an algorithm an interrogator might use for setting the slot-count parameter *Q* in a *BeginRound* 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 *BeginRound.* 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.

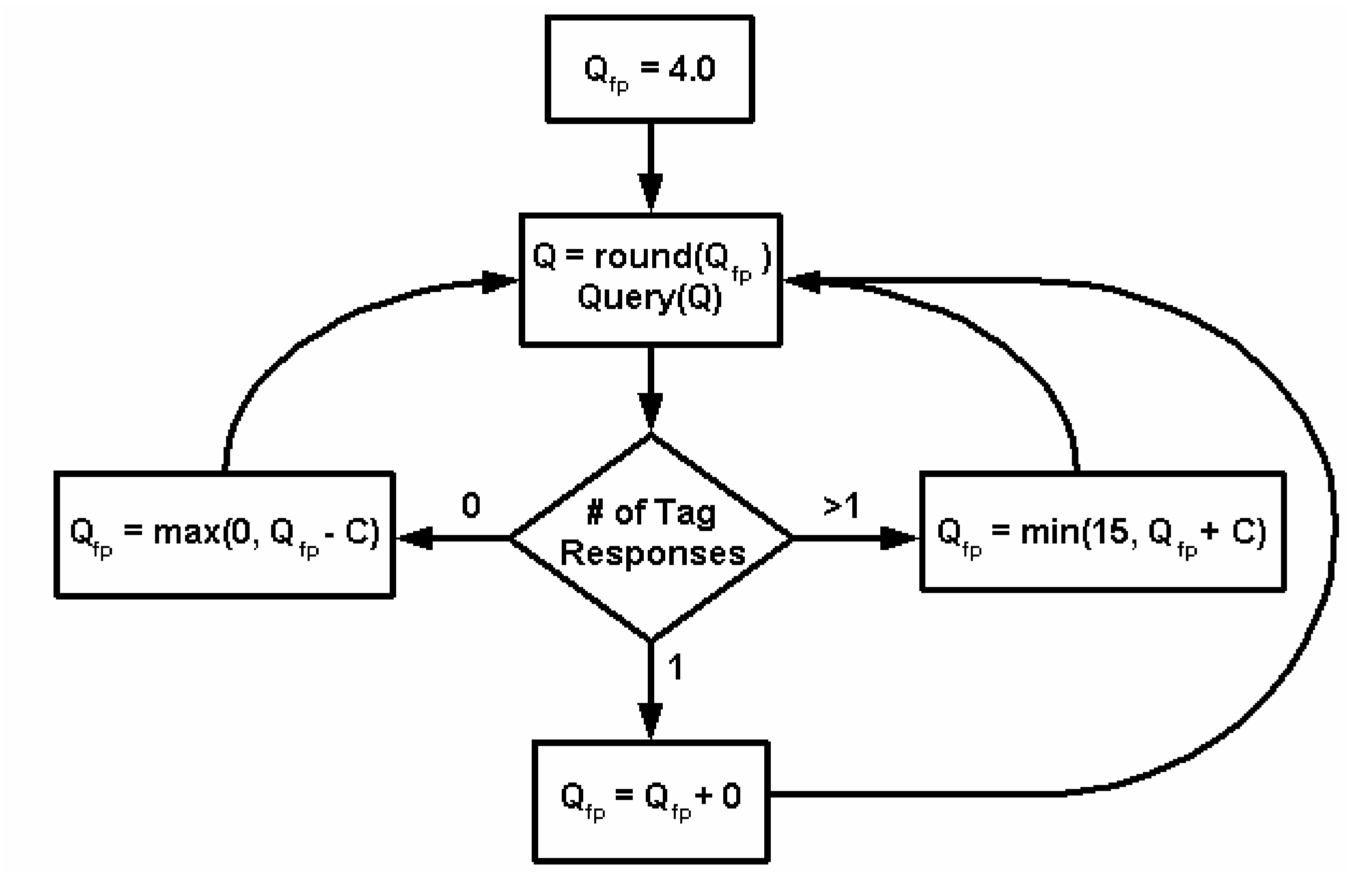


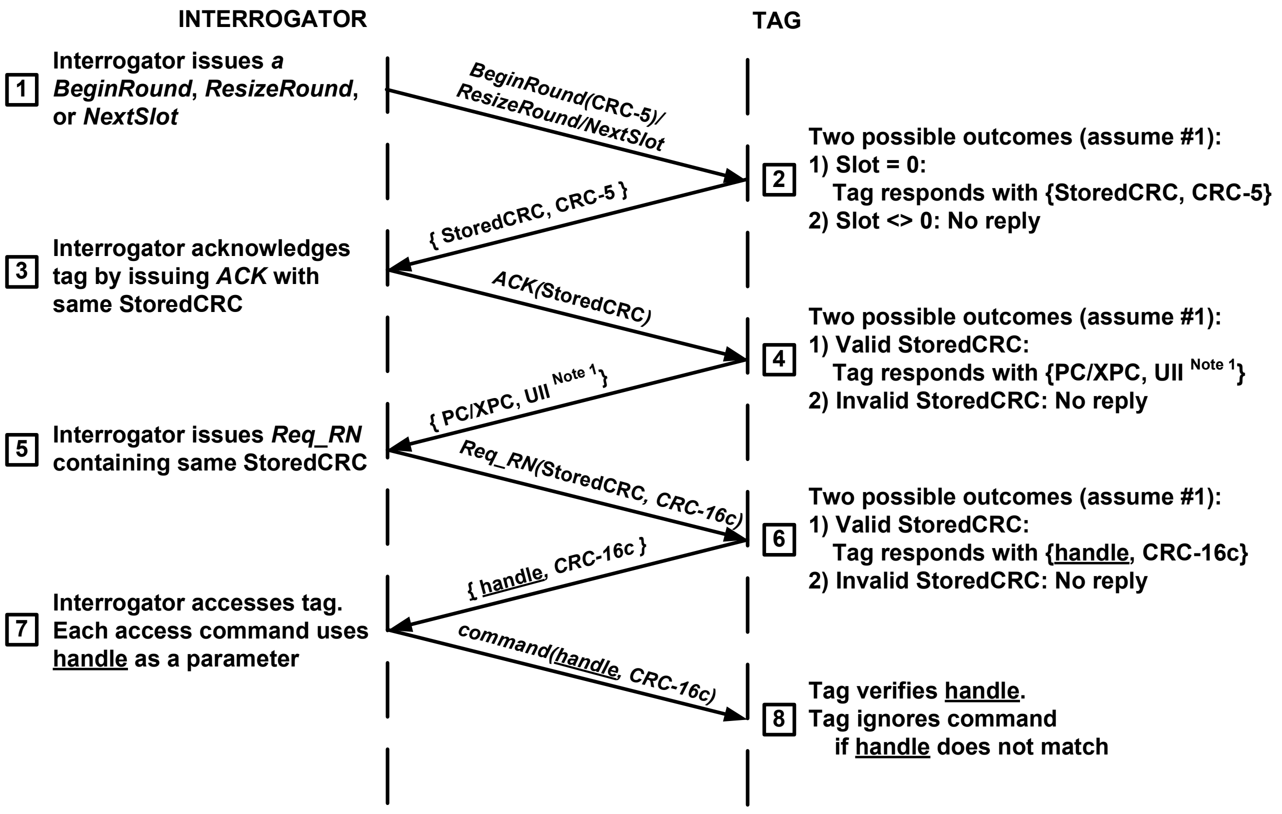
Figure G.1 — Example algorithm for choosing the slot count parameter *Q*

Annex H
(informative)

MODE 3: Example of tag inventory and access

H.1 ASK Method: Example inventory and access of a single tag.

Figure H.1 — ASK Method: Example of tag inventory and access shows the steps by which an interrogator inventories and accesses a single tag for ASK Method.

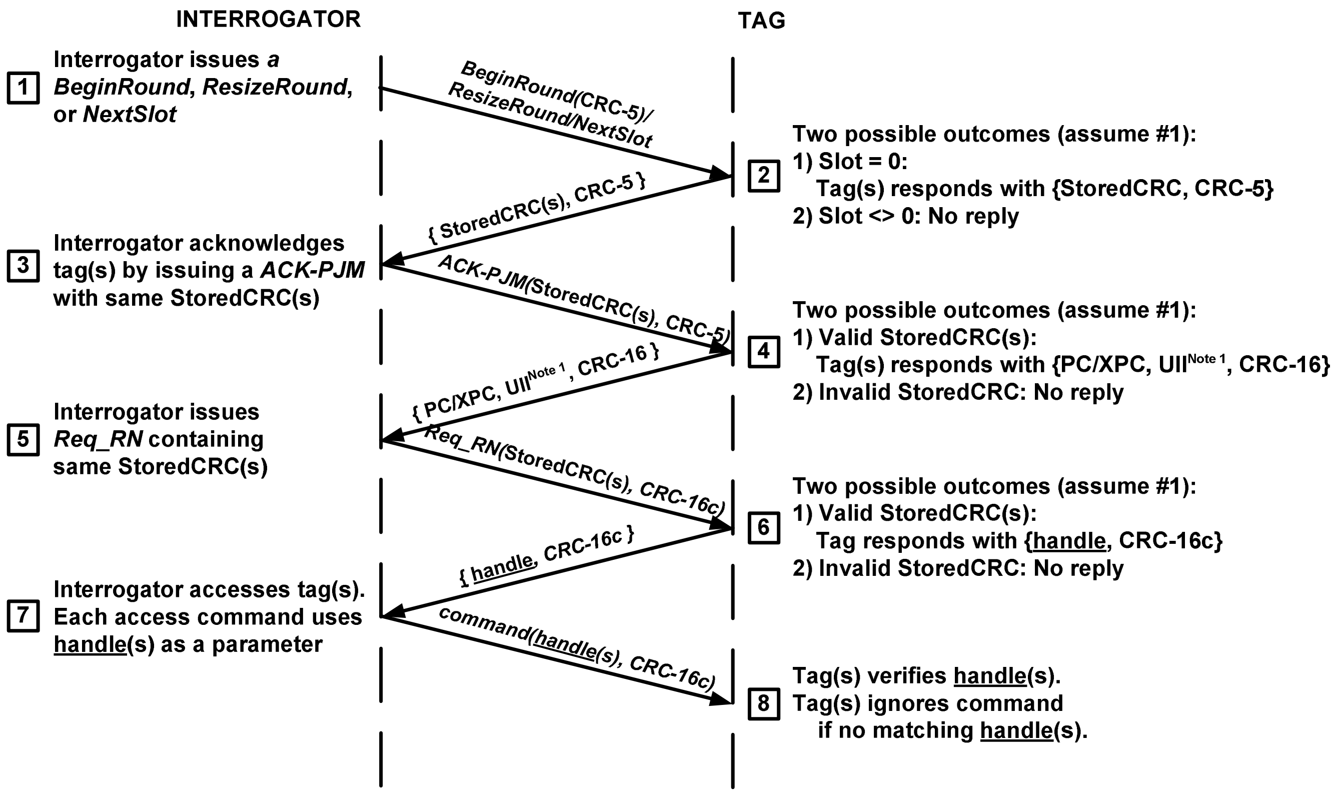

INTERROGATOR
TAG
1 Interrogator issues *a* *BeginRound*, *ResizeRound*, or *NextSlot*
BeginRound(CRC-5)/ ResizeRound/NextSlot
2 Two possible outcomes (assume #1):
1) Slot = 0:
Tag responds with {StoredCRC, CRC-5}
2) Slot <> 0: No reply
{ StoredCRC, CRC-5 }
3 Interrogator acknowledges tag by issuing *ACK* with same StoredCRC
ACK(StoredCRC)
4 Two possible outcomes (assume #1):
1) Valid StoredCRC:
Tag responds with {PC/XPC, UII Note 1}
2) Invalid StoredCRC: No reply
{ PC/XPC, UII Note 1}
5 Interrogator issues *Req_RN* containing same StoredCRC
Req_RN(StoredCRC, CRC-16c)
6 Two possible outcomes (assume #1):
1) Valid StoredCRC:
Tag responds with {handle, CRC-16c}
2) Invalid StoredCRC: No reply
{ handle, CRC-16c }
7 Interrogator accesses tag. Each access command uses handle as a parameter
command(handle, CRC-16c)
8 Tag verifies handle.
Tag ignores command if handle does not match

Note 1: In certain cases a PacketCRC should be added if required. See table 42 for the definition of the cases.

Figure H.1 — ASK Method: Example of tag inventory and access

H.2 PJM Method: Example inventory and access of a single or multiple tags

Figure H.2 — PJM Method: Example of tag inventory and access shows the steps by which an Interrogator inventories and accesses a single or multiple tags for PJM Method.


INTERROGATOR
TAG
1 Interrogator issues a BeginRound, ResizeRound, or NextSlot
BeginRound(CRC-5)/ ResizeRound/NextSlot
2 Two possible outcomes (assume #1):
1) Slot = 0:
Tag(s) responds with {StoredCRC, CRC-5}
2) Slot <> 0: No reply
{ StoredCRC(s), CRC-5 }
3 Interrogator acknowledges tag(s) by issuing a ACK-PJM with same StoredCRC(s)
ACK-PJM(StoredCRC(s), CRC-5)
4 Two possible outcomes (assume #1):
1) Valid StoredCRC(s):
Tag(s) responds with {PC/XPC, UII Note 1, CRC-16}
2) Invalid StoredCRC: No reply
{ PC/XPC, UII Note 1, CRC-16 }
5 Interrogator issues Req_RN containing same StoredCRC(s)
Req_RN(StoredCRC(s), CRC-16c)
6 Two possible outcomes (assume #1):
1) Valid StoredCRC(s):
Tag responds with {handle, CRC-16c}
2) Invalid StoredCRC: No reply
{ handle, CRC-16c }
7 Interrogator accesses tag(s). Each access command uses handle(s) as a parameter
command(handle(s), CRC-16c)
8 Tag(s) verifies handle(s).
Tag(s) ignores command if no matching handle(s).

Note 1: In certain cases a PacketCRC should be added if required. See table 42 for the definition of the cases.

Figure H.2 — PJM Method: Example of tag inventory and access

Annex I
(informative)

MODE 3: Calculation of 5-bit and 16-bit cyclic redundancy checks

I.1 Example CRC-5 encoder/decoder

An exemplary schematic diagram for a CRC-5 encoder/decoder is shown in Figure I.1 — Example CRC-5 circuit, using the polynomial and preset defined in Figure 40 — Link timing – Both Modes.

To encode a CRC-5, first preload the entire CRC register (i.e. C[4:0]) with 01001_2, then clock the data bits to be encoded into the input labeled DATA, MSB first. After clocking in all the data bits, C[4:0] holds the CRC-5 value.

To decode a CRC-5, first preload the entire CRC register (C[4:0]) with 01001_2, then clock the received data and CRC-5 {data, CRC-5} bits into the input labeled DATA, MSB first. The CRC-5 check passes if C[4:0] = 00000_2.

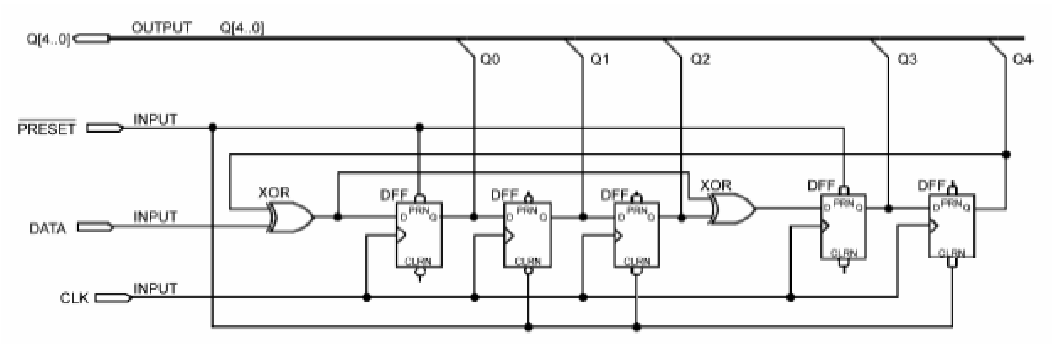


Figure I.1 — Example CRC-5 circuit

Table I.1 — CRC-5 register preload values

Register	Preload value
Q0	1
Q1	0
Q2	0
Q3	1
Q4	0

I.2 Example CRC-16 calculations

This example shows the CRC-16 that a tag would compute at power-up. As shown in Figure 41 — Logical memory map, UII memory contains a StoredCRC starting at address 00_h, a Stored PC starting at address 10_h, zero or more UII words starting at address 20_h, an XPC_W1 starting at address 210_h and an optional XPC_W2 starting at address 220_h As described in 6.3.3.4.1.2.1 a tag calculates its StoredCRC over its StoredPC and UII but omits the XPC_W1 and XPC_W2 from the calculation. Figure 40 — Link timing – Both Modes shows the CRC-16 that a tag would calculate and logically map into UII memory at power-up, for the indicated example StoredPC and UII word values. In each successive column, one more word of UII memory is written, with the entire UII memory written in the rightmost column. The indicated StoredPC values correspond to the number of UII words written, with StoredPC bits 15_h–$1F_h$ set to zero. Entries marked N/A mean that that word of UII memory is not included as part of the CRC calculation.

Table I.2 — UII memory contents for an example tag

UII word starting address	UII word contents	UII wordvalues						
00_h	CRC-16	$E2F0_h$	$CCAE_h$	$968F_h$	$78F6_h$	$C241_h$	$2A91_h$	1835_h
10_h	PC	0000_h	0800_h	1000_h	1800_h	2000_h	2800_h	3000_h
20_h	UII word 1	N/A	1111_h	1111	1111_h	1111_h	1111_h	1111_h
30_h	UII word 2	N/A	N/A	2222_h	2222_h	2222_h	2222_h	2222_h
40_h	UII word 3	N/A	N/A	N/A	3333_h	3333_h	3333_h	3333_h
50_h	UII word 4	N/A	N/A	N/A	N/A	4444_h	4444_h	4444_h
60_h	UII word 5	N/A	N/A	N/A	N/A	N/A	5555_h	5555_h
70_h	UII word 6	N/A	N/A	N/A	N/A	N/A	N/A	6666_h

I.3 Example CRC-16c encoder/decoder

An exemplary schematic diagram for a CRC-16c encoder/decoder is shown in Figure I.2 — Example CRC-16c circuit, using the polynomial and preset defined in Figure 40 — Link timing – Both Modes. (the polynomial used to calculate the CRC-16c, $x^{16} + x^{12} + x^5 + 1$, is the CRC-CCITT International Standard, ITU Recommendation X.25). This is applicable for both CRC-16 and CRC-16c calculations.

To encode a CRC-16c, first preload the entire CRC register (i.e. C[15:0]) with $FFFF_h$, then clock the data bits to be encoded into the input labeled DATA, MSB first. After clocking in all the data bits, C[15:0] holds the ones complement of the CRC-16c value.

To decode a CRC-16c, first preload the entire CRC register (C[15:0]) with $FFFF_h$, then clock the received data and CRC-16c {data, CRC-16} bits into the input labeled DATA, MSB first. The CRC-16c check passes if C[15:0] $=1D0F_h$.

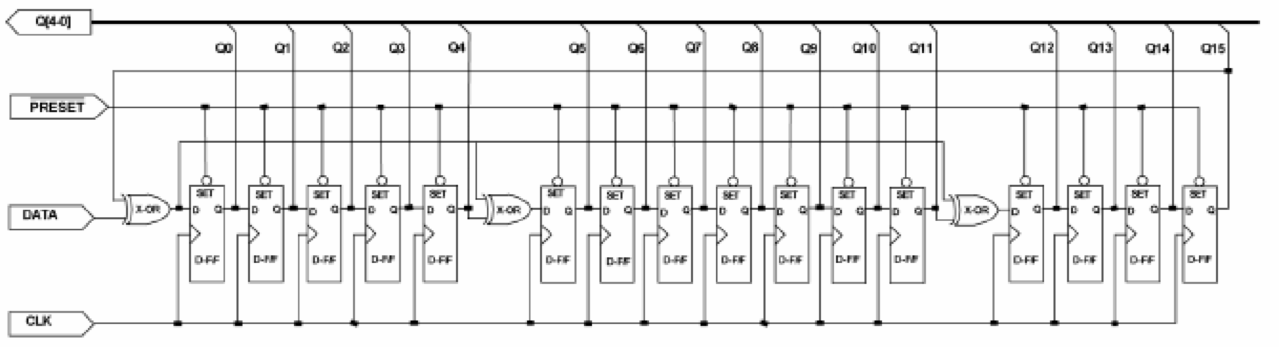


Figure I.2 — Example CRC-16c circuit

Annex J
(informative)

MODE 3: ASK Method: Interrogator-to-tag link modulation

J.1 Baseband waveforms, modulated RF, and detected waveforms

Figure J.1 — Interrogator-to-tag modulation shows R=>T baseband and modulated waveforms as generated by an interrogator, and the corresponding waveforms envelope-detected by a tag, for DSB-ASK modulation.

DSB-A SK Baseband Data: 010

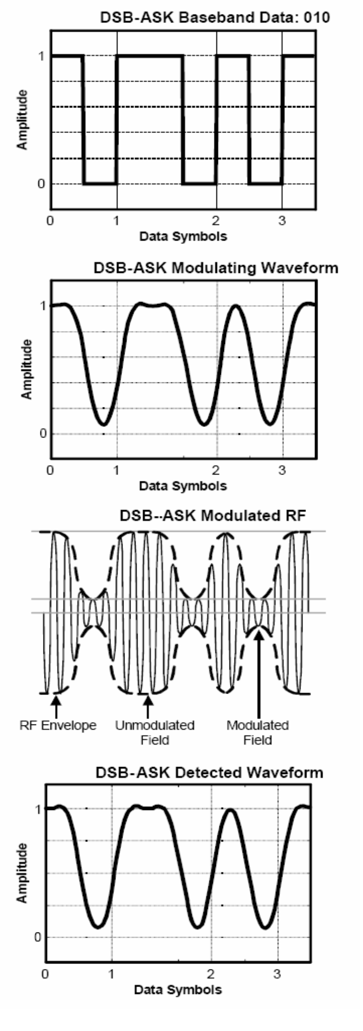


Figure J.1 — Interrogator-to-tag modulation

Note: Drawing not to scale. The DSB-ASK Modulated RF waveform has to respect the modulation index specified in Table 32 — ASK Method: RF envelope parameters.

Annex K
(informative)

MODE 3: Example data flow exchange

K.1 Overview of the data-flow exchange

The following example describes a data exchange, between an interrogator and a single tag, during which the interrogator reads the kill password stored in the tag Reserved memory. This example assumes that:

- The tag has been singulated and is in the **acknowledged** state.
- The tag Reserved memory is locked but not permalocked, meaning that the interrogator must issue the access password and transition the tag to the **secured** state before performing the read operation.
- The random numbers the tag generates (listed in sequence, and not random for reasons of clarity) are:
    - CRC16XXXX$_h$ (the StoredCRC the tag transmitted prior to entering **acknowledged**)
    - RN16_1 1601$_h$ (shall become the handle for the entire access sequence)
    - RN16_2 1602$_h$
    - RN16_3 1603$_h$
- The tag UII is 64 bits in length.
- The tag access password is ACCEC0DE$_h$.
- The tag kill password is DEADC0DE$_h$.
- The 1st half of the access password EXORed with RN16_2 = ACCE$_h$ $\otimes$ 1602$_h$ = BACC$_h$.
- The 2nd half of the access password EXORed with RN16_3 = C0DE$_h$ $\otimes$ 1603$_h$ = D6DD$_h$.

K.2 Tag memory contents and lock-field values

Table K.1 — Tag memory contents and Table K.2 — Lock-field values show the example tag memory contents and lock-field values, respectively.

Table K.1 — Tag memory contents

Memory Bank	Memory Contents	Memory Addresses	Memory Values
TID	TID[15:0]	10$_h$–1F$_h$	54E2$_h$
	TID[31:16]	00$_h$–0F$_h$	A986$_h$
UII	UII[15:0]	50$_h$–5F$_h$	3210$_h$
	UII[31:16]	40$_h$–4F$_h$	7654$_h$
	UII[47:32]	30$_h$–3F$_h$	BA98$_h$
	UII[63:48]	20$_h$–2F$_h$	FEDC$_h$
	StoredPC[15:0]	10$_h$–1F$_h$	2000$_h$
	StoredCRC-16[15:0]	00$_h$–0F$_h$	XXXX$_h$ as calculated (see Annex I)
Reserved	access password[15:0]	30$_h$–3F$_h$	C0DE$_h$
	access password[31:16]	20$_h$–2F$_h$	ACCE$_h$
	kill password[15:0]	10$_h$–1F$_h$	C0DE$_h$
	kill password[31:16]	00$_h$–0F$_h$	DEAD$_h$

Table K.2 — Lock-field values

Kill Password		Access Password		UII Memory		TID Memory		User Memory	
1	0	1	0	0	0	0	0	N/A	N/A

K.3 Data-flow exchange and command sequence

The data-flow exchange follows the ***Access*** procedure outlined in Figure 49 — Access procedure with a Read command added at the end. The sequence of interrogator commands and tag replies is:

Step 1: *Req_RN*[CRC16, CRC-16c]
Tag loadmodulates RN 16_1, which becomes the handle for the entire access sequence

Step 2: *Req_RN*[handle CRC-16c]
Tag loadmodulates RN 16_2

Step 3: *Access*[access password[31:16] EXORed with RN16_2, handle CRC-16c]
Tag loadmodulates handle

Step 4: *Req_RN*[handle CRC-16c]|
Tag loadmodulates RN16_3

Step 5: *Access*[access password[15:0] EXORed with RN16_3, handle CRC-16c]
Tag loadmodulates handle

Step 6: *Read*[MemBank=Reserved, WordPtr=00_h, WordCount=2, handle CRC-16c]
Tag loadmodulates kill password

Table K.3 — Interrogator commands and tag replies shows the detailed interrogator commands and tag replies. For reasons of clarity, the CRC-16c has been omitted from all commands and replies.

Table K.3 — Interrogator commands and tag replies [1]

Step	Data Flow	Command	Parameter and/or Data	Tag State
1a: *Req_RN* command	R => T	11000001	0001 0110 0000 0000 (CRC16=$XXXX_h$)	**acknowledged → open**
1b: tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
2a: *Req_RN* command	R => T	11000001	0001 0110 0000 0001 (handle=1601_h)	**open → open**
2b: tag response	T => R		0001 0110 0000 0010 (RN16_2=1602_h)	
3a: *Access* command	R => T	11000110	1011 1010 1100 1100 ($BACC_h$) 0001 0110 0000 0001 (handle=1601_h)	**open → open**
3b: tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
4a: *Req_RN* command	R => T	11000001	0001 0110 0000 0001 (handle=1601_h)	**open → open**
4b: tag response	T => R		0001 0110 0000 0011 (RN16_2=1603_h)	
5a: *Access* command	R => T	11000110	1101 0110 1101 1101 ($D6DD_h$) 0001 0110 0000 0001 (handle=1601_h)	**open → secured**
5b: tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
6a: *Read* command	R => T	11000010	00 (MemBank=Reserved) 00000000 (WordPtr=kill password) 00000010 (WordCount=2) 0001 0110 0000 0001 (handle=1601_h)	**secured → secured**
6b: tag response	T => R		0 (header) 1101 1110 1010 1101 ($DEAD_h$) 1100 0000 1101 1110 ($C0DE_h$)	

Note: The example above shows the optional usage of the cover-coding of the access password.

----

[1] The tag response in one step can be used by the interrogator as parameter in the command for the next step

Annex L
(informative)

MODE 3: Tag Features

This annex provides a summary of the features that are available for tags that are compliant with MODE3.

L.1 Optional Tag passwords

Kill password: A tag may optionally implement a kill password. 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. See 6.3.3.4.1.1.1.

Access password: A tag may optionally implement an access password. 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. See 6.3.3.4.1.1.2.

L.2 Optional Tag memory banks and memory-bank sizes

Reserved memory: Reserved memory is optional. If a tag does not implement either a kill password or an access password, then the tag need not physically implement Reserved memory. Because a tag with non-implemented passwords operates as if it has zero-valued password(s) that are permanently read/write locked, these passwords must still be logically addressable in Reserved memory at the memory locations specified in 6.3.3.4.1.1.1 and 6.3.3.4.1.1.2.

UII memory: UII memory is required, but its size is vendor-defined. The minimum size is 32 bits, to contain a 16-bit CRC-16 and 16-bit PC word. UII memory may be larger than 32 bits, to contain an UII whose vendor-specified length may be 16 bits to 464 bits in 16-bit increments, as well as a mandatory XPC_W1 and an optional XPC_W2 word. See 6.3.3.4.1.2.

TID memory: TID memory is required, but its size is vendor-defined. The minimum-size TID memory contains an 8-bit ISO/IEC 15963 allocation class identifier, as well as sufficient identifying information for an Interrogator to uniquely identify the custom commands and/or optional features that a tag supports. TID memory may optionally contain vendor-specific data. See 6.3.3.4.1.4.

User memory: User memory is optional. See 6.3.3.4.1.4, 6.3.3.4.1.4.1, and 6.3.3.4.1.4.2.

L.3 Optional Tag commands

Proprietary: A tag may support proprietary commands. See 6.0.9.

Custom: A tag may support custom commands. See 6.0.8.

Access: A tag may support the Access command. See 6.3.3.4.11.3.6.

BlockWrite: A tag may support the BlockWrite command. See 6.3.3.4.11.3.7.

BlockErase: A tag may support the BlockErase command. See 6.3.3.4.11.3.8.

BlockPermalock: A tag may support the BlockPermalock command. See 6.3.3.4.11.3.9.

Kill: A tag may support the *Kill* command. Only support of Recom bit 3SB is mandatory. See 6.3.3.4.11.3.4

L.4 Optional Tag error-code reporting format

A tag may support error-specific or non-error-specific error-code reporting. See Annex E.

L.5 Optional Tag functionality

A tag may implement the UMI by one of two methods. See 6.3.3.4.1.2.2.

A tag may implement a second XPC word (XPC_W2). See 6.3.3.4.1.2.2 and 6.3.3.4.1.2.5.

A tag may implement Recom bits LSB and 2SB. See 6.3.3.4.4.7, 6.3.3.4.10, and 6.3.3.4.11.3.4.

A tag may implement cover-coding. See 6.3.3.4.11.3.3

L.6 Optional Tag Feature

A tag may optionally implement a PJM Method (see section 6)

Annex M
(informative)

Cyclic Redundancy Check (CRC) (16 bit)

M.1 The CRC error detection method

Mode 1: The Cyclic Redundancy Check (CRC) is calculated on all data contained in a message, from the start of the flags through to the end of data. This CRC is used from interrogator to RF tag and from RF tag to interrogator, as appropriate.

Mode 2: The Cyclic Redundancy Check (CRC) is calculated from the end of the start flag field. This CRC is used from interrogator to RF tag only.

Table M.1 — CRC definition

CRC Type	Length	Polynomial	Direction	Preset	Residue
ISO/IEC 13239	16 bits	$x^{16} + x^{12} + x^{5} + 1 \quad = 8408_h$	Backward	$FFFF_h$	$F0B8_h$

To add extra protection against shifting errors, a further transformation on the calculated CRC is made. The One's complement of the calculated CRC is the value attached to the message for transmission. This transformation is included in the example below.

For checking of received messages the 2 CRC bytes are often also included in the re-calculation, for ease of use.

In this case, given the expected value for the generated CRC is the residue of $F0B8_h$.

M.2 CRC calculation example

This example in C language illustrates one method of calculating the CRC on a given set of bytes comprising a message.

```
#include <stdio.h>"
#define  POLYNOMIAL     0x8408        //  x^16 + x^12 + x^5 + 1
#define  PRESET_VALUE   0xFFFF
#define  CHECK_VALUE    0xF0B8
#define  NUMBER_OF_BYTES    4         // Example: 4 data bytes
#define  CALC_CRC          1
#define  CHECK_CRC          0
void main()
{
  unsigned int  current_crc_value;
  unsigned char array_of_databytes[NUMBER_OF_BYTES + 2] = {1, 2, 3, 4, 0x91, 0x39};
  int          number_of_databytes = NUMBER_OF_BYTES;
  int          calculate_or_check_crc;
  int          i, j;
  calculate_or_check_crc = CALC_CRC;
  //  calculate_or_check_crc = CHECK_CRC;  // This could be an other example
  if (calculate_or_check_crc == CALC_CRC)
  {
     number_of_databytes = NUMBER_OF_BYTES;
  }
  else    // check CRC
  {
     number_of_databytes = NUMBER_OF_BYTES + 2;
  }
  current_crc_value = PRESET_VALUE;
  for (i = 0; i < number_of_databytes; i++)
  {
     current_crc_value = current_crc_value ^ ((unsigned int)array_of_databytes[i]);
     for (j = 0; j < 8; j++)
     {
        if (current_crc_value & 0x0001)
        {
           current_crc_value = (current_crc_value >> 1) ^ POLYNOMIAL;
        }
        else
        {
           current_crc_value = (current_crc_value >> 1);
        }
     }
  }
  if (calculate_or_check_crc == CALC_CRC)
  {
     current_crc_value = ~ current_crc_value;
..}
..else // check CRC
..{
     if (current_crc_value == CHECK_VALUE)
     {
        printf ("Checked CRC is ok (0x%04X)\n", current_crc_value);
     }
     else
     {
        printf ("Checked CRC is NOT ok (0x%04X)\n", current_crc_value);
     }
  }
}
```

Annex N
(informative)

Cyclic redundancy check (CRC) mode 2 (32 bit)

N.1 The CRC 32 error detection method

The Cyclic Redundancy Check (CRC 32) is calculated on all data contained in a reply, from the end of the flag through to the end of data. This CRC is used for replies from tag to interrogator. See Table N.1 — CRC 32 Definition.

Table N.1 — CRC 32 Definition

Length	Polynomial	Direction	Preset	Residue
32 bits	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$ = EDB88320$_h$	Backward	FFFFFFFF$_h$	2144DF1C$_h$

To add extra protection against shifting errors, a further transformation on the calculated CRC is made. The ones complement of the calculated CRC is the value attached to the message for transmission. This transformation is included in the example below.

For checking of received messages the 2 CRC words may be also included in the re-calculation, for ease of use. In this case, the expected value for the generated CRC is the residue '2144DF1C$_h$'.

N.2 CRC 32 calculation example

This example in C language illustrates one method of calculating the CRC on a given set of words comprising a reply.

```
#define POLYNOMIAL      0xEDB88320

#define PRESET_VALUE  0xFFFFFFFF

#define CHECK_VALUE    0x2144DF1C

#include <stdio.h>
#include <stddef.h>

unsigned long calculate_CRC32(unsigned long current_crc_value, unsigned int new_word)
{
  int  i;

  current_crc_value = current_crc_value ^ ((unsigned int)new_word);

  for (i = 0; i < 16; i++)
  {
    if (current_crc_value & 0x0001)
    {
       current_crc_value = (current_crc_value >> 1) ^ POLYNOMIAL;
    }
    else
    {
      current_crc_value = (current_crc_value >> 1);
    }
  }
  return(current_crc_value);
}

void main()
{

  unsigned long crc32;
  unsigned long crc_send;

  crc32 = PRESET_VALUE;                    // preset crc to 0xFFFFFFFF
  crc32 = calculate_CRC32(crc32, 0x1234);      // typical tag reply
  crc32 = calculate_CRC32(crc32, 0x0002);
  crc32 = calculate_CRC32(crc32, 0x0003);
  crc32 = calculate_CRC32(crc32, 0x0010);
  crc32 = calculate_CRC32(crc32, 0x0011);

  printf("CRC Results:\n");
  printf("-------------\n");

  crc_send = ~ crc32;              // inverse of calculated crc is sent
  printf("Send    = 0x%08X\n",crc_send);

  // if transmitted crc is included in calculation by interrogator,
  // CHECK_VALUE should result

  crc32 = calculate_CRC32(crc32, (unsigned int)(crc_send) & 0x0000FFFF);
  crc32 = calculate_CRC32(crc32, (unsigned int)((crc_send >> 16) &  0x0000FFFF));

  crc32 = ~ crc32;                // invert output to recover residue
  printf("Residue = 0x%08X\n",crc32);

  if (crc32 == CHECK_VALUE)       // should equal CHECK_VALUE
      printf("CRC OK  \n");
  else printf("CRC BAD \n");

}
```

This program, when run should produce the following output:

```
CRC Results:

-------------

Send    = 0xC7D5219F

Residue = 0x2144DF1C

CRC OK
```

N.3 Practical example of CRC 32 calculation

This example refers to a Short Reply command for a two word read starting from tag memory address 10_h.

The Specific Identifier of the tag in this example is 00030002_h and the command number has been set to 1234_h. The tag memory locations could contain any value, in this example they are assumed to be as in Table N.2 — Sample Tag Memory Location Values:

Table N.2 — Sample Tag Memory Location Values

Address	Contents
10_h	0010_h
11_h	0011_h

The reply in this example consists of the following fields:

- The 'Time Stamp' whose value is the transmitted command number '1234_h'
- The tag's Specific Identifier '00030002_h' which is transmitted as two words, least significant word first.
- The contents of the requested two memory locations, lowest address first: '0010_h' '0011_h'
- The CRC: '$C7D5219F_h$' where '$219F_h$' is the least significant word, which is transmitted first.

The reply is then transmitted as follows:

1234 0002 0003 0010 0011 219F C7D5

Table N.3 — Sample Calculated CRC in interrogator

Step	Input	Calculated CRC in Interrogator
1	Initialised	$FFFFFFFF_h$
2	1234_h	$094A9040_h$
3	0002_h	$7399A576_h$
4	0003_h	$B52DBBB2_h$
5	0010_h	$815B13BD_h$
6	0011_h	$C7D5219F_h$
7	$219F_h$	$41D9D52A_h$
8	$C7D5_h$	$2144DF1C_h$

NOTE: The CRC which is finally transmitted, '$C7D5219F_h$' is present as the 'Calculated CRC in interrogator' at step 6, after each of the data words in the reply has been included in the calculation.

Continuing the calculation with the next two words (the transmitted CRC) produces the final result whose value should be the residue '$2144DF1C_h$'.

NOTE: The CRC is transmitted as two words, with the least significant word transmitted first.

NOTE: Each word is transmitted least significant bit first.

Annex O
(informative)

Known possible interferences between the MODES determined in this part of ISO/IEC 18000

No interference known.

Note: Interference is unlikely because the three modes are completely different and all have a high level of data integrity (ITF, framing, CRC).

Bibliography

[1] ERC RECOMMENDATION 70-03, relating to the use of SRDs. Recommendation adopted by the Frequency Management, Regulatory Affairs and Spectrum Engineering Working Groups

[2] ETSI EN 300 220-1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW; Part 1: Technical characteristics and test methods*

[3] ETSI EN 300 220-2, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW; Part 2: Supplementary parameters not intended for conformity purposes*

[4] ETSI EN 300 220-3, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW; Part 3: Harmonized EN covering essential requirements under article 3.2 of the R&TTE Directive*

[5] ETSI EN 300 330, Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Technical characteristics and test methods for radio equipment in the frequency range 9 kHz to 30 MHz

[6] ETSI I-ETS 300 440, *Radio Equipment and Systems (RES); Short range devices (SRDs); Technical characteristics and test methods for radio equipment to be used in the 1 GHz to 25 GHz frequency range*

[7] ISO/IEC 10373-7, *Identification cards — Test methods — Part 7: Vicinity cards*

[8] ISO/IEC 18000-6, Information technology — Radio frequency identification for item management — Part 6: Parameters for air interface communications at 860 MHz to 960 MHz