INTERNATIONAL STANDARD

ISO/IEC 18000-61

First edition
2012-07-15

Information technology — Radio frequency identification for item management —

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

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

Partie 61: Paramètres de communications d'une interface radio entre 860 MHz et 960 MHz, Type A

Reference number
ISO/IEC 18000-61:2012(E)

Contents

Page

Foreword

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

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

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

ISO/IEC 18000-61 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

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

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

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

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

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

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

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

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

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

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

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

Introduction

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

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

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

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

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

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

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

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

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

Information on the declared patents may be obtained from:

Contact details	
Patent Holder Legal Name: Impinj, Inc.	**Contact for license application** Name & Department: Chris Diorio, CTO Address: 701 N 34th St Suite 300, Seattle, WA 98103, USA Tel.: +1 206 834 1115 Fax: +1 206 517 5262 E-mail: diorio@impinj.com URL (optional): www.impinj.com
Patent Holder Legal Name: NXP B.V.	**Contact for license application** Name & Department: Aaron Waxler, Intellectual Property & Licensing Address: 411 East Plumeria, San José, CA 95134-1924, USA Tel.: +1 914 860 4296 E-mail: Aaron.Waxler@nxp.com

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

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

Information technology — Radio frequency identification for item management —

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

1 Scope

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

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

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

Type A is ITF. Type A uses Pulse-Interval Encoding (PIE) in the forward link and an adaptive ALOHA collision-arbitration algorithm.

This part of ISO/IEC 18000 specifies

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

— Interrogator and tag operating procedures and commands,

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

2 Conformance

2.1 Claiming conformance

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

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

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

2.2 Interrogator conformance and obligations

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

— support Type A

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

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

— operate within the applicable local regulations.

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

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

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

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

— implement any command that conflicts with this part of ISO/IEC 18000 or any of the parts 62, 63 and 64, or

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

2.3 Tag conformance and obligations

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

— support Type A;

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

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

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

— conform to local radio regulations.

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

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

— implement any proprietary and/or custom commands as defined in 6.9 and 6.10.

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

— implement any command that conflicts with this part of ISO/IEC 18000 or any of the parts 62, 63 and 64;

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

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

3 Normative references

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

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

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

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

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

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

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

4 Terms, definitions, symbols and abbreviated terms

4.1 Terms and definitions

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

4.1.1
collision arbitration
algorithm used to prepare for and handle a dialogue between an Interrogator and a tag

4.1.2
physical layer
data coding and modulation waveforms used in Interrogator-to-tag and tag-to-Interrogator signalling

4.1.3
battery assistance
battery support for radio frequency communication

4.1.4
battery assisted mode
working mode of battery-assisted tags with non-empty battery

4.1.5
passive mode
working mode of passive tags or battery assisted tags with battery drained below a manufacturer-specific threshold

4.2 Symbols

BAP	battery assisted passive
BLF	backscatter link frequency
Cht	carrier high-level tolerance
Clt	carrier low-level tolerance
D	modulation depth of data coding pulse
Taq	quiet time
Tapf	pulse fall time
Tapr	pulse rise time
Tapw	pulse width
Tari	reference time interval for a data-0 in Interrogator-to-tag signalling
Tcf	carrier fall time
Tcr	carrier rise time
Tcs	carrier steady state time
Tfhf	carrier FHSS fall time
Tfhr	carrier FHSS rise time
Tfhs	carrier FHSS steady time
Trlb	return link bit time
$xxxx_2$	binary notation
$xxxx_h$	hexadecimal notation

4.3 Abbreviated terms

AFI	application family identifier
ASF	application sub family
BAP	battery assisted passive
CRC	cyclic redundancy check
CRC-16	sixteen bit CRC
CRC-5	five bit CRC
CW	continuous wave

DSFID	data storage format identifier
DSSS	direct sequence spread spectrum
EOF	end of frame
FCC	Federal Communications Commission
FHSS	frequency hopping spread spectrum
ITF	Interrogator-talks-first NOTE The common usage is RTF (Reader-talks-first) but the more precise term is ITF, which is used throughout this part of ISO/IEC 18000.
LSB	least significant bit
MSB	most significant bit
PIE	pulse interval encoding
ppm	parts per million
RFU	reserved for future use
SOF	start of frame
SUID	sub unique identifier
TEL	tag excitation level

5 Overview

5.1 Parameter tables

Table 1, Table 2, Table 3 and Table 4 contain the parameters for Type A. in accordance with ISO/IEC 18000-1. Detailed description of the operating modes and parameters are specified in the subsequent clauses.

Table 1 — Interrogator to tag link parameters

Ref.	Parameter Name	Description
Int:1	Operating Frequency Range	860 MHz – 960 MHz, as required by the local regulations
Int:1a	Default Operating Frequency	In accordance with the local radio regulations.
Int:1b	Operating Channels (spread-spectrum systems)	In accordance with the local radio regulations.
Int:1c	Operating Frequency Accuracy	In accordance with the local radio regulations.
Int:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	In accordance with the local radio regulations.
Int:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	In accordance with the local radio regulations.
Int:2	Occupied Channel Bandwidth	In accordance with the local radio regulations.
Int:2a	Minimum Receiver Bandwidth	In accordance with the local radio regulations.
Int:3	Interrogator Transmit Maximum EIRP	In accordance with the local radio regulations.

Ref.	Parameter Name	Description
Int:4	Interrogator Transmit Spurious Emissions	In accordance with the local radio regulations.
Int:4a	Interrogator Transmit Spurious Emissions, In-Band (spread-spectrum systems)	In accordance with the local radio regulations.
Int:4b	Interrogator Transmit Spurious Emissions, Out-of-Band	In accordance with the local radio regulations.
Int:5	Interrogator Transmitter Spectrum Mask	In accordance with the local radio regulations.
Int:6	Timing	See below Int: 6x.
Int:6a	Transmit-to-Receive Turn-Around Time	The Interrogator transmit/receive settling time shall not exceed 85 µs.
Int:6b	Receive-to-Transmit Turn-Around Time	As determined by the communication protocol – refer Tag: 6a.
Int:6c	Dwell Time or Interrogator Transmit Power-On Ramp	1500 µs, maximum settling time
Int:6d	Decay Time or Interrogator Transmit Power-Down Ramp	Maximum 1 ms.
Int:7	Modulation	Amplitude Modulation.
Int:7a	Spreading Sequence (direct-sequence [DSSS] systems)	Not applicable.
Int:7b	Chip Rate (spread-spectrum systems)	Not applicable.
Int:7c	Chip Rate Accuracy (spread-spectrum systems)	Not applicable.
Int:7d	Modulation Depth	Nominal 30 % to 100 %. See Table 11 and clause 7.1.1.
Int:7e	Duty Cycle	In accordance with the local regulations.
Int:7f	FM Deviation	Not applicable.
Int:8	Data Coding	Pulse Interval Encoding
Int:9	Bit Rate	33 kbit/s as constrained by the local radio regulations
Int:9a	Bit Rate Accuracy	100 ppm
Int:10	Interrogator Transmit Modulation Accuracy	Not applicable.
Int:11	Preamble	Has no preamble
Int:11a	Preamble Length	Not applicable.
Int:11b	Preamble Waveform(s)	Not applicable.
Int:11c	Bit Sync Sequence	Not applicable.
Int:11d	Frame Sync Sequence	Not applicable.
Int:12	Scrambling (spread-spectrum systems)	Not Applicable.
Int:13	Bit Transmission Order	MSB is transmitted first
Int:14	Wake-up process	Presence of an appropriate RF signal at the tag followed by a wake-up command as required by the tag type. See relevant clauses.
Int:15	Polarization	Interrogator dependent. Not defined in this part of ISO/IEC 18000.

Table 2 — Tag to Interrogator link parameters

Ref.	Parameter Name	Description
Tag:1	Operating Frequency Range	860 MHz – 960 MHz, inclusive
Tag:1a	Default Operating Frequency	The tag shall respond to an Interrogator signal within the frequency range specified in Tag: 1.
Tag:1b	Operating Channels (spread-spectrum systems)	The tag shall respond to an Interrogator signal within the frequency range specified in Tag: 1.
Tag:1c	Operating Frequency Accuracy	The tag shall respond to an Interrogator signal within the frequency range specified in Tag: 1.
Tag:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	Not applicable.
Tag:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	Not applicable.
Tag:2	Occupied Channel Bandwidth	In accordance with the local regulations
Tag:3	Transmit Maximum EIRP	In accordance with the local regulations
Tag:4	Transmit Spurious Emissions	In accordance with the local regulations
Tag:4a	Transmit Spurious Emissions, In-Band (spread spectrum systems)	In accordance with the local regulations
Tag:4b	Transmit Spurious Emissions, Out-of-Band	In accordance with the local regulations
Tag:5	Transmit Spectrum Mask	In accordance with the local regulations
Tag:6a	Transmit-to-Receive Turn-Around Time	The tag shall open its receive command window within 2 bit periods of the end of its response.
Tag:6b	Receive-to-Transmit Turn-Around Time	Range from 150 to 1150 µs (see clause 7.5.3 and Table 28)
Tag:6c	Dwell Time or Transmit Power-On Ramp	Not applicable.
Tag:6d	Decay Time or Transmit Power-Down Ramp	Not applicable.
Tag:7	Modulation	Bi-state amplitude modulated backscatter.
Tag:7a	Spreading Sequence (direct sequence [DSSS] systems)	Not applicable.
Tag:7b	Chip Rate (spread spectrum systems)	Not applicable.
Tag:7c	Chip Rate Accuracy (spread spectrum systems)	Not applicable.
Tag:7d	On-Off Ratio	The tag Delta RCS (Varying Radar Cross Sectional area) affects system performance. A typical value is greater than 0.005m2.
Tag:7e	Subcarrier Frequency	Not applicable.
Tag:7f	Subcarrier Frequency Accuracy	Not applicable.
Tag:7g	Subcarrier Modulation	Not applicable.
Tag:7h	Duty Cycle	The tag shall transmit its response when commanded to do so by the Interrogator.
Tag:7i	FM Deviation	Not applicable.
Tag:8	Data Coding	Bi-phase space (FM0)

Ref.	Parameter Name	Description
Tag:9	Bit Rate	Typical 40 kbit/s or 160 kbit/s (subject to tag clock tolerance see Table 9),
Tag:9a	Bit Rate Accuracy	+/- 15% (refer to Table 9)
Tag:10	Tag Transmit Modulation Accuracy (frequency-hopping [FHSS] systems	Not applicable.
Tag:11	Preamble	The preamble is defined in clause 6.5.6
Tag:11a	Preamble Length	16 bits made up of a quiet period, followed by sync, followed by a code violation followed by an orthogonal code.
Tag:11b	Preamble Waveform	Bi-phase encoded data '1'.
Tag:11c	Bit-Sync Sequence	Included in the preamble.
Tag:11d	Frame-Sync Sequence	Included in the preamble.
Tag:12	Scrambling (spread-spectrum systems)	Not applicable.
Tag:13	Bit Transmission Order	MSB is transmitted first
Tag:14	Reserved	Deliberately left blank.
Tag:15	Polarization	Product design feature. Not defined in this part of ISO/IEC 18000.
Tag:16	Minimum Tag Receiver Bandwidth	860 – 960 MHz

Table 3 — Protocol parameters

Ref.	Parameter Name	Description
P:1	Who talks first	Interrogator
P:2	Tag addressing capability	see clause 7.3.1
P:3	Tag ID	Contained in tag memory and accessible by means of a command.
P:3a	Tag ID Length	64 bits.
P:3b	Tag ID Format	See clause 7.2.1
P:4	Read size	Addressable in blocks of up to 256 bits, but always an integer multiple of bytes.
P:5	Write Size	Addressable in blocks of up to 256 bits, but always an integer multiple of bytes.
P:6	Read Transaction Time	A single tag can typically be identified and have its first 128 bits of user memory read in less than 10 ms. This time may vary depending on the data rate used as constrained by the local radio regulations.
P:7	Write Transaction Time	Once a tag has been identified and selected, a 32-bit data block can typically be written in less than 20 ms. This time may vary depending on the data rate used as constrained by the local radio regulations.

Ref.	Parameter Name	Description
P:8	Error detection	Interrogator to tag: 16-bit commands: CRC-5. Commands of more than 16 bits have an additional CRC-16, bit protection. Tag to Interrogator: CRC-16
P:9	Error correction	No forward error correction code used. Errors are handled by signalling an error to the Interrogator that then repeats its last transmission. None
P:10	Memory size	No minimum user memory size is specified, but if user memory is provided it shall be an integer multiples of 4 bytes.
P:11	Command structure and extensibility	Several command codes are reserved for future use. In addition, a Protocol extension flag allows for further extensions.

Table 4 — Anti-collision parameters

Ref.	Parameter Name	Description
A:1	Type (Probabilistic or Deterministic)	Probabilistic
A:2	Linearity	Essentially linear using adaptive slot allocation up to 256 slots for 250 tags in an Interrogator zone.
A:3	Tag inventory capacity	The algorithm permits the reading of not less than 250 tags in the reading zone of the Interrogator.

6 Type A

6.1 Physical layer and data coding

6.1.1 Interrogator power-up waveform

The Interrogator power-up waveform shall comply with the mask specified in Figure 1 and Table 5.

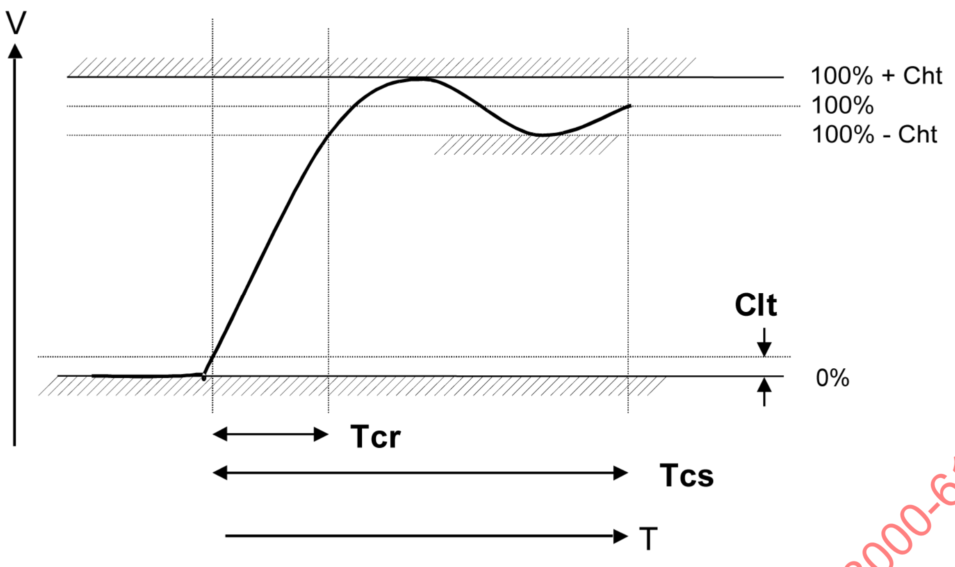


Figure 1 — Interrogator power-up waveform

Table 5 — Interrogator power-up waveform parameter values

Parameter	**Min**	**Max**
Tcs		1500µs
Tcr	1 µs	500 µs
Cht		10%
Clt		1%

6.1.2 Interrogator power-down

Once the carrier level has dropped below the ripple limit Cht, power down shall be monotonic and of duration Tcf, as specified in Figure 2 and Table 6.

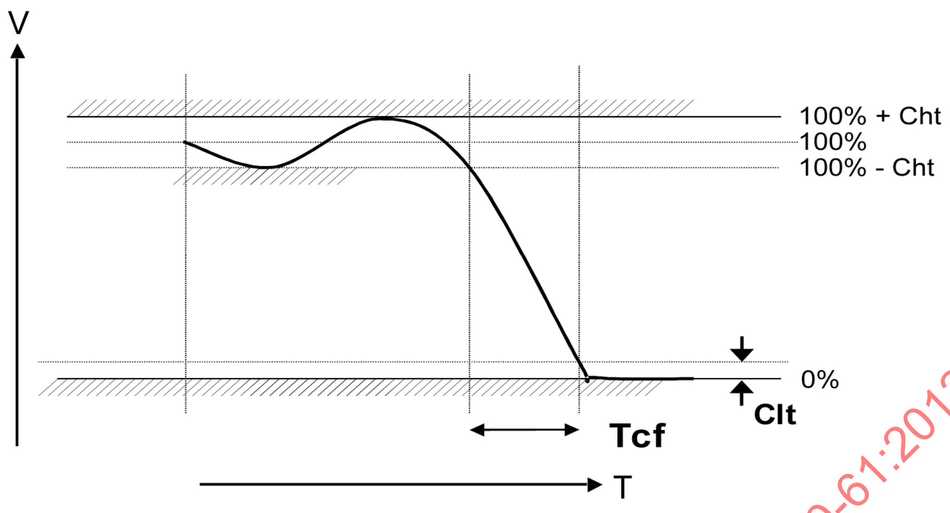


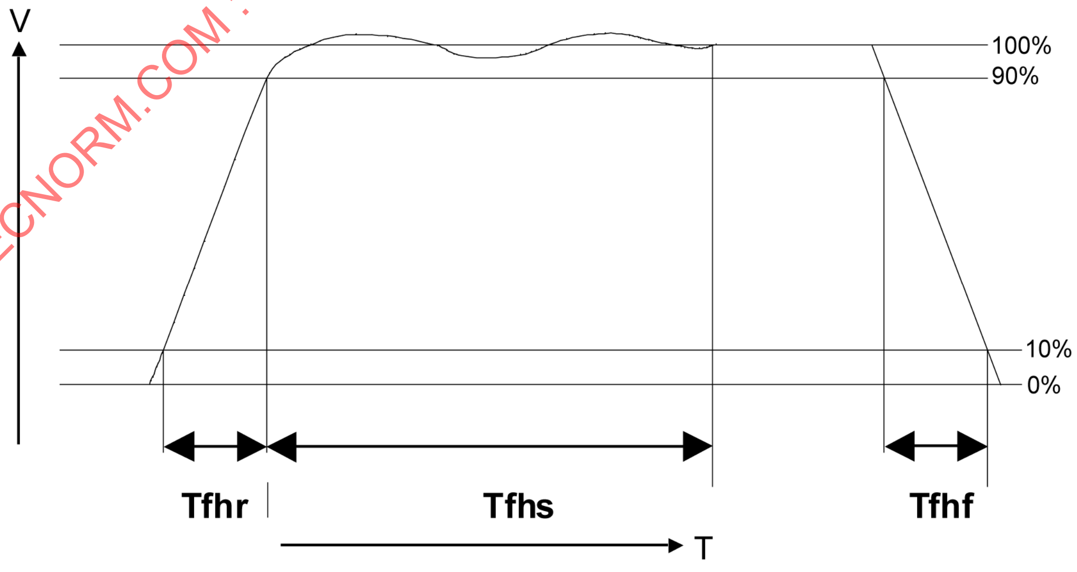
Figure 2 — Interrogator power-down waveform

Table 6 — Interrogator power-down timings

Parameter	Min	Max
Tcf	1 µs	500 µs
Cht		± 5% of steady state (100 %) level
Clt		1 %

6.1.3 Frequency hopping carrier rise and fall times

When the Interrogator operates in the frequency operating hopping spread spectrum mode (FHSS), the carrier rise and fall times shall conform to the characteristics specified in Figure 3 and Table 7. The Interrogator shall complete a frequency hop in a time not exceeding 30 µs (to ensure that the tag is not reset by the frequency hop). The frequency hop is measured from the beginning of Tfhf to the end of Tfhr.



NOTE Ripple is ± 5 % of 100 % of steady state level

Figure 3 — FHSS carrier rise and fall characteristics

Table 7 — FHSS carrier rise and fall parameters

Parameter	Min	Max
Tfhr		30 µs
Tfhs	400 µs	
Tfhf		30 µs

6.1.4 FM0 return link

6.1.4.1 FM0 return link general

The tag transmits information to the Interrogator by modulating the incident energy and reflecting it back to the Interrogator (backscatter).

6.1.4.2 Modulation

The tag switches its reflectivity between two states. The “space” state is the normal condition in which the tag is powered by the Interrogator and able to receive and decode the forward link. The “mark” state” is the alternative condition created by changing the antenna configuration or termination.

6.1.4.3 Data rate

The return link data rate shall be 40 or 160 kbit/s, which may be selected at the time of tag configuration. The Interrogator shall be able to read and decode the tag reply at either data rate without the need to have prior knowledge of the tag configuration in order to handle mixed populations.

6.1.4.4 Data coding

Data is coded using the FM0 technique, also known as Bi-Phase Space.

One symbol period Trlb, as specified in Table 8, is allocated to each bit to be sent. In FM0 encoding, data transitions occur at all bit boundaries. In addition, data transitions occur at the mid-bit of logic 0 being sent.

Table 8 — Return link parameters

Data rate	Trlb	Tolerance	Note
40kbit/s	25 µs	+/-15%	Chip set to 40kbit/s Return Link Data Rate
160kbit/s	6.25 µs	+/-15%	Chip set to 160kbit/s Return Link Data Rate

Coding of data is MSB first. Figure 4 illustrates the coding for the 8 bits of 'B1'.

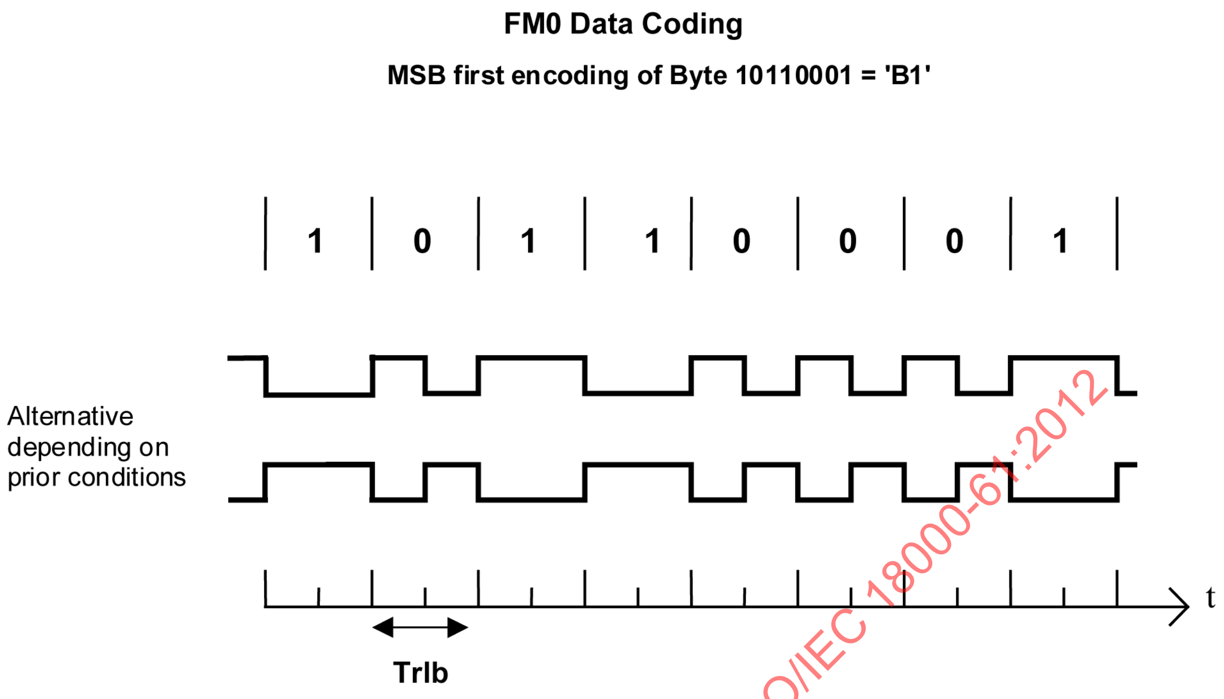

FM0 Data Coding
MSB first encoding of Byte 10110001 = 'B1'
1
0
1
1
0
0
0
1
Alternative depending on prior conditions
t
Trlb

Figure 4 — Tag to Interrogator data coding

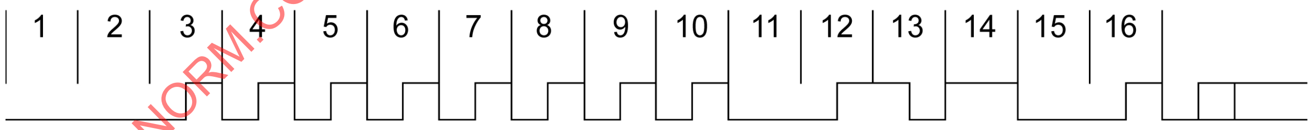
6.1.4.5 Message format

A return Link Message consists of n data bits preceded by the Preamble. The data bits are sent MSB first.

The Preamble enables the Interrogator to lock to the tag data clock and begin decoding of the message. It consists of 16 bits as shown in Figure 5. There are multiple code violations (sequences not conforming to FM0 rules) that act as a frame marker for the transition from Preamble to Data.

6.1.4.6 Return preamble

The return preamble is a sequence of backscatter modulation specified in Figure 5.



NOTE The high state represents high reflectivity and the low state represents low reflectivity.

Figure 5 — Preamble waveform

Changing the tag's modulator switch from the high impedance state to the low impedance state causes a change in the incident energy to be back-scattered, see Figure 6.

The tag shall execute backscatter, a half-low and half-high sent by the tag defined as follows:

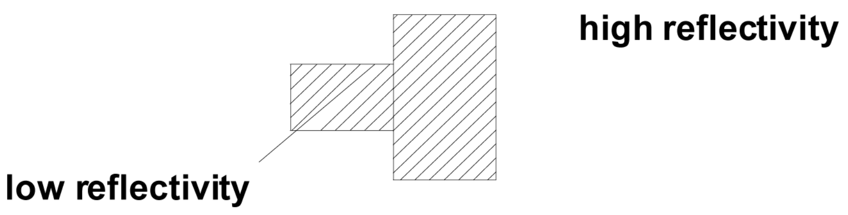


Figure 6 — Return link bit coding

6.1.4.7 Cyclic redundancy check (CRC)

6.1.4.7.1 CRC General

The same CRC-16 is used for the forward and the return links. In addition, short commands use a CRC-5 when it satisfies the required level of protection against errors.

On receiving a command from the Interrogator, the tag shall verify that the checksum or the CRC value is valid. If it is invalid, it shall discard the frame, shall not respond and shall not take any other action.

6.1.4.7.2 Interrogator to tag 5-bit CRC-5

The 5-bit CRC shall be calculated on all the command bits after the SOF up to but not including the first CRC bit.

The polynomial used to calculate the CRC is x^5+x^3+1.

A possible implementation is using a 5-bit shift register as defined in Clause A.1. The 5-bit CRC register is named Q4 to Q0, with Q4 being the MSB and Q0 being the LSB. The 5-bit register must be preloaded with '01001' (MSB to LSB) or in hexadecimal notation 0x09 (HEX), as shown in Table A.1

The 11 bits of data must be clocked through the CRC register, using the MSB first. The 5 CRC bits are then sent, MSB first. After the LSB of the CRC-5-bit is clocked through, the 5-bit CRC register should contain all zero's.

NOTE A schematic of a possible implementation is provided in Annex A.

6.1.4.7.3 Interrogator to tag 16-bit CRC-16

6.1.4.7.3.1 Interrogator to tag CRC-16 general

The 16-bit CRC shall be calculated on all the command bits after the SOF up to but not including the first CRC bit.

The polynomial used to calculate the CRC is $x^{16} + x^{12} + x^5 + 1$. The 16-bit register shall be preloaded with 'FFFF'. The resulting CRC value shall be inverted, attached to the end of the packet and transmitted.

The most significant byte shall be transmitted first. The most significant bit of each byte shall be transmitted first.

NOTE A schematic of a possible implementation is provided in Annex A.

The CRC may be implemented in one of two ways:

6.1.4.7.3.2 Inversion of incoming CRC bits by the tag

At the tag, the incoming CRC bits are inverted and then clocked into the register. After the LSB CRC bit is clocked into the register the 16-bit CRC register should contain all zero's.

6.1.4.7.3.3 Non-inversion of incoming CRC bits by the tag

If the received CRC bits are not inverted before clocking into the register, then after the LSB CRC bit is clocked into the register the 16-bit CRC register will have the value 0x1D0F (HEX)

6.1.4.7.4 Tag to Interrogator 16-bit CRC-16

6.1.4.7.4.1 Tag to Interrogator CRC-16 general

The 16-bit CRC shall be calculated on all data bits up to, but not including, the first CRC bit.

The polynomial used to calculate the CRC is $x^{16} + x^{12} + x^5 + 1$. The 16-bit register shall be preloaded with 0xFFFF (HEX). The resulting CRC value shall be inverted, attached to the end of the packet and transmitted.

The most significant byte shall be transmitted first, see Table 9. The most significant bit of each byte shall be transmitted first.

On receiving of a response from the tag, it is recommended that the Interrogator verifies that the CRC value is valid. If it is invalid, appropriate remedial action is the responsibility of the Interrogator designer.

NOTE A schematic of a possible implementation is provided in Annex A.

Table 9 — CRC-16 bits and bytes transmission rules

MSByte	LSByte
MSB LSB	MSB LSB
CRC-16 (8 bits)	CRC-16 (8 bits)
↑ first transmitted bit of the inverted CRC	

The CRC may be implemented in one of two ways.

6.1.4.7.4.2 Inversion of incoming CRC bits by the Interrogator

At the Interrogator receiver, the incoming CRC bits are inverted and then clocked into the register. After the LSB CRC bit is clocked into the register the 16-bit CRC register should contain all zero's.

6.1.4.7.4.3 Non-inversion of incoming CRC bits by the Interrogator

If the received CRC bits are not inverted before clocking, the CRC register will have the value 0x1D0F (HEX).

6.1.5 PIE (Pulse interval encoding) forward link

6.1.5.1 Carrier modulation pulses

The data transmission from the Interrogator to the tag is achieved by modulating the carrier amplitude (ASK). The data coding is performed by generating pulses at variable time intervals. The duration of the interval between two successive pulses carries the data coding information.

The tag shall measure the inter-pulse time on the high to low transitions (falling) edges of the pulse as shown in Figure 7.

The carrier modulation pulses are negative-going raised cosine shaped, their characteristics are specified in Figure 8 and Table 10.

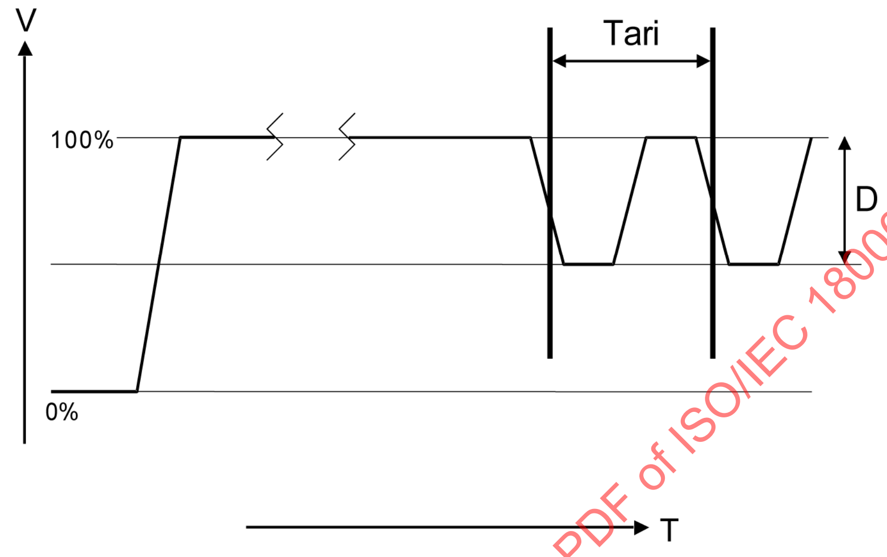
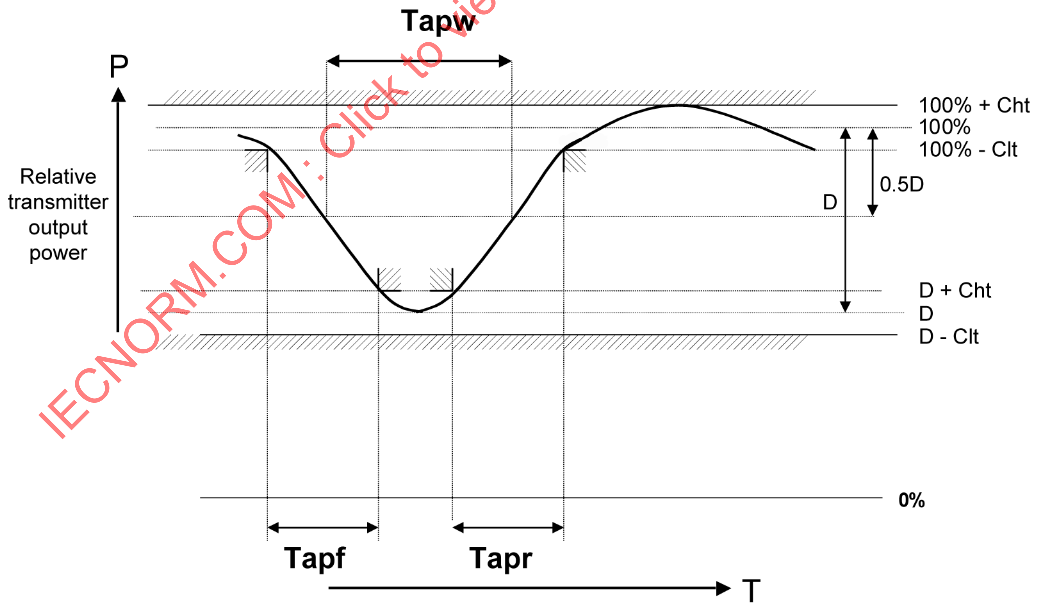


Figure 7 — Inter-pulse mechanism



NOTE The vertical lines occur at the half power (3dB) point on the modulation carrier dip.

Figure 8 — Modulation shaping

Table 10 — Modulation parameters

Parameter	Min	Nominal	Max
Tapw		10µs	
D	27% See NOTE 1		See NOTE 2
Tapf		4µs	
Tapr		4µs	
Cht			0.1 D
Clt			0.1 D
NOTE 1 The minimum modulation depth value is the absolute limit over the Interrogator's operating temperature range. NOTE 2 The maximum modulation depth may be in the range of 0.27 to 1 times the steady state value and will depend on the radio regulatory environment.			

The Interrogator shall maintain a constant modulation depth during the transmission of a command, within the Cht and Clt tolerances.

6.1.5.2 Data coding and framing

The time Tari specifies the reference interval between the falling edges of two successive pulses representing the symbol '0' as transmitted by the Interrogator; its value is specified in Table 11.

Table 11 — Reference interval timing

Tari	Stability
10 to 20 µs	±100 ppm

Four symbols are defined as shown as shown in Table 12 and Figure 9. The symbols are referenced to the interval Tari and have a duration of 1 Tari, 2 Tari, and 4 Tari being integer multiples of Tari and shall have a tolerance with respect to Tari as defined in Table 12. The Start of Frame, SOF symbol is a combinational symbol made up by concatenating a '0' symbol with a symbol having a duration of 3 Tari, for an overall length of 4 Tari.

Table 12 — PIE symbols, coding

Symbol	Duration	Tolerance with respect to 1 Tari
0	1 Tari	
1	2 Tari	± 100 ppm
SOF	1 Tari followed by 3 Tari	± 100 ppm
EOF	4 Tari	± 100 ppm

Symbol	Number of Tari
'0'	1
'1'	2
SOF	4
EOF	4

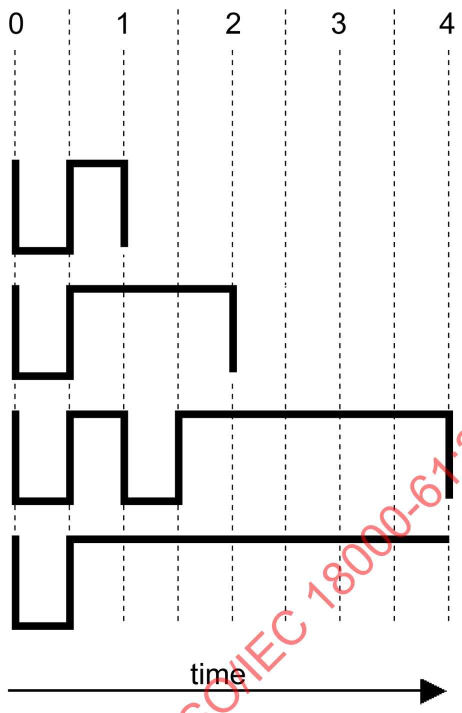


Figure 9 — PIE symbols

6.1.5.3 Decoding of PIE symbols by the tag

The tag shall be able to decode a PIE encoded transmission having symbols of duration as specified in Table 13.

Table 13 — PIE symbols, decoding

Symbol	Mean duration	Limits
0	1 Tari	1/2 Tari < ‘0’ ≤ 3/2 Tari
1	2 Tari	3/2 Tari < ’1’ ≤ 5/2 Tari
SOF	1 Tari followed by 3 Tari	Calibration sequence
EOF	4 Tari	≥ 4 Tari

The reference for the decoder is derived from the SOF and is = 3/2 Tari

The tag shall interpret an interval between received symbols of >4 Tari as an EOF and shall discard the preceding data.

If the tag does not receive data for a period greater than EOF, the tag shall wait for a SOF.

6.1.5.4 Frame format

The bits transmitted by the Interrogator to the tag are embedded in a frame as specified in Figure 10.

Before sending the frame, the Interrogator shall ensure that it has established an unmodulated carrier for a duration of at least Taq (Quiet time) of 300 μs.

The frame consists of a Start Of Frame (SOF), immediately followed by the data bits and terminated by an End Of Frame (EOF). After having sent the EOF the Interrogator shall maintain a steady carrier for the time specified by the protocol so that the tags are powered for transmitting their response.

If the tag does not receive data for a period greater than EOF, then the command decoder shall revert to the ready state and await another SOF.

The tag shall interpret an interval between received symbols of >4 Tari as an error and shall discard the preceding data.

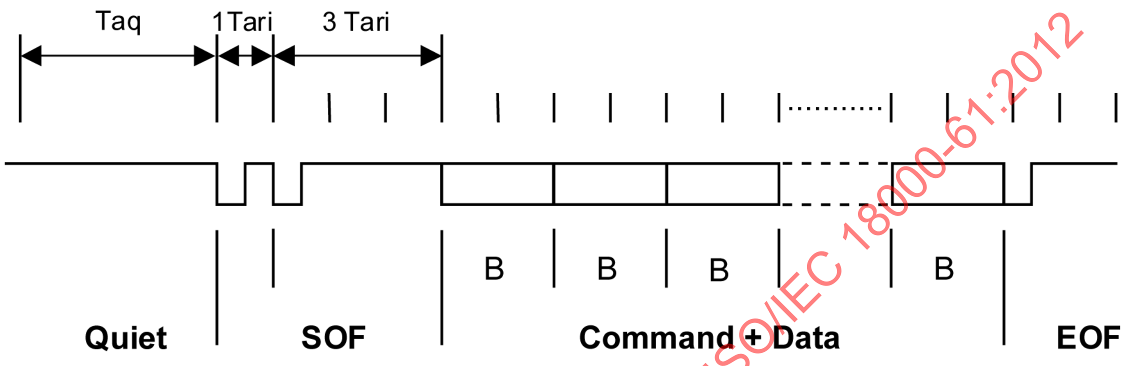


Figure 10 — Forward link frame format

6.1.5.5 Data decoding

The tag shall decode the symbols specified in Table 12.

The tag shall be able to decode an incoming signal having a modulation depth (as specified in 6.1.5.1, Figure 8 and Table 10) with respect to the Interrogator steady state carrier of 27 % or greater (i.e. 27 % to 100 %). In other words the modulation depth may be anywhere in the range of 0.27 to 1 times the steady state carrier amplitude.

If the tag detects an invalid code, it shall discard the frame and wait for an unmodulated carrier of 4 Tari duration.

6.1.5.6 Bits and byte ordering

Coding of data into symbols shall be MSB first. The coding for the 8 bits of hex byte 'B1' is shown in Figure 11.

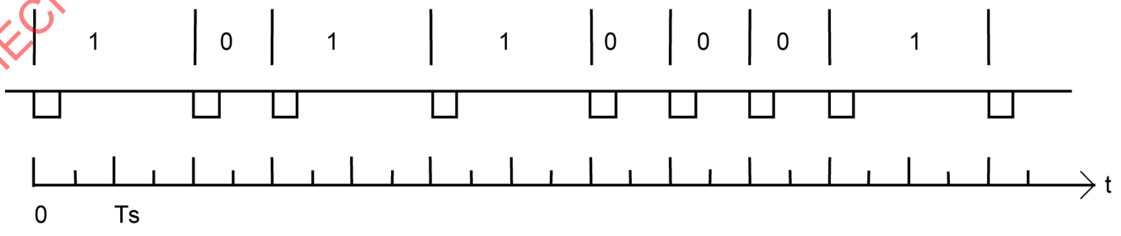


Figure 11 — Example of PIE byte encoding for 'B1'

6.2 Data elements

6.2.1 Unique identifier (Tag ID)

Tags may be uniquely identified in order to provide a one-to-one transaction between the tag and the Interrogator and to provide traceability information relating to the transponder chip. When thus used, the tag unique identification number (Tag ID) shall be in this form as specified in Table 14 and ISO/IEC 15963.

If used, the Tag ID shall be set permanently by the IC manufacturer in accordance with Table 14.

Table 14 — Tag ID format

MSB			**LSB**
b64 .. b57	b56 .. b49	b48 .. b33	b32 .. b1
‘E0'	IC Mfg code	RFU set to ‘0’	IC manufacturer serial number

The tag Tag ID shall comprise

- The 8 MSB bits defined as 'E0',
- The IC manufacturer code of 8 bits according to ISO/IEC 7816-6, and
- A unique serial number of 48 bits assigned by the IC manufacturer.

6.2.2 Sub-UID

When the Aloha protocol is used, only a part of the Tag ID, called Sub-Tag ID (SUID) is transmitted in most commands and in the tag response during a collision arbitration process. The only exception is the Get_system_information command that returns the complete Tag ID of 64 bits. See 6.8.7.

The SUID consists of 40 bits: the 8 bits manufacturer code followed by the 32 LSBs of the serial number.

The 16 MSBs (bits 33 to 48) of the serial number shall be set to 0, as bits 33 to 48 are ignored in the SUID.

The mapping of the 64 bit Tag ID to the transmitted 40 bits and back is described in Figure 12.

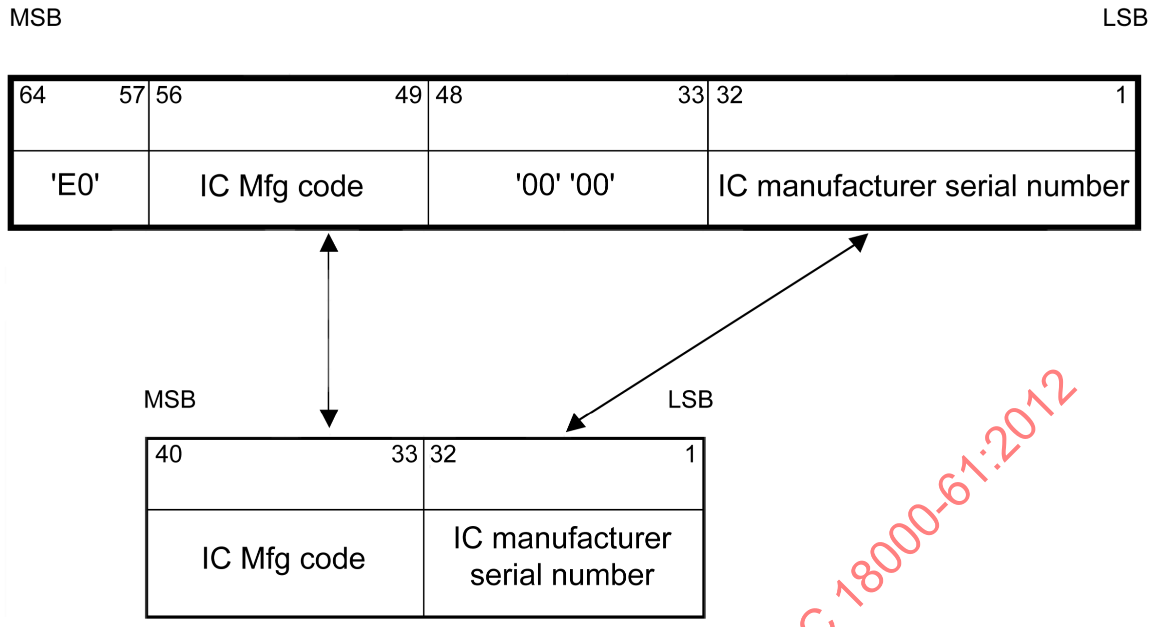


Figure 12 — Tag ID/SUID mapping from 64 to 40

The Interrogator shall use the 64-bit format specified in 6.2.1 when exchanging the Tag ID with the application. It shall perform the required mapping described in Figure 12.

6.2.3 Application family identifier

An Application Family Identifier (AFI) may optionally be used to group select tags. The AFI shall comprise a byte (8 bits). The AFI is a parameter in the Init_round command, or the first 8 bits in the selection mask of the Begin_round command.

When the AFI is used in the Init_round or the Begin_round command it shall conform to the format specified in ISO/IEC 15961 and ISO/IEC 15962 and shown in Table 15.

When an arbitration sequence is initiated using the Init_round or Begin_round command, the AFI parameter shall contain the value of either 0x00 in which case all tags shall respond or; a specific 8 bit value, in which case only those tags which have a matching value shall respond.

Table 15 — AFI/ASF coding

AFI	Meaning tags respond from	Examples / note
‘00’	All families respond	No applicative preselection
‘XY’	Tags match ‘XY’ respond	

NOTE Table 15 is taken from ISO/IEC 15961 and ISO/IEC 15962. Users should refer to the latest version of ISO/IEC 15961 and ISO/IEC 15962, for their reference.

6.2.4 Data storage format identifier (DSFID)

The Data storage format identifier indicates how the data is structured in the tag memory.

It may be programmed and locked by the respective commands. It is coded using one byte. It provides knowledge of the logical organisation of the data.

Its coding shall conform to ISO/IEC 15962.

Where the tag does not support programming of DSFID, it shall respond with the value of DSFID being set to zero ('00').

6.3 Protocol elements

6.3.1 Tag memory organisation

The commands specified in this part of ISO/IEC 18000 assume that the physical memory is organised in blocks of a fixed size.

Up to 256 blocks can be addressed, block sizes can be up to 256 bits, which leads to a maximum memory capacity of up to 8 kbytes (64 kbits).

NOTE The command structure allows for extension of the maximum memory capacity, if required, in revisions of this part of ISO/IEC 18000.

The commands described in this part of ISO/IEC 18000 allow access (read and write) by block(s). There is no implicit or explicit restriction regarding other access methods (e.g., by byte or by logical object in revisions of this part of ISO/IEC 18000, or in custom commands).

6.3.2 Support of battery assisted passive (BAP) tags

This part of ISO/IEC 18000 makes provision for the support of battery assisted passive (BAP) tags.

In normal operation, there is no functional difference between passive and BAP tags.

The mechanisms provided to handle the differences in performance and maintenance of BAP tags and passive tags are as follows:

a) The type of tag and its sensitivity level are returned in the answer to the Get_system_information command. See 6.8.7.

b) The type of tag (battery/no battery) and the battery status is returned in the tag answer during the collision arbitration sequence. See 6.7.5.

6.3.3 Block lock status

The block lock status is returned by the tag as a parameter in the response to an Interrogator command as specified in 6.8.16. The block lock status is coded using two bits. Table 16 shows the 2 bits used for each block diagrammatically, with the actual implementation left to the IC designer.

The block lock status is an element of the protocol. There is no implicit or explicit assumption that the 2 bits are actually implemented in the physical memory structure of the tag.

User locking is performed by the Lock_blocks command.

Factory locking may be performed by a proprietary command.

Table 16 — Block lock status

Bit	Flag name	Value	Description
b1	User_lock_flag	0	Not user locked
		1	User locked
b2	Factory_lock_flag	0	Not factory locked
		1	Factory locked

6.3.4 Tag signature

The tag signature comprises 4 bits. It is used in the Aloha collision arbitration mechanism.

The purpose of the tag signature is to provide a transient session identity that gives differentiation between tags that have answered in the same slot.

The signature is designed to avoid inadvertent isolation, selection or acknowledgement of a tag masked by another tag due to weak signal masking. The tag may generate its signature by a number of mechanisms. For example, by means of a 4 bit pseudo-random number generator, or; by using a part of the tag's Tag ID, or; tag's CRC or; by having a circuit that measures the tag's excitation level [TEL]. The means by which the signature is generated is left to the manufacturer and is not specified in this part of ISO/IEC 18000.

During the collision arbitration process, the tag transmits its signature along with its data or SUID. The Interrogator then uses the signature in its communication with the tag by including it in transmitted commands so that only the tag with a matching signature will be responsive to those commands.

On receiving a command requiring a signature, the tag shall test the received signature against the one it has generated and sent. The tag shall behave in accordance with the state diagram, see Figure 13, and the respective state transition tables. For the Next_slot command see Table 30 and for the Standby_round command see 6.7.3.

6.4 Protocol description

6.4.1 Protocol concept

The transmission protocol defines the mechanism for exchanging commands and data between the Interrogator and the tag, in both directions.

It is based on the concept of "Interrogator-Talks-First".

This means that the tag shall wait to receive a correctly decoded command from the Interrogator before transmitting.

a) The protocol is based on an exchange of

- A command from the Interrogator to the tag, and
- A response from the tag(s) to the Interrogator.

The conditions under which the tag sends a response are defined in 6.7 and 6.8.

b) Each command and each response are contained in a frame. Each command consists of the following fields:

- SOF,
- RFU field (reserved for protocol extension),

- A command code,
- Parameters (or RFU),
- CRC-5,
- Optional parameter fields, depending on the command,
- Application data fields,
- CRC-16, and
- EOF.

c) Each response consists of the following fields:

- SOF,
- Flags,
- Mandatory and optional parameter fields, depending on the command,
- Application data fields,
- CRC-16, and
- EOF.

d) The protocol is bit-oriented. A single field is transmitted most significant bit (MSB) first.

e) A multiple-byte field is transmitted most significant byte (MSByte) first, each byte is transmitted most significant bit (MSB) first.

f) The setting of the flags indicates the presence of the optional fields. When the flag is set (to one), the field is present. When the flag is reset (to zero), the field is absent.

g) RFU flags shall be set to zero (0).

6.4.2 Command format

6.4.2.1 Type A command format general

There are two forms of commands, short commands of 16 bits, and longer.

6.4.2.2 Short command format

A short command consists of the fields defined in Table 17.

Table 17 — Short command format

SOF	RFU	Command code	Parameters/Flags	CRC-5	EOF
	1 bit	6 bits	4 bits	5 bits	

6.4.2.3 Long command format

A long command consists of the fields defined in Table 18.

Table 18 — Long command format

SOF	RFU	Command code	Parameters or flags	CRC-5	SUID (optional)	Data	Data (optional)	CRC-16	EOF
	1 bit	6 bits	4 bits	5 bits	40 bits	8 bits	8 to n bits	16 bits	

The upper value of the optional data field length is specified as n, where n = m+8, and where m = the number of bits to program in the tag's memory. Currently, the value of m = 32 is used, but the protocol allows m to be up to 256.

6.4.3 Command flags

6.4.3.1 Command flags

There is only one command flag, and if present is the SUID flag.

6.4.3.2 SUID flag

6.4.3.2.1 SUID flag general

The SUID flag is contained in the most significant bit position of the command parameter field. Its function is modified by the particular command. Refer to Table 21.

6.4.3.2.2 SUID flag and inventory commands

Inventory commands are:

- Init_round,
- Init_round_all,
- Begin_Round,
- New_Round,
- Close_Slot, and
- Next_Slot.

The first four of the six inventory commands are used for round Initialisation. These four commands can select between receiving the first page of user memory or the tag's SUID as the data message received as part of the inventory process.

If the SUID flag is not set ('0'), the tag shall return the first page of user memory. It shall not include the SUID in its response. (A page of user memory may be as little as 0 bits, but if user memory is present it must be a multiple of 4 bytes (32 bits) depending on the specific implementation.)

If the SUID flag is set ('1'), the tag shall return the SUID. See 6.8.2 and 6.7.5

6.4.3.2.3 SUID flag and non-inventory commands

If the SUID flag is present in the Select, Read_blocks, Get_system_information, Write_block, Write_multiple_blocks, Lock_blocks, Write_AFI, Lock_AFI, Write_DSFID, Lock_DSFID or Get_block_lock_status, then only that tag addressed by the SUID will process the command.

When the SUID flag is set (‘1’), the command shall contain the SUID of the addressed tag. On receiving a command with the SUID flag set, the tag shall compare the SUID contained in the command with its SUID. If the SUID matches, the tag shall execute the command. If the SUID does not match, the tag shall ignore the command.

When the SUID flag is not set (‘0’), the command shall not contain the SUID. On receiving a command the tag shall respond to the command if it is in the select state. (i.e. it was previously selected by means of a Standby_round command with valid signature, or by means of the *Select* command.) If the tag is not in the select state it shall ignore the command.

6.4.4 Round size

The round size is coded using 3 bits according to Table 19.

Table 19 — Round size coding

Value	Bit coding		Round size
	MSB	LSB	
'0'	0 0 0		1
'1'	0 0 1		8
'2'	0 1 0		16
'3'	0 1 1		32
'4'	1 0 0		64
'5'	1 0 1		128
'6'	1 1 0		256
'7'	1 1 1		1024

6.4.5 Command code definition and structure

The size of the command code is 6 bits.

6.4.6 Command classes

6.4.6.1 Command classes general

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

All commands defined in this International Standard are either mandatory or optional, as specified in the definition of the command itself. Custom or proprietary commands are vendor-defined. The four sets of commands are shown in Table 20.

Table 20 — Command classes

Code	Class	Number of possible codes
’00’, 02’, ‘04’, ’06’, ‘0A’, and '0C' – ‘0F'	Mandatory	10
’01’, ‘03’, ‘05’, ’07’, ‘08’, ‘09’, '0B' and ‘10’ – '27', '38', '39'	Optional	32
'28' – '37'	Custom	16
'3A' – '3F'	Proprietary	6

All tags with the same IC manufacturer code and same IC version number shall behave the same.

6.4.6.2 Mandatory

Conformant tags shall support all mandatory commands.

Conformant Interrogators shall support all mandatory commands.

The mandatory command codes are ‘02’, ‘04’, ‘06’ and ‘0A’. Command codes ‘00’ and ‘0C’ to '0F', are RFU.

6.4.6.3 Optional

Tags may or may not support optional commands.

Interrogators may or may not support optional commands.

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

The optional command codes are ‘01’, ‘03’, ’05’, ’07’, ‘08’, ‘09’, '0B', '10' to '27', '38' and '39'.

If a tag in the selected state does not support the optional command it shall return the error code “command is not supported”, see 6.4.8.2.

If the command specifies an SUID and the tag with the matching SUID does not support the optional command it shall return the error code “command is not supported”, see 6.4.8.2.

All other tags shall remain silent.

6.4.6.4 Custom

Custom commands are not specified in this document.

An Interrogator shall use a custom command only in accordance with the specifications of the IC manufacturer identified in the tag ID.

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

The custom command codes range from '28' to '37'.

Tags may support custom commands, at their option, to implement manufacturer specific functions. The function of flags (including reserved bits) shall not be modified. The only fields that can be customised are the parameter and the data fields.

Any custom command contains as its first parameter the IC manufacturer code. This allows IC manufacturers to implement custom commands without risking duplication of command codes and thus misinterpretation.

A tag in the selected state that does not support the Custom command shall return the error code “command is not supported”, see 6.4.8.2.

If the command specifies an SUID and the tag with the matching SUID, does not support the Custom command, it shall return the error code “command is not supported”, see 6.4.8.2.

All other tags shall remain silent.

6.4.6.5 Proprietary

Proprietary commands are not specified in this document.

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

The proprietary command codes range from '3A' to '3F'. A proprietary command shall not solely duplicate the functionality of any mandatory or optional command defined in this part of ISO/IEC 18000 by a different method.

These commands are used by IC and tag manufacturers for various purposes such as tests, programming of system information, etc. They are not specified in this part of ISO/IEC 18000. The IC manufacturer may at its option document them or not.

6.4.7 Command codes and CRC

All mandatory and optional commands are mandatory for a Interrogator implementation, although the Interrogator may not be set-up to provide them all.

This part of ISO/IEC 18000 currently defines 4 mandatory tag commands, Next_slot, Standby_round, Reset_to_ready and Init_round_all, and 6 RFU mandatory commands. All commands are summarised in Table 21, with the 4 mandatory tag commands being highlighted in bold.

Table 21 — Command codes

Command 6 bits	**Op code**	**Parameter / flags 4 bits**		**CRC-5**	**Extended parameters**				**CRC-16**	**Comments**
RFU	00									
Init_round	01	SUID 1 bit	Round size 3 bits	5 bits	AFI 8 bits				16 bits	SUID = 0 tag responds with first page of user data SUID = 1 tag responds with SUID If AFI field = 0x00 (HEX), all tags respond, else if AFI is other value, only tags with matching AFI respond

Command 6 bits	**Op code**	**Parameter / flags 4 bits**		**CRC-5**	**Extended parameters**				**CRC-16**	**Comments**
Next_slot	**02**	**Signature 4 bits**		**5 bits**	**none**					The signature may be one of 16 values between 0x00 and 0x0F as transmitted by the tag in its last transmission.
Close_slot	03	Null		5 bits	none					
Standby_round	**04**	**Signature 4 bits**		**5 bits**	**none**					**The signature may be one of 15 values between 0x01 (HEX) and 0x0F (HEX) as transmitted by the tag in its last transmission.**
New_round	05	SUID 1 bit	Round size 3 bits	5 bits	none					
Reset_to_ready	**06**	**Null**		**5 bits**	**none**					
Select	07	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits				16 bits	
Read_blocks	08	SUID=0 1 bit	Null 3 bits	5 bits	First Block 8 bits	Number of blocks 8 bits			16 bits	
Read_blocks	08	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits	First Block 8 bits	Number of blocks 8 bits		16 bits	
Get_system _information	09	SUID=0 1 bit	Null 3 bits	5 bits						
Get_system _information	09	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits				16 bits	
Init_round_all	**0A**	**SUID 1 bit**	**Round size 3 bits**	**5 bits**						**SUID = 0 tag responds with first page of user data** **SUID = 1 tag responds with SUID**
Begin_round	0B	SUID=0	Round size 3 bits	5 bits	Mask Length 8 bits	Mask n bits				n=0 to 255
RFU	0C to 0F									
Write_block	10	SUID=0 1 bit	Null 3 bits	5 bits	Block number 8 bits	Data n bits			16 bits	n is currently 32 bits, but can be defined up to 256 bits

Command 6 bits	Op code	Parameter / flags 4 bits		CRC-5	Extended parameters				CRC-16	Comments
Write_block	10	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits	Block number 8 bits	Data n bits		16 bits	n is currently 32 bits, but can be defined up to 256 bits
Write_multiple_blocks	11	SUID=0 1 bit	Null 3 bits	5 bits	First Block 8 bits	Number of blocks 8 bits	Data n bits		16 bits	n is currently 32 bits, but can be defined up to 256 bits B is the number of blocks, with valid numbers being 1 to 256.
Write_multiple_blocks	11	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits	First Block 8 bits	Number of blocks 8 bits	Data n bits	16 bits	B is the number of blocks, with valid numbers being 1 to 256
Lock_blocks	12	SUID=0 1 bit	Lock Block number 3 bits	5 bits	Lock block data 32 bits				16 bits	
Lock_blocks	12	SUID=1 1 bit	Lock Block number 3 bits	5 bits	SUID 40 bits	Lock block data 32 bits			16 bits	
Write_AFI	13	SUID=0 1 bit	Null 3 bits	5 bits	AFI 8 bits				16 bits	
Write_AFI	13	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits	AFI 8 bits			16 bits	
RFU	0x38	RFU	RFU	5 bits						
Init_Fast_Slot	0x39	SUID 1 bit	Round Size 3 bits	5 bits						SUID=0 tag responds with 1st page of user data; SUID=1 tag responds with SUID
Lock_AFI	14	SUID=0 1 bit	Null 3 bits	5 bits						
Lock_AFI	14	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits				16 bits	
Write_DSFID	15	SUID=0 1 bit	Null 3 bits	5 bits	DSFID 8 bits				16 bits	
Write_DSFID	15	SUID=1 1 bit	Null 3 bits	5 bits	SUID 40 bits	DSFID 8 bits			16 bits	
Lock_DSFID	16	SUID=0 1 bit	Null 3 bits	5 bits						

Command 6 bits	Op code	Parameter / flags 4 bits		CRC-5	Extended parameters				CRC-16	Comments
Lock_DSFID	16	SUID 1 bit	Null 3 bits	5 bits	SUID 40 bits				16 bits	
Get_block_lock_status	17	SUID=0 1 bit	Lock Block number 3 bits	5 bits						
Get_block_lock_status	17	SUID=1 1 bit	Lock Block number 3 bits	5 bits	SUID 40 bits				16 bits	
RFU	18-27									
Custom	28-37									
Proprietary	38-3F									

6.4.8 Response format

6.4.8.1 Response format general

The response from the tag, as shown in Table 22, consists of the following fields:

- Preamble (See 6.1.4.6),
- Flags (See 6.4.8.2),
- One or more parameter fields (Defined in each command),
- Data (Defined in each command), and
- CRC-16 (See 6.1.4.7.4).

Table 22 — General response format

Preamble	Flags	Parameters	Data	CRC-16

6.4.8.2 Response error flags

In a response, two flags, as shown in Table 23, Table 24 and Table 25, indicate how the tag has performed actions and whether corresponding fields are present or not.

Table 23 — Response format when Error_flag is set

Preamble	Flags	Error code	Terminator
	2 bits	Optional 4 bits See Table 26	‘0’ 1 bit

Table 24 — Response format when Error_flag is NOT set, and there is NO appended data.

Preamble	Flags	Terminator
	2 bits	‘0’ 1 bit

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 25 — Response flags b1 to b2 definition

Bit	Flag name	Value	Description
b1	Error_flag	0	No error
		1	Error detected. Error code is in the "Error" field.
b2	RFU	0	Shall be set to 0.

6.4.8.3 Response error code

The error code comprises four bits.

When the tag sets the Error_flag, the error code field shall be included and provides information about the error that occurred. Error codes are specified in Table 26.

If the tag does not support specific error code(s) listed in Table 26, it shall answer with the error code 'F' (“Error with no information given”).

Table 26 — Response error codes

Code	Description
'0'	RFU
'1'	The command is not supported, i.e. the command code is not recognised
'2'	The command is not recognised, for example: a format error occurred
'3'	The specified block is not available (does not exist)
'4'	The specified block is locked and its content cannot be changed
'5'	The specified block was not successfully programmed and/or locked
'6'–'A'	RFU
'B-'E'	Custom command error codes
'F'	Error with no information given or a specific error code is not supported.

6.4.9 Tag states

6.4.9.1 Tag states general

A tag can be in one of the following states:

- RF field off,
- Ready,
- Quiet,
- Selected,
- Round_active, and
- Round_standby.

The transitions between states are shown in Figure 13 and are specified in tables in each command description.

6.4.9.2 RF field off state

The tag is in the RF field off state when it does not receive the required RF field from the Interrogator.

For passive tags, it means that the tag is not powered.

For BAP tags, it means that the level of RF excitation is insufficient to turn on the tag circuits.

6.4.9.3 Ready state

The tag is in the Ready state when it is receiving sufficient energy from the Interrogator to function correctly. It shall process any command where the select flag is not set.

6.4.9.4 Quiet state

When in the Quiet state, the tag shall process any command where the SUID flag is set and the SUID matches.

NOTE A tag will move to the Quiet state after it has been correctly identified within its slot by the Interrogator and takes no further part within the current or subsequent rounds.

6.4.9.5 Selected state

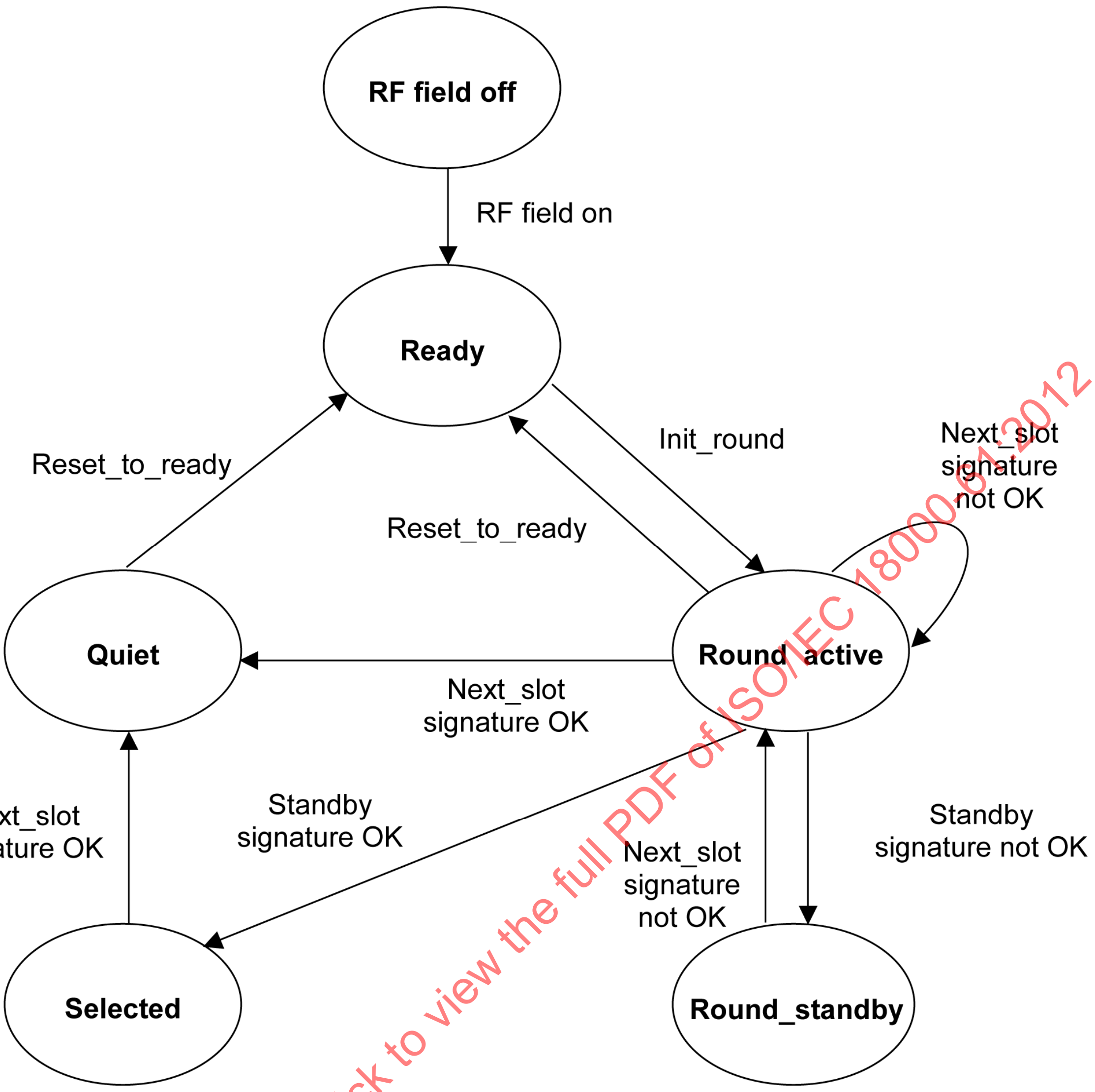
Only a tag in the Selected state shall process commands with the SUID flag set to 0, with the exception of Init_round, Init_round_all, Begin_round and New_round commands, which will be processed even if the SUID flag is set to 1.

6.4.9.6 Round_active state

When in the Round_active state, the tag shall participate in the collision arbitration described in 6.4.10 and 6.4.11.

6.4.9.7 Round_standby state

When in the Round_standby state, the tag shall suspend its participation in the collision arbitration as described in 6.4.10 and 6.4.11.



NOTE This state diagram shows some of the main transitions. State transitions are specified in detail in each command description.

Figure 13 — Tag state transition diagram

6.4.10 Collision arbitration

The purpose of the collision arbitration sequence is to perform a census of the tags present in the Interrogator field and to receive information on the tag capabilities and data contents, all in a single sequence. The information that the tag shall return, is specified by the SUID flag, set in the command from the Interrogator.

The Interrogator is the master of the communication with one or multiple tags.

NOTE This mechanism is also some times referred to as the "wake-up" sequence.

6.4.11 General explanation of the collision arbitration mechanism

The collision arbitration uses a mechanism that allocates tag transmissions into rounds and slots. A round consists of a number of slots. Each slot has a duration long enough for the Interrogator to receive a tag response. The Interrogator determines the actual duration of a slot.

In the absence of an RF field, the tags are in the RF_field off state. When the tags enter the energizing field of an Interrogator, they go through a power on reset sequence and move into the Ready state.

The Interrogator initiates a tag census or collision arbitration process by sending an Init_round command, or the Init_round_all command.

Tags receiving an Init_round command randomly select a slot in which to respond but do not immediately start transmitting. The number of slots in a round referred to as round size, is determined by the Interrogator and signalled to the tag in the Init_round command. The initial round size is predetermined by the user. During the subsequent collision arbitration process the Interrogator dynamically chooses an optimum round size for the next round based on the number of collisions in the round. The number of collisions is a function of the number of tags in the active state present in the Interrogator field and the round size.

On receiving an Init_round command, tags select a slot in which to respond. The selection is determined by a pseudo-random number generator. If the tag has selected the first slot it will wait a pseudo-random time delay equal to a time of between 0 and 7 periods and then transmit its response.

The tag includes its four bit tag signature in its response.

If the tag has selected a slot number greater than one, it will retain its slot number and wait for a further command.

After the Interrogator has sent the Init_round command there are three possible outcomes:

1) The Interrogator does not receive a response because either no tag has selected slot one or the Interrogator has not detected a tag response. It then issues a Close_slot command because it has not received a response.

2) The Interrogator detects a collision between two or more tag responses. Collisions may be detected either as contention from the multiple transmissions or by detecting an invalid CRC. Having verified that there are no tags transmitting, the Interrogator sends a Close_slot command.

3) The Interrogator receives a tag response without error, i.e. with a valid CRC. The Interrogator sends a Next_slot command containing the signature of the tag just received.

When tags that have not transmitted in the current slot receive a Close_slot or Next_slot command, they increment their slot counters by one. When the slot counter equals the slot number previously selected by the tag, the tag transmits according to the rules above, otherwise the tag waits for another command.

When a tag in the active state receives a Close_slot command, it increments its slot counter.

When a tag that has transmitted its data in the current slot receives a Next_slot command, it:

- verifies that the signature in the command matches the signature it sent in its last response, and
- verifies that the Next_slot command has been received within the time window specified in 6.5.5.

If the tag has met these acknowledge conditions it enters the Quiet state. Otherwise, it retains its current state.

The round continues until all slots have been explored.

During a round, the Interrogator can suspend the round by sending a Standby_round command. The tag processes the command in a similar way to a Next_slot command, except that if the acknowledge conditions are met, the tag enters the Selected state. If they are not met, the tag enters the Round_standby state.

NOTE The Round_standby mechanism allows the Interrogator to conduct a dialogue with a selected tag before continuing the round.

A tag, which has not been acknowledged by a next-slot OK or round-standby OK, keeps track of the slot count.

The Interrogator keeps track of the slot count each time it issues a Close_slot or Next_slot command.

When the slot count equals the round-size specified in either of the Init_round or Init_round_all commands, the tag shall select a new slot in which to transmit, select a new random signature and enter a new round.

The detail of the tag state transition is specified for each command in tables detailed below.

Tag manufacturer may optionally configure the tag to power-up in the FST mode. When the tag is configured to power up in FST mode, and enters the energizing field of an Interrogator, it goes through a power on reset sequence and then moves to the Round_Standby state until it receives a Next_Slot, Close_Slot, New_Round or Init_Fast_Slot command, at which time it commences a Fast Slot Mode collision arbitration sequence.

Each slot has a duration at least as long as the duration of a tag preamble. The actual duration of the slot is determined by the tag and is equal to 16 tag bit times. If a tag has selected the current slot in which to transmit its reply, the Slot length is increased for that tag to the duration of a message length so that the tag can send its complete message. In order to prevent other tags (those that have not yet started their replies) from transmitting during the first tag's reply slot, the Interrogator issues a MUTE command to place the tag into the Round_Standy state. After the active tag has finished transmitting its message, and if the Interrogator has successfully read the tag it issues a Next_Slot command synchronously with the tag's signature. If the tag message was not successfully read then the Interrogator issues a Close_Slot command, which will cause all the tags currently in the Round_Standby state to re-enter the Round_Active state.

The number of slots in a round, referred to as round size, is determined by the Interrogator and is signalled to the tag in the Init_Fast_Slot or New_Round command. In the FST mode the tag elects a default roundsize of 16, which may be overridden by an Interrogator command, however the FST mode is able to operate without any round initialising command. During the subsequent collision arbitration process the Interrogator dynamically chooses an optimum round size for the following rounds based on the number of collisions and/or unproductive time in a round. The number of collisions is a function of the number of tags in the Round_Active state present in the Interrogator field and the current round size. The Interrogator signals a change in round size to tags by sending a New_Round command containing the required round size.

The tag on entering the Round_Active State or on re-entering the Round_Active state having completed a round, selects a pseudo slot at random in which to reply. Pseudo slots are equal to tag preamble in duration. If the tag has selected the first pseudo slot, it will transmit immediately, if not it will hold off until it has reached the selected pseudo-slot and then transmit.

On receiving and recognizing a valid tag transmission preamble, the Interrogator sends a MUTE command (SOF), which tells all tags that have not yet started transmitting, to move to the Round_Standy state. When the Interrogator receives the tag Reply without error, it sends a Next_Slot command containing the signature of the tag that it has just received.

When the tag has reached the end of a round, it will self-trigger a new round, randomly select a new slot in which to transmit and it will transmit its identity or data when it reaches the selected slot. The process continues until the tag has been successfully read and acknowledged by a valid Next_Slot command or removed from the RF energizing field.

6.5 Timing specifications

6.5.1 Timing specifications general

The Interrogator and the tag shall comply with the following timing specifications.

6.5.2 Tag state storage

In the case of absent or insufficient energizing field, the tag shall retain its state for at least 300µs.

In addition, if the tag is in the Quiet state, it shall retain its Quiet state for at least 2s.

NOTE Implementation of the Quiet state storage may imply that the tag will retain this condition during a time greater than 2s, up to several minutes in low temperature conditions. The Reset_to_ready command allows an exit of the Quiet state in these circumstances.

6.5.3 Forward link to return link handover

After the reception of an EOF from the Interrogator, the tag shall wait a time Trs, that starts from the negative going edge of the EOF. See Figure 14 and Table 27.

Table 27 — Forward link to return link handover timing

Minimum	Maximum
150µs	1150µs

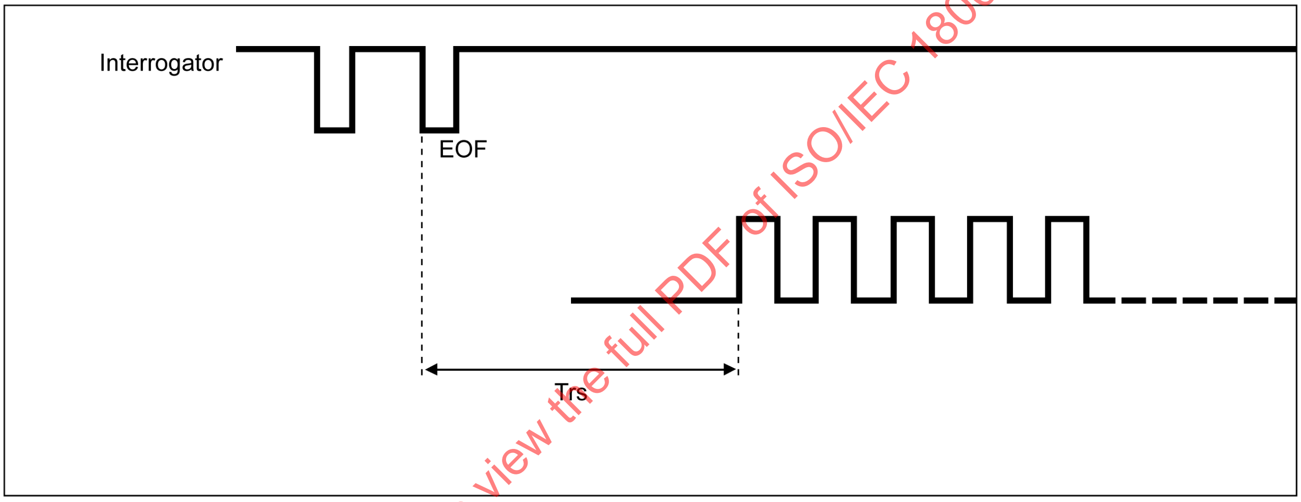


Figure 14 — Forward link to return link handover

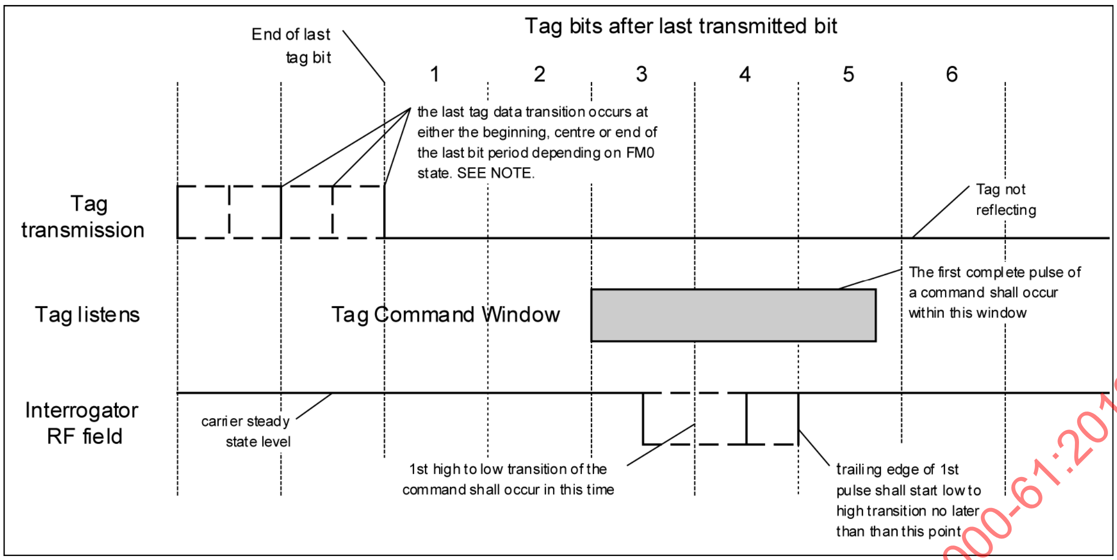
6.5.4 Return link to forward link handover

The protocol is half-duplex. The tag is not required to receive and decode properly a command from the Interrogator while it is transmitting.

6.5.5 Acknowledgement time window

6.5.5.1 Acknowledgement time window general

The purpose of the acknowledgement time window, shown in Figure 15, is to ensure that only those tags that receive a synchronised command starting within a command window will respond to those commands. This reduces considerably the possibility that a tag may be incorrectly acknowledged during the collision arbitration or census process.


End of last tag bit
Tag bits after last transmitted bit
1
2
3
4
5
6
the last tag data transition occurs at either the beginning, centre or end of the last bit period depending on FM0 state. SEE NOTE.
Tag not reflecting
Tag transmission
Tag listens
Tag Command Window
The first complete pulse of a command shall occur within this window
Interrogator RF field
carrier steady state level
1st high to low transition of the command shall occur in this time
trailing edge of 1st pulse shall start low to high transition no later than than this point

NOTE The four cases of the FM0 state are detailed in Figure 16.

Figure 15 — Acknowledgement time window

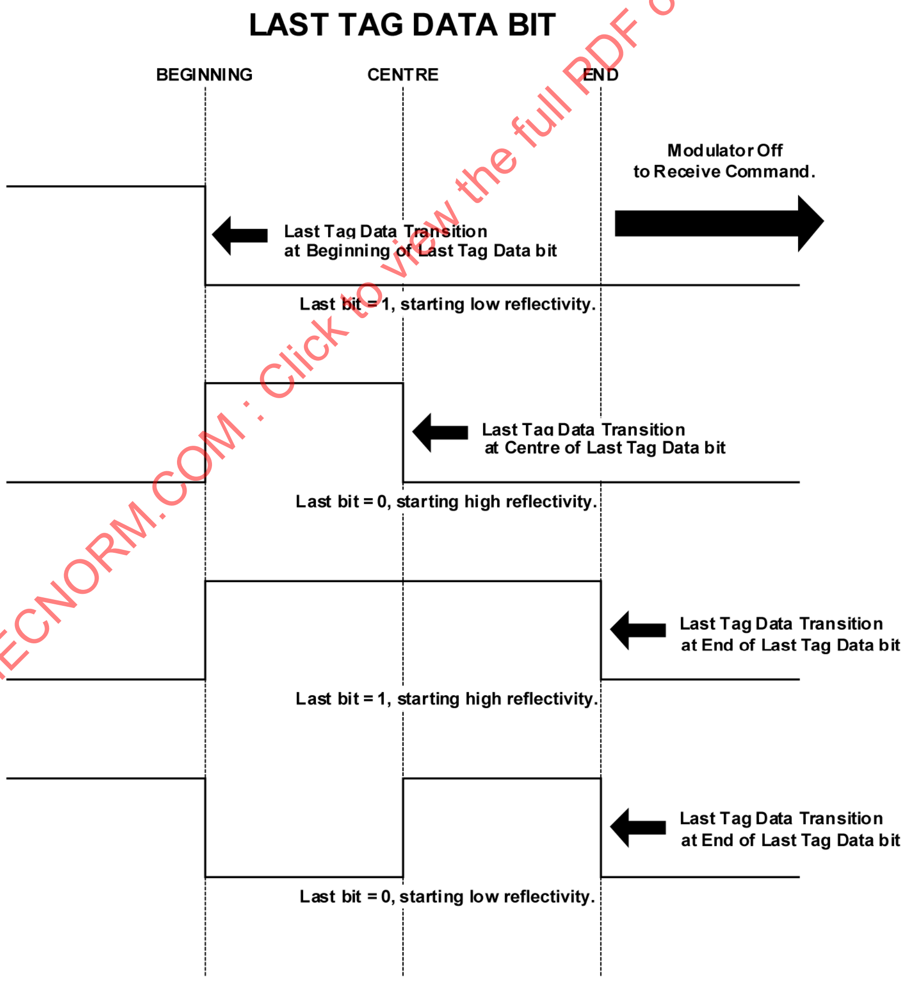

LAST TAG DATA BIT
BEGINNING
CENTRE
END
Modulator Off to Receive Command.
Last Tag Data Transition at Beginning of Last Tag Data bit
Last bit = 1, starting low reflectivity.
Last Tag Data Transition at Centre of Last Tag Data bit
Last bit = 0, starting high reflectivity.
Last Tag Data Transition at End of Last Tag Data bit
Last bit = 1, starting high reflectivity.
Last Tag Data Transition at End of Last Tag Data bit
Last bit = 0, starting low reflectivity.

Figure 16 — Details of last FM0 tag data bit

6.5.5.2 Interrogator

The Interrogator shall send a Next_slot or a Standby_round command (that acknowledges the answer of a specific tag) within a specific time window. The command window shall be open from the boundary between bits two and three, for a duration of 2.75 tag bit periods, see Figure 15.

The Interrogator shall measure the tag clock frequency so that it can start a command in the time window.

The Interrogator shall not modulate the carrier for at least 3 tag bit periods prior to sending the command start pulse.

The Interrogator shall send the first negative going edge of the command frame within bit period ‘4’.

6.5.5.3 Tag

When a tag, in the Round active state (and not in the FST mode), has just responded in the current slot and receives a Next_slot command, which was not received within the window specified in 6.5.5, the tag shall not move to the quiet state, but shall remain in the Round active state and shall increment its slot counter as specified in 6.7.2 and Table 30.

When a tag in the Round active state, has just responded in the current slot and receives a Standby_round command, which was not received within the window specified in 6.5.5, the tag shall not move to the selected state, but shall transition to the Round_standby state as specified in 6.7.3 and Table 32.

6.6 Command format examples

For examples of the bit pattern for a short command, see 6.7.2 and for a long command see 6.8.6.

6.7 Mandatory commands

6.7.1 Mandatory commands general

There are four mandatory commands, command codes 02, 04, 06 and 0A. Command codes 00 and 0B to 0F are mandatory commands reserved for future use.

Interrogator shall implement all mandatory commands.

Tags shall implement all mandatory commands.

6.7.2 Next_slot

Command code = '02'

The Next_slot command has two functions:

- It acknowledges the tag that has been identified, or
- It instructs all tags in the Round_active state to switch to the next slot by incrementing their slot counter.

The command contains:

- The Next_slot command code, and
- The tag signature.

No flags are used by this command.

Table 28 shows the Next_slot command format.

Table 28 — Next_slot command format

Protocol extension	Next_slot	Tag signature	CRC-5
1 bit	6 bits	4 bits	5 bits

Table 29 is an example of a short command. The complete bit pattern for the Next_slot command, with a tag signature of 6, is as follows:

Table 29 — Example of a short command, Next_slot

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	PE	Command Code Next slot =02						Tag Signature (egg = 6)				CRC-5 which for this case is 0					
SOF	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	EOF

There is no direct tag response to the Next_slot command. If the tag is in the Round_active state, it shall switch to the next slot by incrementing their slot counter and send its response if its slot counter matches the chosen slot.

The tag shall perform the actions specified in Table 30. It may either enter the Quiet state or switch to the next slot by incrementing its slot counter.

Table 30 — Tag state transition for Next_slot

Command: Next_slot			
Current state	**Criteria**	**Action**	**New state**
Ready	none	none	Ready
Quiet	none	none	Quiet
Selected	none	none	Quiet
Round_active	Tag has answered in previous slot, AND Signature matches AND Next_slot was received in the acknowledgement time window. See 6.5.5	none	Quiet
	Tag has not answered in previous slot, OR Signature does not match OR Next_slot was not received in the acknowledgement time window. See 6.5.5	The tag shall increment its slot counter and send its response if slot counter matches the chosen slot.	Round_active
Round_standby	none	The tag shall increment its slot counter and send its response if slot counter matches the chosen slot.	Round_active

In the Fast Slot Mode the Next_Slot command instructs all tags in the Round_Standby state to switch to the Active state.

6.7.3 Standby_round

Command code = '04'

The Standby_round command has two functions:

- It acknowledges a valid response from a tag and instructs this tag to enter the Selected state if the tag supports the optional Selected state (individual commands can then be sent to this tag such as Read and Write), or
- It instructs all other tags in the Round_active state to enter the Round_standby state. For these tags, their participation in the round is suspended. tags that do not support the optional Selected state will move from the Round_active state to the Round_standby state irrespective of whether the Standby_round command acknowledged a valid response from the tag or not. The round will be resumed by a Next_slot command, or a Close_slot command. In addition, a new round will be initiated by tags in the Round_standby state by a New_round command, an Init_round command or an Init_round_all command.

The command contains:

The Standby_round command code, and

The tag signature.

No flags are used by this command.

Table 31 shows the Standby_round command format.

Table 31 — Standby_round command format

Protocol extension	**Standby_round**	**Tag signature**	**CRC-5**
1 bit	6 bits	4 bits	5 bits

Table 32 — Tag state transition for Standby_round

Command: Standby_round			
Current state	**Criteria**	**Action**	**New state**
Ready	None	None	Ready
Quiet	None	None	Quiet
Selected	None	None	Quiet
Round_active	Tag has answered in previous slot, AND Signature matches, AND Next_slot was received in the acknowledgement time window. See 7.5.5, AND Tag supports the optional SELECTED state.	None	Selected
	Tag has not answered in previous slot, OR Signature does not match OR Next_slot was not received in the acknowledgement time window. See 7.5.5, OR Tag does not support the optional SELECTED state.	None	Round_standby
Round_standby	None	None	Round_standby

6.7.4 Reset_to_ready

Command code = '06'

The Reset_to_ready command has one function:

- It instructs all tags to enter the ready state.

The command contains:

- The Reset_to_ready command code, and
- Four RFU bits. These bits shall be set to zero.

On receiving the Reset_to_ready command, the tag shall reset to zero all stored parameters such as the AFI and the slot counter.

This command has the same effect as removing the tag from the RF field for an extended period of time long enough to reset the "quiet" memory and then returning it to the RF field.

Table 33 shows the Reset_to_ready command format.

Table 33 — Reset_to_ready command format

Protocol extension	Reset_to_ready	RFU	CRC-5
1 bit	6 bits	4 bits	5 bits

There is no response to the Reset_to_ready command.

The tag shall perform the actions specified in Table 34.

Table 34 — Tag state transition for Reset_to_ready

Command: Reset_to_ready			
Current state	**Criteria**	**Action**	**New state**
Ready	None	None	Ready
Quiet	None	None	Ready
Selected	None	None	Ready
Round_active	None	None	Ready
Round_standby	None	None	Ready

6.7.5 Init_round_all

Command code = '0A'

This command is functionally equivalent to the Init_round command, with AFI set to 00.

The command contains:

- The Init_round_all command code of 6 bits,
- The SUID Flag of 1 bit, and
- The Round size of 3 bits.

Table 35 shows the Init_round_all command format.

Table 35 — Init_round_all command format

Protocol extension	Init_round_all	SUID flag	Round size	CRC-5
1 bit	6 bits	1 bit	3 bits	5 bits

The response, as shown in Table 36 or Table 37 shall contain:

- The DSFID if the SUID flag is set in the command,
- The tag signature,

- The tag type (passive or BAP),
- The battery status flags,
- The SUID if the SUID flag is set in the command, and
- The first n bits of the tag memory if the SUID flag is NOT set in the command.
- If the tag detects an error, it shall remain silent.

Table 36 — Init_round_all response format when the SUID flag is NOT set

Preamble	Flags see 6.4.8	Tag type see Table 38	Battery status see Table 39	Signature	Random number See NOTES 1 and 2	First n bits of memory See NOTE 3	CRC-16
	2 bits	1 bit	1 bit	4 bits	8 bits	n = 32, 64, 96 or 128 bits	16 bits

NOTE 1 The purpose of the random number is to increase the probability of detecting a collision between two or more tags whose responses contain the same data. Combined with the 4 bit signature, it represents a 12 bits random number. It is possible that if the SUID flag is NOT set, and the tag returns the first "page" of data (could be 32,64,96 or 128 bits), multiple tags could be programmed with the same memory content. To ensure a collision is detected, the 4 bit signature is extended by 8 bits, giving 12 bits, in which a collision could be detected.

NOTE 2 The DSFID field is not included since in situations where the first "page" of memory is read, it is expected that no further exchange with the tag will take place. The DSFID can nevertheless be determined using other commands, see 6.8.7.

NOTE 3 The tag shall return the first n bits of user data according to its memory size, with a maximum of 128 bits. If there is no user memory, the tag shall return the SUID.

This part of ISO/IEC 18000 does not specify how the 8 bit random number shall be generated (It could for instance be the last 8 significant bits of the Tag ID, or a random number, etc.).

The generation of the signature and the random number shall be independent of each other.

Table 37 — Init_round_all response format when the SUID flag is set

Preamble	Flags see 6.4.8	Tag type see Table 38	Battery status see Table 39	Signature	DSFID	SUID	CRC-16
	2 bits	1 bit	1 bit	4 bits	8 bits	40 bits	16 bits

NOTE The 8 bits random number is not required when an SUID is sent back since the collisions will be detected using the combination of the signature and SUID.

Table 38 shows the Tag type field, while Table 39 shows the Battery Status field.

Table 38 — Tag type (passive or BAP)

Tag type	Meaning
0	Passive Tag (NOT battery assisted)
1	BAP Tag (battery assisted)

Table 39 — Battery status (for BAP Tag)

Battery status	Meaning
0	Battery is low. A passive tag shall set this bit to 0.
1	Battery is good.

The tag shall perform the actions specified in Table 40.

Table 40 — Tag state transition for Init_round_all

Command: Init_round_all			
Current state	**Criteria**	**Action**	**New state**
Ready	None	The tag shall choose the slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
Quiet	None	The tag shall discard the command and remain silent.	Quiet
Selected	None	The tag shall discard the command and transition to the Quiet state.	Quiet
Round_active	None	The tag shall reset the previously chosen slot and chose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
Round_standby	None	The tag shall reset the previously chosen slot and chose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active

NOTE The slots are numbered from 1 to Round_size.

6.8 Optional commands

6.8.1 Optional commands general

The following commands are optionally supported by the tag.

If a tag in the selected state does not support the optional command it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID does not support the optional command it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

All other tags shall remain silent.

6.8.2 Init_round

Command code = '01'

The command contains:

- The Init_round command code of 6 bits,
- The SUID flag of 1 bit,
- The Round size of 3 bits, and
- The AFI of 8 bits.

NOTE A tag that is in the Round_active state at the end of a round shall automatically enter a new round

Table 41 shows the Init_round command format.

Table 41 — Init_round command format

Protocol extension	Init_round	SUID	Round size	CRC-5	AFI	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	16 bits

The response is the same as that specified for the Init_round_all command, see 6.7.5, Table 36 and Table 37.

The tag shall perform the actions specified in Table 42.

Table 42 — Tag state transition for Init_round

Command: Init_round			
Current state	**Criteria**	**Action**	**New state**
Ready	AFI Matches	The tag shall choose the slot in which it will send its response by generating a random number. It shall reset its slot counter to 1. See NOTE.	Round_active
	AFI does not match	Tag ignores the command.	Ready
Quiet	None	The tag shall discard the command and remain silent.	Quiet
Selected	None	The tag shall discard the command and transition to the Quiet state.	Quiet
Round_active	AFI Matches	The tag shall reset the previously chosen slot and choose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
	AFI does not match	The tag shall discard the command and returns to the Ready State.	Ready
Round_standby	AFI Matches	The tag shall reset the previously chosen slot and choose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
	AFI does not match	The tag shall discard the command and returns to the Ready State	Ready

NOTE The slots are numbered from 1 to Round_size.

6.8.3 Close_slot

Command code = '03'

The Close_slot has one function, it instructs all tags in the Round_active state to switch to the next slot by incrementing their slot counter.

In the Fast Slot Mode the Close_Slot command instructs all tags in the Round_Standby state to switch to the Round_Active state.

The Close_slot command shall be sent by the Interrogator when no tag response has been received in the slot or when a collision has been detected. If the Interrogator detects a collision, then the Interrogator must wait until all tags have replied, and the tag Tx/Rx turnaround time has expired, before it can issue the Close_slot command.

The command contains:

- The Close_slot command code, and
- 4 RFU bits (these bits shall be set to 0).

Table 43 shows the Close_slot command format.

Table 43 — Close_slot command format

Protocol extension	Close_slot	RFU	CRC-5
1 bit	6 bits	4 bits	5 bits

There is no direct tag response to the Close_slot command. If the tag is in the Round_active state, it shall switch to the next slot by incrementing its slot counter and send its response if its slot counter matches the chosen slot.

The tag shall perform the actions specified in Table 44.

Table 44 — Tag state transition for Close_slot

Command: Close_slot			
Current state	**Criteria**	**Action**	**New state**
Ready	None	None	Ready
Quiet	None	None	Quiet
Selected	None	None	Quiet
Round_active	Long_Slot Mode	The tag shall increment its slot counter and send its response if slot counter matches chosen slot.	Round_active
	Fast_Slot Mode	The tag will automatically increment its slot counter at internally determined intervals and send its response if slot counter matches chosen slot.	Round_active
Round_standby	Long_Slot Mode	The tag shall increment its slot counter and send its response if slot counter matches chosen slot.	Round_active
	Fast_Slot Mode	The tag will automatically increment its slot counter at internally determined intervals and send its response if slot counter matches chosen slot.	Round_active

6.8.4 New_round

Command code = '05'

The New_round command has two functions:

- It instructs the tags that are neither in the Quiet nor in the Selected state to enter a new round, to reset their slot counters to 1 and to enter the Round_active state (tags in the Ready state will not enter a new round), and
- It instructs the tag in the Selected state to enter the Quiet state (tags in the Ready or Quiet state shall remain in their prior states).

The command contains:

- The New_round command code,
- The SUID Flag, and
- The new round size.

All other parameters (such as the AFI/ASF) shall be the same as in the previous round.

Table 45 shows the New_round command format.

Table 45 — New_round command format

Protocol extension	New_round	SUID Flag	Round size	CRC-5
1 bit	6 bits	1 bit	3 bits	5 bits

Assuming that the SUID flag is the same, the response shall be the same as the response to the previous round, with the exception of the signature, that might be different, for instance if based on the TEL or a random-number.

If the SUID = 0, the tag shall respond with the first page of user data (32,64,96 or 128 bits) as in Table 36.

If the SUID = 1, the tag shall respond with the SUID as in Table 37.

The tag shall perform the actions specified in Table 46.

Table 46 — Tag state transition for New_round

Command: New_round			
Current state	**Criteria**	**Action**	**New state**
Ready	None	None	Ready
Quiet	None	None	Quiet
Selected	None	None	Quiet
Round_active	None	The tag shall reset the previously chosen slot and chose a new slot in which it will send its response by generating a random number.	Round_active
Round_standby	None	The tag shall reset the previously chosen slot and chose a new slot in which it will send its response by generating a random number.	Round_active

6.8.5 Select (by SUID)

Command code = '07'

The Select command has two functions:

- It instructs the tag specified by its SUID to enter the Selected state from any other state (individual commands, (such as Read and Write), can then be sent to this tag), and
- It instructs all tags in the Round_active state with non-matching SUIDs to enter the Round_standby state. For these tags, their participation in the round is suspended. The round will be resumed following a Next_slot command or a Close_slot command or a new round will be initiated following a New_round command, Init_round or Init_round_all command.

On receiving the Select command:

- If the SUID matches the tag SUID, the tag shall enter the selected state and shall send a response, or
- If the SUID does not match the tag SUID, and if the tag is in the Round_active state, the tag shall enter the Round_standby state, and shall not send a response.

Command parameter:

SUID (mandatory)

Table 47 shows the Select command format.

Table 47 — Select command format

Protocol extension	Select	SUID (always set to 1)	RFU	CRC-5	SUID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	16 bits

Response format:

If the tag with the matching SUID does not support the Select command it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the tag with the matching SUID does support the Select command, and if the error flag is set, the response shall be according to Table 23.

If the tag with the matching SUID does support the Select command, and if the error flag is not set, the response shall be according to Table 24.

All other tags shall remain silent.

The tag shall perform the actions specified in Table 48.

Table 48 — Tag state transition for Select

Command: Select			
Current state	**Criteria**	**Action**	**New state**
Ready	The SUID does not match	None	Ready
	The SUID matches	The tag shall send back its response	Selected
Quiet	The SUID does not match	None	Quiet
	The SUID matches	The tag shall send back its response	Selected
Selected	The SUID does not match	None	Quiet
	The SUID matches	The tag shall send back its response	Selected
Round_active	The SUID does not match	None	Round_standby
	The SUID matches	The tag shall send back its response	Selected
Round_standby	The SUID does not match	None	Round_standby
	The SUID matches	The tag shall send back its response	Selected

6.8.6 Read_blocks

Command code = '08'

On receiving the Read_blocks command, the tag shall respond with the requested data and the block lock status.

The blocks are numbered from '00' to 'FF' (0 to 255).

The number of blocks in the command is one less than the number of blocks that the tag shall return in its response.

For example, a value of '06' in the “Number of blocks” field is a request to read 7 blocks. A value of '00' is a request to read a single block.

There are two formats of the Read_blocks command, one specifying the SUID, see Table 49, and the other that does not specify the SUID, see Table 50, but is used to read blocks from the tag in the selected state.

Table 49 — Read_blocks command format SUID specified

Protocol extension	Read_blocks	SUID flag = 1	RFU	CRC-5	SUID	First block number	Number of blocks	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	8 bits	8 bits	16 bits

Table 50 — Read_blocks command format SUID NOT specified

Protocol extension	Read_ blocks	SUID flag = 0	RFU	CRC-5	First block number	Number of blocks	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	8 bits	16 bits

Command parameter:

(Optional) SUID

First block number

Number of blocks

Table 51 gives an example of a long command showing the bit pattern for the Read_blocks command, with SUID flag = 1, SUID = FEDCBA4321 (hex), First Block = 27 (Hex), Number of blocks to read is 3. Total message length = 88 bits, plus SOF and EOF:

Table 51 — Example of long command, Read_blocks with SUID.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
	PE	Command Code Read Blocks =08						*	NULL			CRC-5, in this example is 08				
SOF	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0

* SUID Flag = 1 (set)

17	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
SUID first 16 bits (40 to 25), In this example the complete SUID is FEDCBA4321 (Hex)															
1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0

33	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**
SUID next 16 bits (24 to 9)															
1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1

49	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**	**64**
SUID last 8 bits (8 to 1)								First Block number, in this example = 27 hex							
0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1

65	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**
Number of Blocks, in this example = 3								First 8 bits of 16-bit CRC-16 = 70							
0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0

81	**82**	**83**	**84**	**85**	**86**	**87**	**88**	
Last 8 bits of 16-bit CRC-16 = F9								
1	1	1	1	1	0	0	1	EOF

Response format:

If a tag in the selected state does not support the Read_blocks command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If a tag in the selected state does support the Read_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If a tag in the selected state does support the Read_blocks command, and if the error flag is NOT set, the response shall be according to Table 52.

If the command specifies an SUID and the tag with the matching SUID does not support the Read_blocks command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID, does support the Read_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If the command specifies an SUID and the tag with the matching SUID, does support the Read_blocks command, and if the error flag is NOT set, the response shall be according to Table 52.

All other tags shall remain silent.

Table 52 — Read_blocks response format when Error_flag is NOT set

Preamble	Flags see 6.4.8	Block security status	Data	CRC-16
	2 bits	2 bits	Block length	16 bits
		Repeated as needed		

The block security status field (2 bits) is defined in Table 15.

The tag response shall be ordered such that the most significant bit of the most significant byte of data is transmitted first.

Response parameter:

Error_flag (and Error code if Error_flag is set), see 6.4.8.2

if Error_flag is not set (the following order shall be respected in the tag response)

Block security status N

Block value N

Block security status N+1

Block value N+1

etc…

where N is the first requested (and returned) block.

The tag shall perform the actions specified in Table 53.

Table 53 — Tag state transition for Read_blocks

Command: Read_blocks			
Current state	**Criteria**	**Action**	**New state**
Ready	The SUID does not match	none	Ready
	The SUID matches	The tag shall send back its response	
Quiet	The SUID does not match	none	Quiet
	The SUID matches	The tag shall send back its response	
Selected	The SUID flag = 0	The tag shall send back its response	Selected
	The SUID flag = 1 and the SUID matches	The tag shall send back its response	
	The SUID flag = 1 and the SUID does not match	none	
Round_active	The SUID does not match	none	Round_active
	The SUID matches	The tag shall send back its response	
Round_standby	The SUID does not match	none	Round_standby
	The SUID matches	The tag shall send back its response	

6.8.7 Get_system_information

Command code = '09'

This command allows for retrieving the system information value from the tag.

There are two formats of the Get_system_information command, one specifying the SUID, shown in Table 54 and the other that does not specify the SUID, as shown in Table 55, but is used to get system information from the tag in the selected state.

Table 54 — Get_system_information command format with SUID

Protocol extension	Get_system_info	SUID flag = 1	RFU	CRC-5	SUID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	16 bits

Table 55 — Get_system_information command format with no SUID

Protocol extension	Get_system_info	SUID flag = 0	RFU	CRC-5
1 bit	6 bits	1 bit	3 bits	5 bits

Command parameter:

(Optional) SUID

Response parameter:

If a tag in the selected state does not support the Get_system_information command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If a tag in the selected state does support the Get_system_information command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If a tag in the selected state does support the Get_system_information command, and if the error flag is NOT set, the response shall be according to Table 56.

If the command specifies an SUID and the tag with the matching SUID does not support the Get_system_information command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID does support the Get_system_information command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If the command specifies an SUID and the tag with the matching SUID, does support the Get_system_information command, and if the error flag is NOT set, the response shall be according to Table 56.

All other tags shall remain silent.

Table 56 — Get_system_information response format when Error_flag is NOT set

Preamble	**Flags** **see 6.4.8**	**Tag type** **see Table 38**	**Battery status** **see Table 39**	**Signature**	**Tag ID** **see NOTE 1**	**DSFID** **see NOTE 2**	**AFI** **see NOTE 3**	**System information** **see Table 57**	**CRC-16**
	2 bits	1 bit	1 bit	4 bits	64 bits	8 bits	8 bits	32 bits	16 bits

NOTE 1 The complete Tag ID on 64 bits shall be returned.

NOTE 2 The field DSFID may be present if and only if supported by the tag

NOTE 3 The field AFI may be present if and only if supported by the tag

Response format:

Error_flag (and Error code if Error_flag is set), see 6.4.8.2

If Error_flag is not set, the response contains:

- Tag ID (mandatory),
- DSFID if supported and present,
- AFI if supported and present, and
- System Information, see Table 57.

Table 57 — System information definition

Bit	Flag name	Value	Description
b1	DSFID	0	DSFID is not supported. DSFID field is not present
		1	DSFID is supported. DSFID field is present
b2	AFI	0	AFI is not supported. AFI field is not present
		1	AFI is supported. AFI field is present
b3	Tag memory size	0	Information on tag memory size is not supported. Memory size field is not present.
		1	Information on tag memory size is supported. Memory size field is present.
b4	IC reference	0	Information on IC reference is not supported. IC reference field is not present.
		1	Information on IC reference is supported. IC reference field is present.
b5-b6	Tag sensitivity	00	Tag sensitivity is undetermined.
		01	Tag sensitivity is S1. See Table 58
		10	Tag sensitivity is S2. See Table 58
		11	Tag sensitivity is S3. See Table 58
b7-b8	Tag type	00	Tag is passive backscatter, not battery assisted.
		01	Tag is passive backscatter and battery assisted.
		10	Tag is active.
		11	RFU
b9	RFU	0	RFU
b10	RFU	0	RFU
b11-b16	IC reference		See b4 above, IC Reference is as defined by the IC manufacturer
b17-b32	Tag memory size		See Table 59

Table 58 — Tag sensitivity class

Sensitivity class	Minimum sensitivity (V/m)
S1	10
S2	4
S3	1.5

Table 59 — Tag memory size information

MSB		LSB
b32-b30	b29-b25	b24-b17
RFU	Block size in bytes	Number of blocks

Block size is expressed in number of bytes using 5 bits, allowing specification of up to 32 bytes i.e. 256 bits. It is one less than the actual number of bytes; e.g., a value of '1F' indicates 32 bytes, a value of '00' indicates 1 byte.

Number of blocks is on 8 bits, allowing specification of up to 256 blocks. It is one less than the actual number of blocks; e.g., a value of 'FF' indicates 256 blocks, a value of '00' indicates 1 block.

The three most significant bits are reserved for future use and shall be set to zero.

The IC reference is on 8 bits and its meaning is defined by the IC manufacturer.

The tag shall perform the actions specified in Table 60.

Table 60 — Tag state transition for Get_system_information

Command: Get_system_information			
Current state	**Criteria**	**Action**	**New state**
Ready	The SUID does not match	none	Ready
	The SUID matches	The tag shall send back its response	
Quiet	The SUID does not match	none	Quiet
	The SUID matches	The tag shall send back its response	
Selected	The SUID flag = 0	The tag shall send back its response	Selected
	The SUID flag = 1 and the SUID matches	The tag shall send back its response	
	The SUID flag = 1 and the SUID does not match	none	
Round_active	The SUID does not match	none	Round_active
	The SUID matches	The tag shall send back its response	
Round_standby	The SUID does not match	none	Round_standby
	The SUID matches	The tag shall send back its response	

6.8.8 Begin_round

Command code = '0B'

The Begin_round command, as shown in Table 61, causes all tags whose user data matches the mask starting with the MSB of the data up to the position of the Mask Length parameter to reply, as shown in Table 62, with their first 32,64,96 or 128 bits of user data (as appropriate) when it is their turn to do so during a collision arbitration, refer to the state transition Table 63.

When the Mask Length exceeds the size of the data field the tag shall not reply but shall move to the ready state unless the tag was in the quiet state

NOTE The SUID flag is always = 0 because the tag will not act on the Tag ID if present, unless there is no user data, in which case the tag returns the SUID

Table 61 — Begin_round command format

Protocol extension	**Begin_round**	**SUID** (always = 0)	**Round size**	**CRC-5**	**Mask length**	**Mask**	**CRC-16**
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	0 – 255 bits	16 bits

Command parameter:

Mask Length

(optional) Mask

Command response

The response shall contain:

- The tag type (passive or BAP),
- The battery status flags,
- The tag signature,
- The AFI, and
- The first 32, 64, 96 or 128 bits of the tag memory.

If the tag detects an error, it shall remain silent.

Table 62 — Begin_round response format

Preamble	**Flags** see 6.4.8.2	**Tag type** see Table 38	**Battery status** see Table 39	**Signature**	**AFI**	**First n bits of memory**	**CRC-16**
	2 bits	1 bit	1 bit	4 bits	8 bits	n = 32, 64, 96 or 128 bits	16 bits

NOTE 1 The tag shall return the first n bits of user data according to its memory size, with a maximum of 128 bits. If there is no user memory, the tag shall return the SUID.

NOTE 2 The selection mask in the begin round command operates on the tag data content which coincides with the tag reply starting at the boundary between first and second bytes i.e. after the 4 bit signature value starting with the field currently marked AFI.

Table 63 — Tag state transition for Begin_round

Command: Begin_round			
Current state	**Criteria**	**Action**	**New state**
Ready	Mask Length = 0 or the mask matches the appropriate part of the user data.	The tag shall choose the slot in which it will send its response by generating a random number. It shall reset its slot counter to 1. See NOTE.	Round_active
	Mask Length > 0 and the mask does not match	The tag shall ignore the command and remain in the ready state.	Ready
Quiet	None	The tag shall discard the command and remain silent.	Quiet
Selected	None	The tag shall discard the command and transition to the Quiet state	Quiet
Round_active	Mask Length = 0 or the mask matches the appropriate part of the user data.	The tag shall reset the previously chosen slot and choose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
	Mask Length > 0 and the mask does not match	The tag shall ignore the command and shall move to the ready state.	Ready
Round_standby	Mask Length = 0 or the mask matches the appropriate part of the user data.	The tag shall reset the previously chosen slot and choose a new slot in which it will send its response by generating a random number. It shall reset its slot counter to 1.	Round_active
	Mask Length > 0 and the mask does not match	The tag shall ignore the command and shall move to the ready state.	Ready

NOTE The slots are numbered from 1 to Round_size.

6.8.9 Write_single_block

Command code = '10'

On receiving the Write_single_block command, the tag shall write the requested block with the data contained in the command and report the success of the operation in the response.

There are two formats of the Write_single_block command, one specifying the SUID, as shown in Table 64 and the other that does not specify the SUID, as shown in Table 65, but is used to write block data to the tag in the selected state.

Table 64 — Write_single_block command format, with SUID

Protocol extension	Write_single_ block	SUID flag = 1	RFU	CRC-5	SUID	Block number	Data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	8 bits	n bits	16 bits

Table 65 — Write_single_block command format, with no SUID

Protocol extension	Write_single _block	SUID flag = 0	RFU	CRC-5	Block number	Data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	n bits	16 bits

In Table 64 and Table 65 n is the block size. Currently, the block size is 32 bits, however, the protocol allows a block size of up to 256 bits.

The Data field Table 64 and Table 65 shall be ordered such that the most significant bit of the most significant byte of data is transmitted first

Command parameter:

(Optional) SUID

Block number

Data

Command response

If a tag in the selected state does not support the Write_single_block command it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If a tag in the selected state does support the Write_single_block command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If a tag in the selected state does support the Write_single_block command, and if the error flag is NOT set, the response shall be according to Table 24.

If the command specifies an SUID and the tag with the matching SUID, does not support the Write_single_block command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID does support the Write_single_block command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If the command specifies an SUID and the tag with the matching SUID does support the Write_single_block command, and if the error flag is NOT set, the response shall be according to Table 24.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see 6.4.8.2

The tag shall perform the actions specified in Table 66.

Table 66 — Tag state transition for Write_single_block

Command: Write_single_block			
Current state	**Criteria**	**Action**	**New state**
Ready	The SUID does not match	none	Ready
	The SUID matches	The tag shall process the command and send back its response	
Quiet	The SUID does not match	none	Quiet
	The SUID matches	The tag shall process the command and send back its response	
Selected	The SUID flag = 0	The tag shall process the command and send back its response	Selected
	The SUID flag = 1 and the SUID matches	The tag shall process the command and send back its response	
	The SUID flag = 1 and the SUID does not match	none	
Round_active	The SUID does not match	none	Round_active
	The SUID matches	The tag shall process the command and send back its response	
Round_standby	The SUID does not match	none	Round_standby
	The SUID matches	The tag shall process the command and send back its response	

6.8.10 Write_multiple_blocks

Command code = '11'

On receiving the Write_multiple_block command, the tag shall write the requested block(s) with the data contained in the command and report the success of the operation in the response.

The blocks are numbered from '00' to 'FF' (0 to 255).

The number of blocks in the command is one less than the number of blocks that the tag shall write.

For example, a value of '06' in the "Number of blocks" field requests to write 7 blocks. The "Data" field shall contain 7 blocks. A value of '00' in the "Number of blocks" field requests to write 1 block. The "Data" field shall contain 1 block.

There are two formats of the Write_multiple_blocks command, one specifying the SUID, as shown in Table 67 and the other that does not specify the SUID, as shown in Table 68, but is used to write multiple blocks to the tag in the selected state.

Table 67 — Write_multiple_blocks command format – with SUID

PE	Write_multiple _block	SUID flag = 1	RFU	CRC-5	SUID	First block number	Number of blocks	Data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	8 bits	8 bits	Block length	16 bits
								Repeated as needed	

Table 68 — Write_multiple_blocks command format – with no SUID

PE	Write_multiple_ block	SUID flag = 0	RFU	CRC-5	First block number	Number of blocks	Data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	8 bits	Block length	16 bits
							Repeated as needed	

In Table 67 and Table 68 the Block length is the block size. Currently, the block size is 32 bits, however, the protocol allows a block size of up to 256 bits. The data is repeated for the number of blocks. The Data field shall be ordered such that the most significant bit of the most significant byte of the first block is transmitted first, with subsequent blocks following.

Command parameter:

(Optional) Tag ID

First block number

Number of blocks

Block data (repeated as defined in the NOTE after Table 68).

Command response

If a tag in the selected state does not support the Write_multiple_blocks command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If a tag in the selected state does support the Write_multiple_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If a tag in the selected state does support the Write_multiple_blocks command, and if the error flag is NOT set, the response shall be according to Table 24.

If the command specifies an SUID and the tag with the matching SUID does not support the Write_multiple_blocks command, it shall return the error code “command is not supported”, error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID does support the Write_multiple_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If the command specifies an SUID and the tag with the matching SUID does support the Write_multiple_blocks command, and if the error flag is NOT set, the response shall be according to Table 24.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see 6.4.8.2

The tag shall perform the actions specified in Table 69.

Table 69 — Tag state transition for Write_multiple_blocks

Command: Write_multiple_blocks			
Current state	**Criteria**	**Action**	**New state**
Ready	The SUID does not match	none	Ready
	The SUID matches	The tag shall process the command and send back its response	
Quiet	The SUID does not match	none	Quiet
	The SUID matches	The tag shall process the command and send back its response	
Selected	The SUID flag = 0	The tag shall process the command and send back its response	Selected
	The SUID flag = 1 and the SUID matches	The tag shall process the command and send back its response	
	The SUID flag = 1 and the SUID does not match	none	
Round_active	The SUID does not match	none	Round_active
	The SUID matches	The tag shall process the command and send back its response	
Round_standby	The SUID does not match	none	Round_standby
	The SUID matches	The tag shall process the command and send back its response	

6.8.11 Lock_blocks

Command code = '12'

On receiving the Lock_blocks command, the tag shall lock permanently the requested block.

The tag “user lock information” is held in blocks of 32 bits. Each block is addressed by means of the command parameter “Lock Block Number”. Each bit in a lock block represents a block of data in the tag; the least significant bit of the lock block representing the lowest order block of tag data (lock bit 1 = tag user block 1). The protocol allows up to 256 blocks, with each block being up to 256 bits. The lock block command specifies a 32 bit quantity to write into the lock area, i.e. specifies 32 blocks out of the possible 256 blocks. The lock_block_number parameter (3 bits) allows the user to write the 32 bit lock data to one of eight 32 bit lock areas. Eight areas of 32 bits each, equals 256 lock areas representing the 256 blocks. To lock a tag data block, a bit is set in the “block lock data” field when writing to the tag. To not lock a block of data, the corresponding bit in the “block lock data” field shall contain a value ‘0’ (not set). Once a block has been locked, the set condition of the lock bit cannot be reversed.

NOTE The tag “factory lock bits” can only be locked by a proprietary command.

There are two formats of the Lock_blocks command, one specifying the SUID, as shown in Table 70 and the other that does not specify the SUID, as shown in Table 71, but is used to lock a block of the tag in the selected state.

Table 70— Lock_blocks command format – with SUID

Protocol extension	Lock_block	SUID Flag = 1	Lock block number	CRC-5	SUID	Block lock data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	32 bits	16 bits

Table 71 — Lock_blocks command format – with no SUID

Protocol extension	Lock_block	SUID Flag = 0	Lock block number	CRC-5	Block lock data	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	32 bits	16 bits

Command parameter:

(Optional) SUID

Block number

Command response:

If a tag in the selected state does not support the Lock_blocks command it shall return the error code "command is not supported", error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If a tag in the selected state does support the Lock_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If a tag in the selected state does support the Lock_blocks command, and if the error flag is NOT set, the response shall be according to Table 24.

If the command specifies an SUID and the tag with the matching SUID does not support the Lock_blocks command it shall return the error code "command is not supported", error code 1, see 6.4.8.2, 6.4.8.3 and Table 26.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_blocks command, and if the error flag is set, the response shall be according to 6.4.8.2 and Table 23.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_blocks command, and if the error flag is NOT set, the response shall be according to Table 24.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see 6.4.8.2

The tag shall perform the actions specified in Table 72.