INTERNATIONAL STANDARD

ISO/IEC 18000-6

First edition
2004-08-15

Information technology — Radio frequency identification for item management —

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

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

Partie 6: Paramètres pour les communications d'une interface d'air entre 860 MHz et 960 MHz

Reference number
ISO/IEC 18000-6:2004(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

ISO/IEC 18000-6 was prepared by 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*

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

Introduction

This part of ISO/IEC 18000 describes a passive backscatter 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.

In this RFID system, the interrogator powers and communicates with the tags that are within range. Tags receive data as amplitude modulation of the power/data signal from the interrogator. During the time that the tag responds to the interrogator, the interrogator transmits at a constant radio frequency power level, while the tag modulates the impedance of its radio frequency load attached to the tag antenna terminals. The interrogator then receives the data back from the tag as a variation in a reflection of its transmitted power.

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 given in the clauses identified below.

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

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate 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 number	Affected clause(s) in this part of ISO/IEC 18000
BTG plc ATTN: Mr David Armstrong 10 Fleet Place Limeburner Lane London EC4M 7SB UK Tel: +44 20 7575 0000 Fax: +44 20 7575 0010 Website: www.btgplc.com, Email: david.armstrong@btgplc.com,	US 5,537,105, US 5,966,083, US 5,995,017, US 5,557,280, US 5,699,066, US 5,519,381, US 5,726,630, EP 0494114B1, EP 0585132B1, EP 0598624B1, WO 98/52142 and WO 99/26081	6, 7

Contact details	Patent number	Affected clause(s) in this part of ISO/IEC 18000
Intercode 12, Rue des Petits Ruisseaux Z.I. des Godets 91370 Verrières le Buisson France Tel : + 33-1-6975 2170 Fax : + 33.1.60.11.00.31 Email: intercode.sarl@wanadoo.fr	US 5426423, EP 90909459.1, CA 2058 947, US 6177858B1, EP 96402556.3, CA 2191787, US 5923251, EP 96402554.8, CA 21911788, US 5808550, EP 96402555.5 and CA 2191794	7, 8
Intermec Technologies Corporation ATTN: Ronald D. Payne, Vice President, Contracts, 6001 36th Ave, West, Everett, WA 98203 USA	US 5942987, US 5521601, US 5995019, US 5030807, US 5828693, US 5850181, US 4786907, US 5550547, US 5673037, US 5777561 and US 5828318	8
Koninklijke Philips Electronics N.V ATTN: Mr.Harald Röggla Intellectual Property & Standards, Triester Strasse 64 A-1101 Vienna Austria	EP 1034503B, JP 00-560535, US 09/352317, WO 00/04485, JP 03-502778, US 2002/0186789A1 and WO 02/099741 A1	7, 8
Matrics Technology ATTN: Mr Kevin J Powell Senior Director, Product Development 8850 Stanford Blvd, Suite 3000 Columbia, MD 21045 USA +1-410-872-0300 (Voice) +1-443-782-0230 (eFax) kpowell@matrics.com	US 6002344	7, 8
SAMSys Technologies, Inc. ATTN: James Wiley, President, 2525 Meridian Parkway, Suite 60, Durham, NC 27713, USA Tel: +1-919- 281-1541, E-mail: tres.wiley@samsys.com	US 6617962	2
SUPERSENSOR (Pty) Ltd ATTN: Mr. Adelhart Kruger, D.M. Kisch Inc., P O Box 3668, Pretoria 0001, South Africa. Tel: +27-12-460-3203. E-mail: adelhartk@dmkisch.com	ZA 9810199, US 6480143 B1, EP 1001366, JP 200230978 and CN 1255689	6, 7
TAGSYS Australia Pty Ltd, ATTN: Alfio R. Grasso, TECHNICAL MANAGER 212 Pirie Street, ADELAIDE SA 5000 Australia, Tel: +61-8-8100 8324, E-mail: alfio.grasso@tagsys.net	EP 0578701B1, AU 664544, AU PCT AU/00/01493, WO 01/41043, AU PCT AU98/00017, WO 98/32092, US 5523749, AU PCT AU01/01676, WO02/054365, FR FR00/01704, and WO 01/01326	7

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.

Information technology — Radio frequency identification for item management —

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

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. Its purpose is to provide a common technical specification for RFID devices that may 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. This part of ISO/IEC 18000 defines the forward and return link parameters for technical attributes including, but not limited to, operating frequency, operating channel accuracy, occupied channel bandwidth, maximum 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 contains one mode with two types. Both types use a common return link and are reader talks first. Type A uses Pulse Interval Encoding (PIE) in the forward link, and an adaptive ALOHA collision arbitration algorithm. Type B uses Manchester in the forward link and an adaptive binary tree collision arbitration algorithm. The detailed technical differences between the two types are shown in the parameter tables.

2 Conformance

2.1 Interrogator conformance and obligations

To conform to this part of ISO/IEC 18000, the interrogator shall support both communication types. It shall be able to switch from one type to the other.

The interrogator shall be locally programmable by the user to switch from one type to the other and to control the sequence and allocation of the ratio of time between the two types.

The proportion of the total time spent by the interrogator in addressing each type of tag shall be field-programmable from 0% to 100%.

Interrogators shall be set up to operate within local regulations.

2.2 Tag conformance and obligations

To conform to this part of ISO/IEC 18000, the tag shall support at least one type. It may optionally support both.

The tag shall operate over the frequency range of 860 MHz to 960 MHz.

NOTE Depending on the tag antenna characteristics, the operating performance (i.e. operating range) may vary depending on the actual frequency used in the 860-960 MHz band.

When the tag receives a modulated signal from the interrogator that it does not support or recognise, it shall remain silent.

2.3 Claiming conformance

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

Relevant conformance test methods will be given in a future Technical Report (ISO/IEC TR 18047-6).

3 Normative references

ISO/IEC 19762-3, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 3: Radio frequency identification (RFID)*[1)]

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

4 Terms, definitions, symbols and abbreviated terms

4.1 Terms and definitions

For the purposes of this document, the terms, definitions, symbols and abbreviated terms given in ISO/IEC 19762-3 and the following apply.

4.1.1
collision arbitration loop
algorithm used to prepare for and handle a dialogue between an interrogator and a tag. This is also known as collision arbitration.

4.1.2
byte
8 bits of data designated b1 to b8, from the most significant bit (MSB, b8) to the least significant bit (LSB, b1).

4.2 Symbols

Cht Carrier high-level tolerance

Clt Carrier low-level tolerance

D Modulation depth of data coding pulse

f_c Frequency of operating field (carrier frequency)

M Modulation Index

Ma Modulation upper tolerance type B

Mb Modulation lower tolerance type B

1) To be published.

Taq Quiet time - type A

Tapf Pulse fall time - type A

Tapr Pulse rise time - type A

Tapw Pulse width - type A

Tari Reference interval time - type A

Tbmf Manchester fall time – type B

Tbmr Manchester rise time – type B

Tcf Carrier fall time

Tcr Carrier rise time

Tcs Carrier steady state time

Tf Fall time

Tfhf Carrier FHSS fall time

Tfhr Carrier FHSS rise time

Tfhs Carrier FHSS steady time

Tr Rise Time

Trlb Return link bit time

4.3 Abbreviated terms

AFI Application family identifier

CRC Cyclic redundancy check

CRC-5 Five bit CRC used in type a Interrogator to tag commands, used in Type A

CRC-16 Sixteen bit CRC used in both Type A and Type B commands and responses

DSFID Data storage format identifier

DSSS Direct Sequence Spread Spectrum

EOF End of frame

FHSS Frequency Hopping Spread Spectrum

LSB Least significant bit

MSB Most significant bit

NRZ Non Return to Zero

PIE Pulse interval encoding, used in Type A

RFU Reserved for future use

SOF Start of frame

SUID Sub unique identifier

TEL Tag excitation level

UID Unique identifier

5 Overview

5.1 General

This part of ISO/IEC 18000 specifies two communication types: Type A and Type B.

Figure 1, Figure 2, and Figure 3 show their architecture; Table 1 provides a comparison.

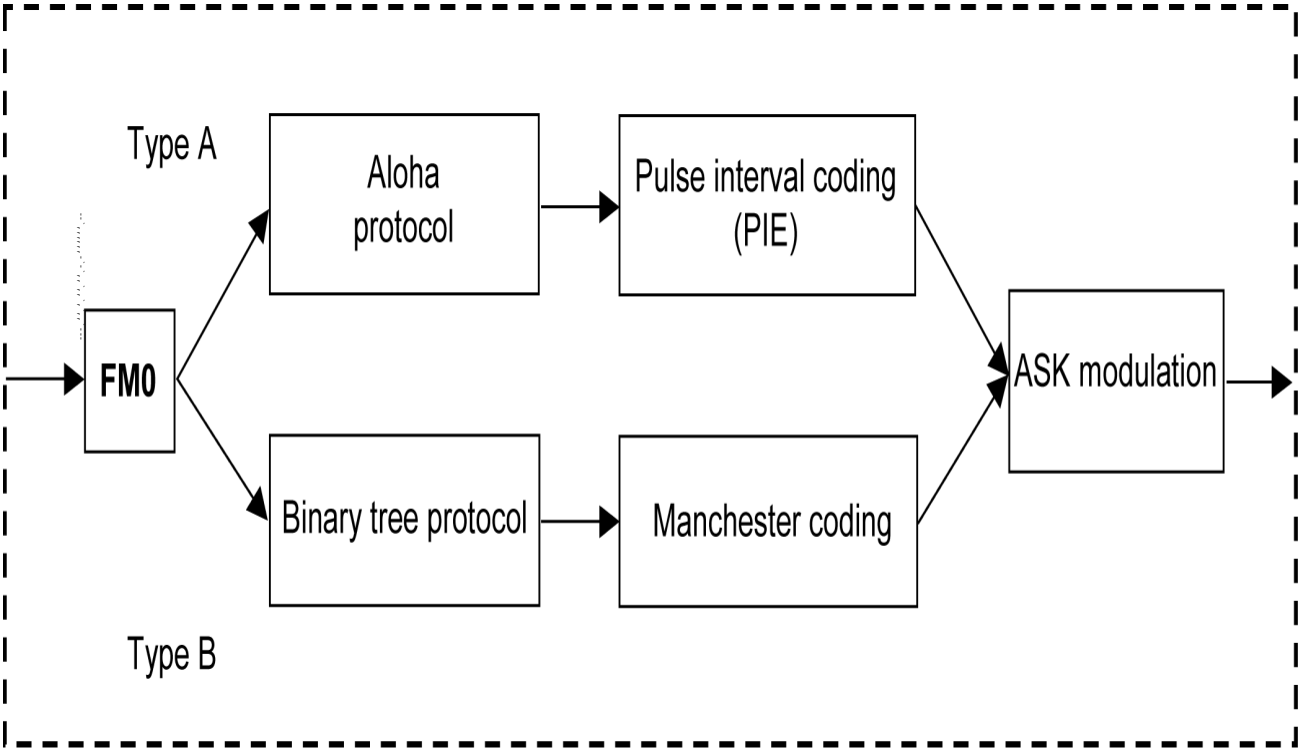


Figure 1 — Interrogator architecture

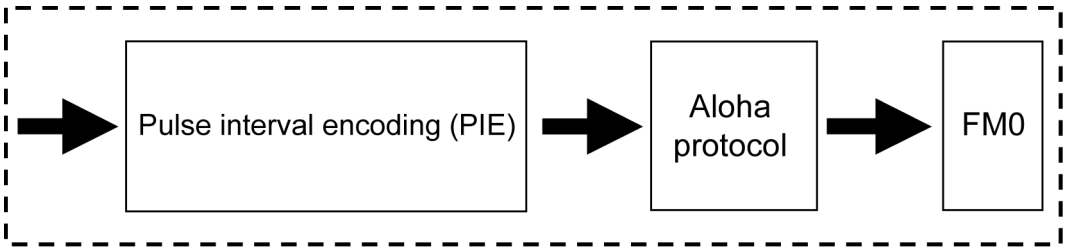


Figure 2 — Type A tag architecture

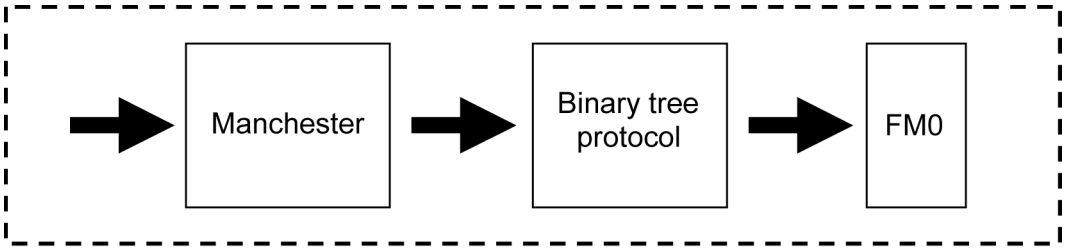


Figure 3— Type B tag architecture

Table 1 — Comparison of Type A and Type B

Parameter	Type A	Type B
Forward link encoding	PIE	Manchester
Modulation index	27% to 100%	18% or 100%
Data rate	33 kbit/s (mean)	10 or 40 kbit/s (according to local regulations)
Return link encoding	FM0	FM0
Collision arbitration	ALOHA	Binary Tree
Tag unique identifier	64 bits (40 bit SUID)	64 bits
Memory addressing	Blocks up to 256 bits	Byte blocks, 1,2,3 or 4 byte writes.
Error detection forward link	5 bit CRC for all commands (with an additional 16 bit CRC appended for all long commands)	16 bit CRC
Error detection return link	16 bit CRC	16 bit CRC
Collision arbitration linearity	Up to 250 tags	Up to 2^{256}

5.2 Parameter tables

Table 2, Table 3, Table 4 and Table 5 contain the parameters for both types A and B in accordance with ISO/IEC 18000-1. Detailed description of the operating modes and parameters are specified in the subsequent clauses.

Table 2 — Interrogator to tag link parameters

Interrogator to tag	Parameter name	Description
Int: 1	Operating frequency range	860 – 960 MHz The interrogator operating frequency range shall be determined by the radio regulations in force in a particular regulatory jurisdiction and by the type approval requirements of the particular jurisdiction. Before an interrogator may be used, it shall meet the local radio regulations of the country in which it is to be used. It is envisaged that there will be more than one version of interrogator having different frequency and power characteristics in order to comply with local regulations. NOTE Performance will vary according to bandwidth and power output permitted by local regulations.
Int: 1a	Default operating frequency	In accordance with the local radio regulations. See Int: 1
Int: 1b	Operating channels	In accordance with the local radio regulations. See Int: 1
Int: 1c	Operating frequency accuracy	In accordance with the local radio regulations See Int: 1
Int: 1d	Frequency hop rate (for frequency hopping [FHSS] systems)	Not applicable for single fixed frequency or channelized Adaptive Frequency Agile systems. Where FHSS is permitted, the hop rate shall be in accordance with the local radio regulations.
Int: 1e	Frequency hop sequence (for frequency hopping [FHSS] systems)	In accordance with the local radio regulations. Where not specified by such regulations a pseudo-random hopping sequence shall be used that ensures an even distribution of transmissions over the available channels.
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 transmitter maximum EIRP	In accordance with the local radio regulations
Int: 4	Interrogator transmitter spurious emissions	In accordance with the local radio regulations.
Int: 4a	Interrogator transmitter spurious emissions, in-band (for spread spectrum systems)	In accordance with the local radio regulations.
Int: 4b	Interrogator transmitter spurious emissions, out-of-band	In accordance with the local radio regulations.
Int: 5	Interrogator transmitter spectrum mask	As per Int: 2 and Int: 4a.
Int:6	Timing	See below Int: 6x.
Int: 6a	Transmit to receive turn around time	The interrogator transmit/receive settling time shall not exceed 85µs.
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	Maximum 1.5ms.
Int: 6d	Decay time of interrogator transmitter power down ramp	Maximum 1ms.

Table 2 *(continued)*

Interrogator to tag	Parameter name	Description
Int: 7	Modulation	Amplitude Modulation.
Int: 7a	Spreading sequence (for direct sequence [DSSS] systems)	Not applicable.
Int: 7b	Chip rate (for spread spectrum systems)	Not applicable.
Int: 7c	Chip rate accuracy (for spread spectrum systems)	Not applicable.
Int: 7d	Modulation index	**Type A**: Nominal 30 % to 100 %. See Table 11 and clause 7.1.1. **Type B**: Nominal 18% or 100% .
Int: 7e	Duty cycle	In accordance with local regulations.
Int: 7f	FM deviation	Not applicable.
Int: 7g	Transmitter modulation pulse shape (falling and rising slopes)	The modulation pulse fall and rise times shall conform to the detailed specifications in subsequent clauses. Type A see Figure 11 and Table 11, and Type B see Figure 20, Figure 21, Table 93 and Table 94.
Int: 8	Data coding	**Type A**: Pulse Interval Encoding **Type B**: Manchester bi-phase
Int: 9	Bit rate	**Type A**: 33 kbit/s as constrained by the local radio regulations **Type B**: 10 kbit/s or 40 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	**Type A** : has no preamble **Type B**: See clause 8.1.4.3
Int:11a	Preamble length	**Type A**: not applicable. **Type B**: 9 bits. See clause 8.1.4.3.
Int:11b	Preamble waveform	**Type A**: not applicable . **Type B**: See clause 8.1.4.3.
Int: 11c	Bit sync sequence	**Type A**: not applicable. **Type B**: see clause 8.1.4.3
Int: 11d	Frame sync sequence	Not applicable.
Int: 12	Scrambling (for spread spectrum systems)	Not Applicable
Int: 13	Bit transmission order	MSB 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 3 — Tag to interrogator link parameters

Tag to interrogator	Parameter name	Description
Tag:1	Operating frequency range	860 MHz – 960 MHz
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 (for 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 (for frequency hopping [FHSS] systems)	Not applicable
Tag:1e	Frequency hop sequence (for frequency hopping [FHSS] systems)	Not applicable
Tag:2	Occupied channel bandwidth	As permitted by the local radio regulations.
Tag:3	Transmit maximum EIRP	As permitted by the local radio regulations.
Tag:4	Transmit spurious emissions	As permitted by the local radio regulations.
Tag:4a	Transmit spurious emissions, in-band (for spread spectrum systems)	As permitted by the local radio regulations.
Tag:4b	Transmit spurious emissions, out-of-band	As permitted by the local radio regulations.
Tag:5	Transmit spectrum mask	As permitted by the local radio regulations.
Tag:6a	Transmit to receive turn around time	**Type A:** The tag shall open its receive command window within 2 bit periods of the end of its response. **Type B:** 400 µs
Tag:6b	Receive to transmit turn around time	**Type A :** Range from 150 to 1150 µs (see clause 7.5.3 and Table 28) **Type B :** Range 85 to 460 µs (see clause 8.1.5.2)
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 (for direct sequence [DSSS] systems)	Not applicable.
Tag:7b	Chip rate (for spread spectrum systems)	Not applicable.
Tag:7c	Chip rate accuracy (for 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 $0.005m^2$.
Tag:7e	Sub-carrier frequency	Not applicable.
Tag:7f	Sub-carrier frequency accuracy	Not applicable.
Tag:7g	Sub-carrier modulation	Not applicable.

Table 3 *(continued)*

Tag to interrogator	Parameter name	Description
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)
Tag:9	Bit rate	Typical 40 kbit/s or 160 kbit/s (subject to tag clock tolerance see Table 9), The return bit rate selection for 160 kbit/s for Type B is defined in clause 8.1.4.4.5.
Tag:9a	Bit rate accuracy	+/- 15% (refer to Table 9)
Tag:10	Tag transmit modulation accuracy (for 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 (for spread spectrum systems)	Not applicable.
Tag:13	Bit transmission order	MSB 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 4 — Protocol parameters

Ref.	Parameter name	Description
P:1	Who talks first	Interrogator talks first.
P:2	Tag addressing capability	Type A see clause 7.3.1 Type B see clause 8.2.2
P:3	Tag unique identifier (UID)	Contained in tag memory and accessible by means of a command.
P:3a	UID length	**Type A:** 64 bits. An SUID of 40 bits is used during census or collision arbitration transactions. **Type B:** 64 bits
P:3b	UID format	The UID format is different for types A and B. See clause 7.2.1 for type A. See clause B.2 for type B.

Table 4 *(continued)*

Ref.	Parameter name	Description
P:4	Read size	**Type A:** Addressable in blocks of up to 256 bits, but always an integer multiple of bytes. **Type B**: Addressable in byte blocks.
P:5	Write size	**Type A:** Addressable in blocks of up to 256 bits, but always an integer multiple of bytes. **Type B:** Addressable in byte blocks. Writing in blocks of 1, 2, 3 or 4 bytes. See details in relevant clauses.
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.
P:8	Error detection	Interrogator to tag **Type A:** 16-bit commands: CRC-5. Commands of more than 16 bits have an additional CRC-16, bit protection. **Type B:** CRC-16 Tag to interrogator: CRC-16 for both types.
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.
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	**Type A:** Several command codes are reserved for future use. In addition, a Protocol extension flag allows for further extensions. **Type B:** Several command codes are reserved for future use.

Table 5 — Anti-collision parameters

Ref.	Parameter name	Description
A:1	Type (Probabilistic or Deterministic)	**Type A**: Probabilistic. **Type B**: Probabilistic.
A:2	Linearity	**Type A**: Essentially linear using adaptive slot allocation up to 256 slots for 250 tags in an interrogator zone. **Type B**: Essentially linear up to 2^{256} tags depending on size of data content.
A:3	Tag inventory capacity	The algorithm permits the reading of not less than 250 tags in the reading zone of the interrogator.

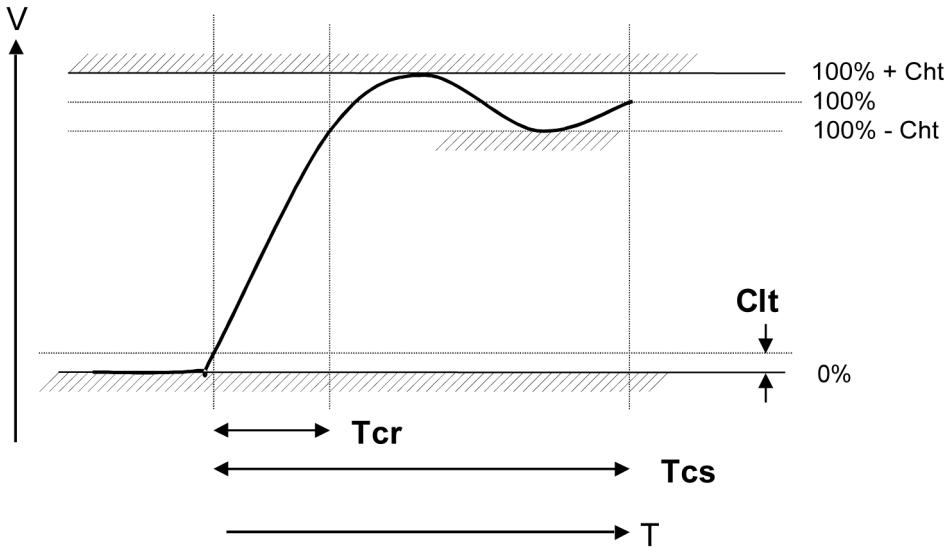


Figure 4 — Interrogator power-up waveform

Table 6 — Interrogator power-up waveform parameter values

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

6 Common elements of the physical layer for types A and B

6.1 General

Types A and B have several common elements for the physical layers. They are specified in the following sub-clauses.

6.2 Interrogator power-up waveform

The interrogator power-up waveform shall comply with the mask specified in Figure 4 and Table 6.

6.3 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 5 and Table 7.

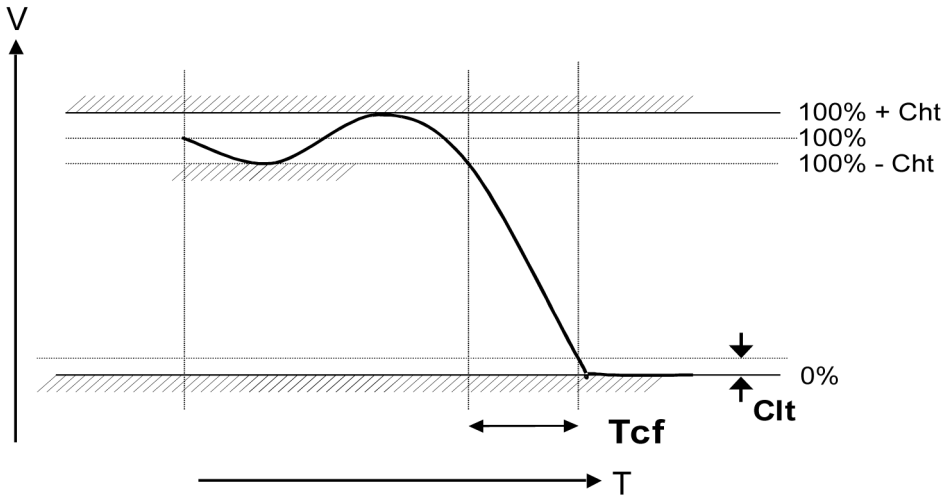


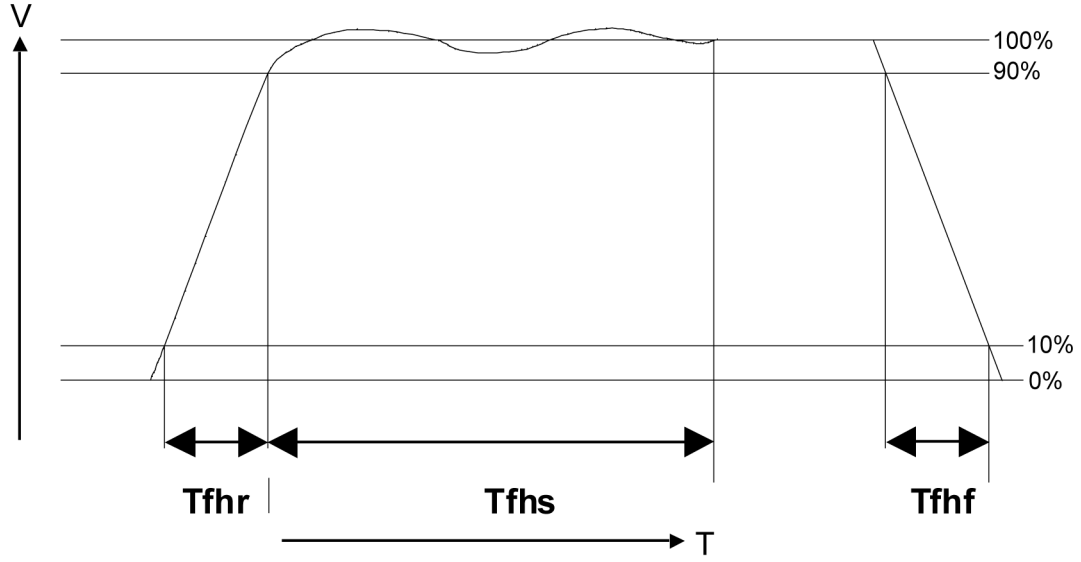
Figure 5 — Interrogator power-down waveform

Table 7 — Interrogator power-down timings

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

6.4 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 6 and Table 8. 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 6 — FHSS carrier rise and fall characteristics

Table 8 — FHSS carrier rise and fall parameters

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

6.5 FM0 return link

6.5.1 FM0 return link general

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

6.5.2 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.

6.5.3 Data rate

The data rate is 40kbit/s.

6.5.4 Data coding

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

One symbol period Trlb, as specified in Table 9, 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 9 — Return link parameters

Data rate	Trlb	Tolerance
40kbit/s	25 µs	+/-15%

Coding of data is MSB first. Figure 7 illustrates the coding for the 8 bits of 'B1'.

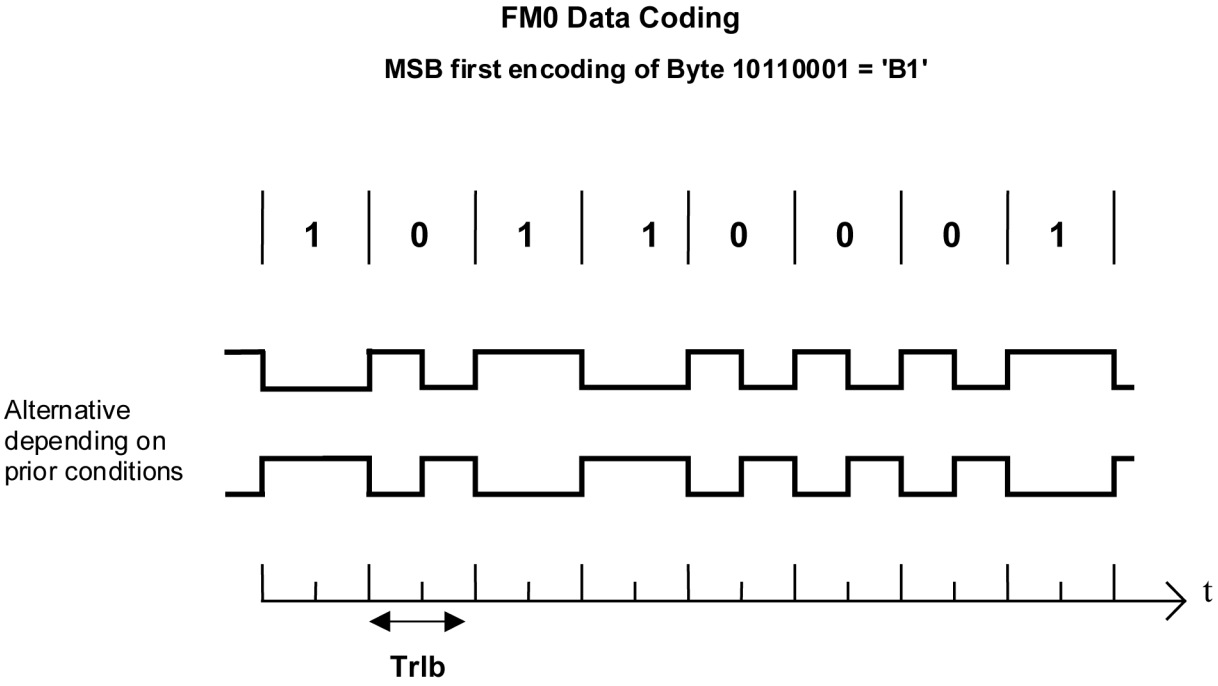


Figure 7 — Tag to interrogator data coding

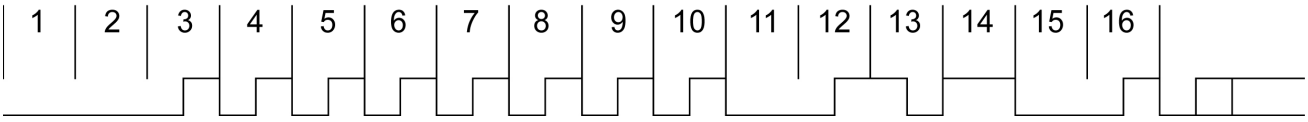
6.5.5 Message format

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 8. There are multiple code violations (sequences not conforming to FM0 rules) that act as a frame marker for the transition from Preamble to Data.

6.5.6 Return preamble

The return preamble is a sequence of backscatter modulation specified in Figure 8.



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

Figure 8 — 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 9.

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

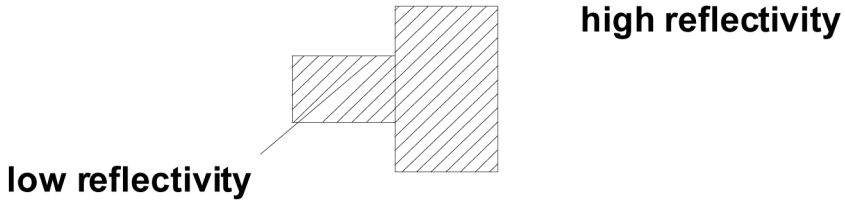


Figure 9 — Return link bit coding

6.5.7 Cyclic redundancy check (CRC)

6.5.7.1 CRC General

Types A and B use the same CRC-16 for the forward and the return links. In addition, Type A uses CRC-5 for short commands 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.

6.5.7.2 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.

6.5.7.3 Interrogator to tag 16 bit CRC-16

6.5.7.3.1 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:

6.5.7.3.2 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.

6.5.7.3.3 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)

6.5.7.4 Tag to interrogator 16 bit CRC-16

6.5.7.4.1 Tag to interrogator to tag 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 10. 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 10 — 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.

6.5.7.4.2 Inversion of incoming CRC bits by the interrogator

At the reader 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.

6.5.7.4.3 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).

7 Type A

7.1 Physical layer and data coding

7.1.1 PIE (Pulse interval encoding) forward link

7.1.1.1 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 10.

The carrier modulation pulses are negative-going raised cosine shaped, their characteristics are specified in Figure 11 and Table 11.

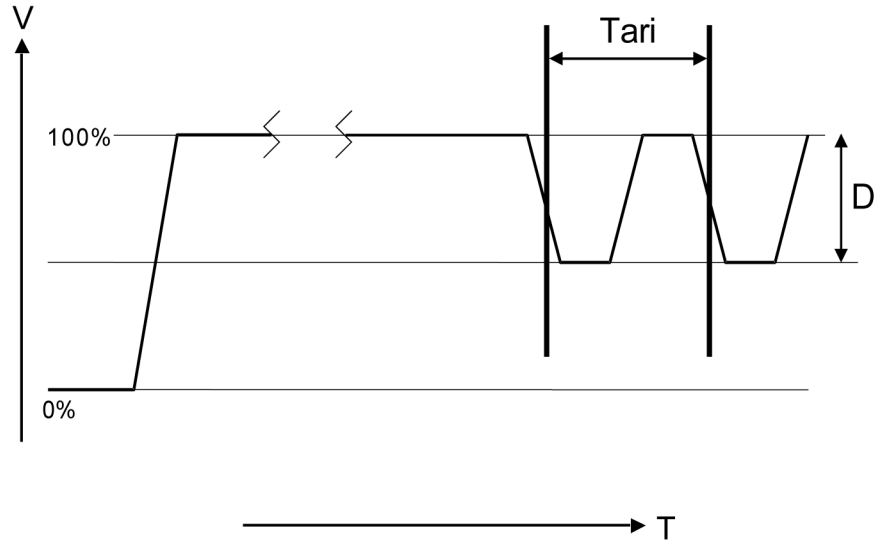
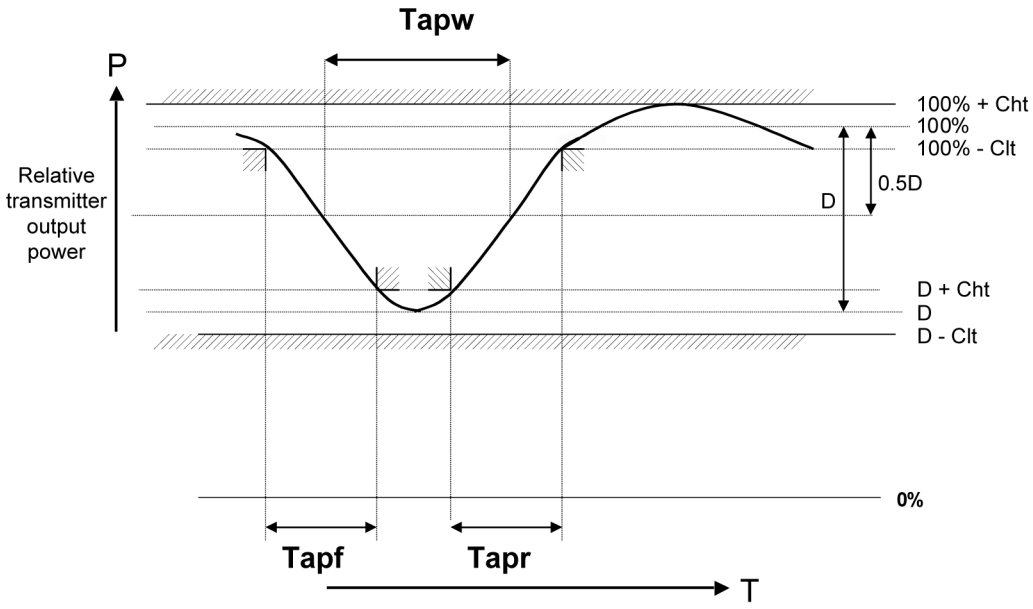


Figure 10 — Inter-pulse mechanism



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

Figure 11 — Modulation shaping

Table 11 — 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 is may be in the range of 0.73 to 0 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.

7.1.1.2 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 12

Table 12 — Reference interval timing

Tari	Tolerance
20 µs	± 100 ppm

Four symbols are defined as shown as shown in Table 13 and Figure 12. 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 13. 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 13 — 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

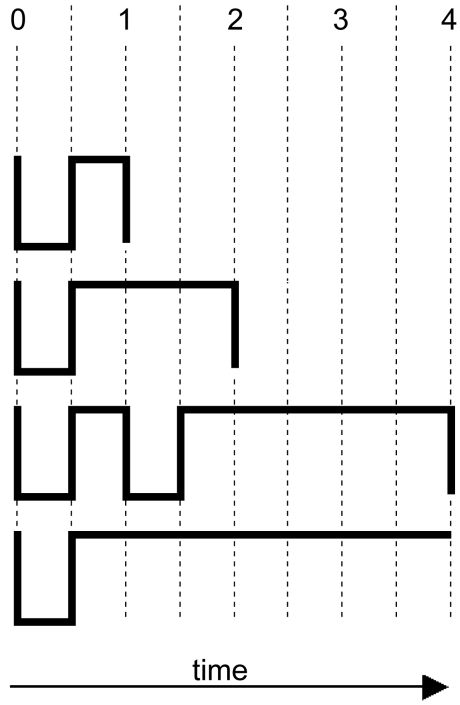


Figure 12 — PIE symbols

7.1.1.3 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 14:

Table 14 — 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.

7.1.1.4 Frame format

The bits transmitted by the interrogator to the tag are embedded in a frame as specified in Figure 13.

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 300µs.

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.

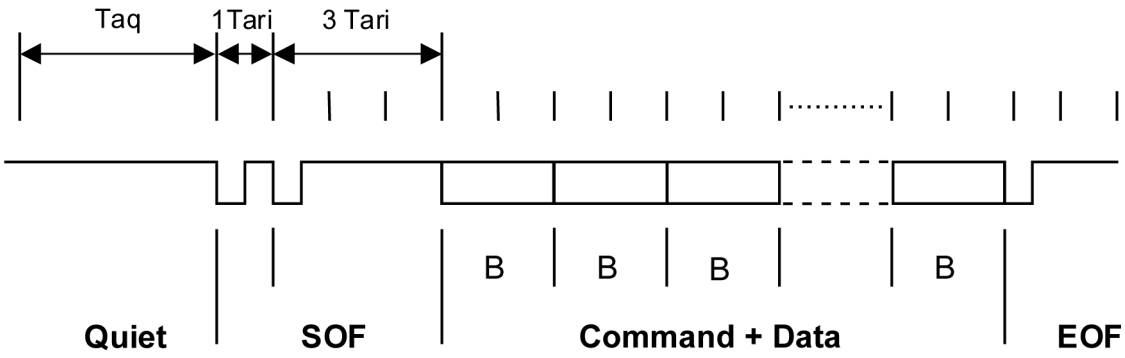


Figure 13 — Forward link frame format

7.1.1.5 Data decoding

The tag shall decode the symbols specified in Table 13.

The tag shall be able to decode an incoming signal having a modulation depth (as specified in clause 7.1.1.1, Figure 11 and Table 11) with respect to the reader steady state carrier of 27 % or greater (ie 27 % to 100 %). In other words the modulation depth may be anywhere in the range of 0.73 to 0 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 4Tari duration.

7.1.1.6 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 14.

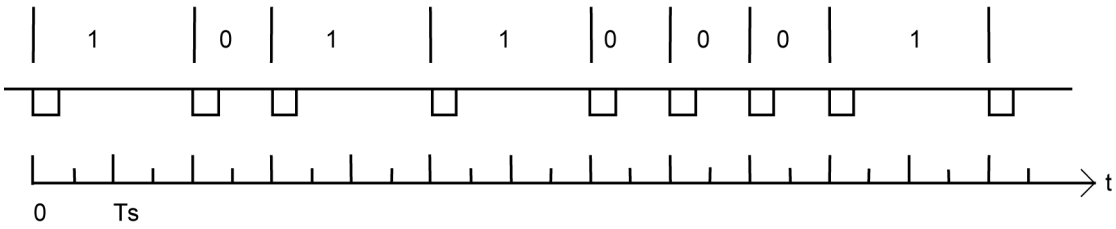


Figure 14 — Example of PIE byte encoding for 'B1'

7.2 Data elements

7.2.1 Unique identifier (UID)

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 (UID) shall be in this form as specified in Table 15 and ISO/IEC 15963.

If used, the UID shall be set permanently by the IC manufacturer in accordance with Table 15.

Table 15 — UID 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 UID 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.

7.2.2 Sub-UID

When the Aloha protocol is used, only a part of the UID, called Sub-UID (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 UID of 64 bits. See clause 7.8.7.

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 UID to the transmitted 40 bits and back is described in Figure 15.

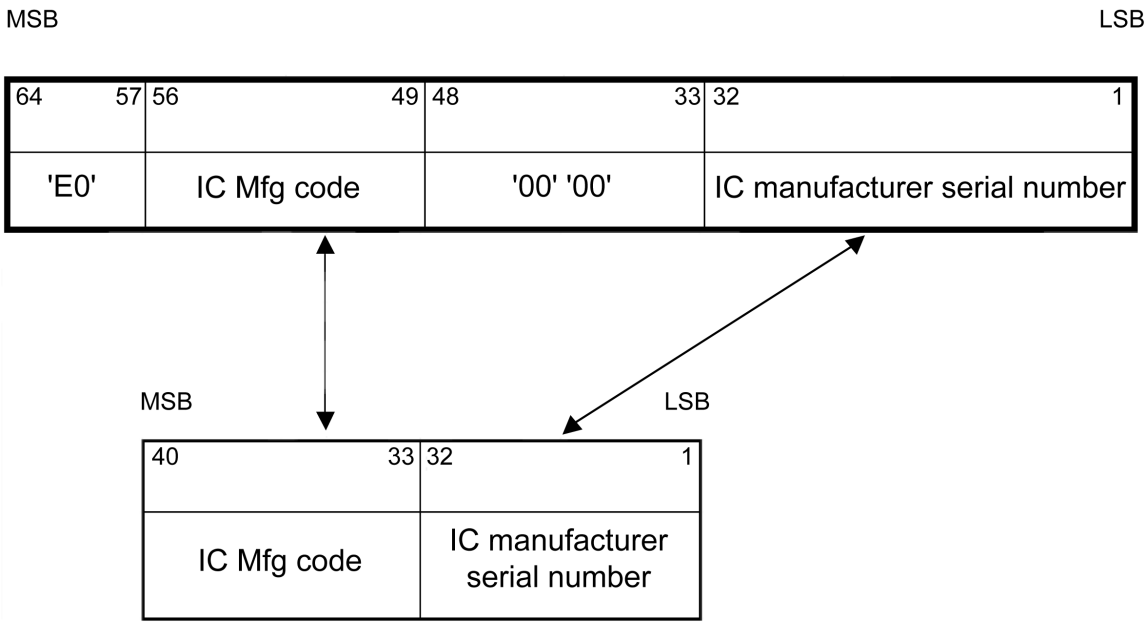


Figure 15 — UID/SUID mapping from 64 to 40

The interrogator shall use the 64-bit format specified in clause 7.2.1 when exchanging the UID with the application. It shall perform the required mapping described in Figure 15.

7.2.3 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 16.

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 16 — AFI/ASF coding

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

NOTE Table 16 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.

7.2.4 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').

7.3 Protocol elements

7.3.1 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).

7.3.2 Support of battery-assisted tags

This part of ISO/IEC 18000 makes provision for the support of battery-assisted tags.

In normal operation, there is no functional difference between passive and battery-assisted tags.

The mechanisms provided to handle the differences in performance and maintenance of tags that are battery-assisted and those that are not battery-assisted are as follows:

a) The type of tag and its sensitivity level are returned in the answer to the Get_system_information command. See clause 7.8.7.

b) The type of tag (battery/no battery) and the battery status is returned in the tag answer during the collision arbitration sequence. See clause 7.7.5.

7.3.3 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 clause 7.8.16. The block lock status is coded using two bits. Table 17 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 17 — 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

7.3.4 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 UID, 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 16, and the respective state transition tables. For the Next_slot command see Table 31and for the Standby_round command see Table 33.

7.4 Protocol description

7.4.1 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 clause 7.7 and 7.8.

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).

7.4.2 Command format

7.4.2.1 Type A command format general

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

7.4.2.2 Short command format

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

Table 18 — Short command format

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

7.4.2.3 Long command format

A long command consists of the fields defined in Table 19.

Table 19 — 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	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.

7.4.3 Command flags

7.4.3.1 Command flags

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

7.4.3.2 SUID flag

7.4.3.2.1 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 22.

7.4.3.2.2 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 clauses 7.8.2 and 7.7.5

7.4.3.2.3 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. (ie. 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.

7.4.4 Round size

The round size is coded using 3 bits according to Table 20.

Table 20 — 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	RFU

7.4.5 Command code definition and structure

The size of the command code is 6 bits.

7.4.6 Command classes

7.4.6.1 Command classes general

Four sets of commands, as shown in Table 21, are defined: mandatory, optional, custom and proprietary.

Table 21 — 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'	Optional	30
'28' – '37'	Custom	16
'38' – '3F'	Proprietary	8

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

7.4.6.2 Mandatory

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

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

7.4.6.3 Optional

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

Optional commands are commands that are specified within this part of ISO/IEC 18000. Interrogators shall be technically capable of performing all optional commands that are specified in this part of ISO/IEC 18000 (although need not be set up to do so). Tags may or may not support optional commands.

If an optional command is used, it shall be implemented in the manner specified in this part of ISO/IEC 18000.

If a tag in the selected state does not support the optional command it shall return the error code "command is not supported", see clause 7.4.8.2.

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 clause 7.4.8.2.

All other tags shall remain silent.

7.4.6.4 Custom

The custom command codes range from '28' to '37'. Custom commands may be enabled by this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000. A custom 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.

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 clause 7.4.8.2.

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 clause 7.4.8.2.

All other tags shall remain silent.

7.4.6.5 Proprietary

The proprietary command codes range from '38' to '3F'. Proprietary commands may be enabled this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000. 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. This part of ISO/IEC 18000 allows these commands to be disabled after IC and/or tag manufacturing.

7.4.7 Command codes and CRC

All mandatory and optional commands are mandatory for a reader implementation, although the reader 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 22, with the 4 mandatory tag commands being highlighted in bold.

Table 22 — 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
Next_slot	**02**	**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. A signature value of 0x00 (HEX) will only advance the slot counter (equivalent to close-slot command.)**
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						

Table 22 *(continued)*

Command 6 bits	Op code	Parameter / flags 4 bits		CRC-5	Extended parameters				CRC-16	Comments
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
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	

Table 22 *(continued)*

Command 6 bits	Op code	Parameter / flags 4 bits		CRC-5	Extended parameters				CRC-16	Comments
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						
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									

7.4.8 Response format

7.4.8.1 Response format general

The response from the tag, as shown in Table 23, consists of the following fields:

- Preamble (See clause 6.5.6),
- Flags (See clause 7.4.8.2),
- One or more parameter fields (Defined in each command),
- Data (Defined in each command), and
- CRC-16 (See clause 6.5.7.4).

Table 23 — General response format

Preamble	Flags	Parameters	Data	CRC-16

7.4.8.2 Response error flags

In a response, two flags, as shown in Table 24, Table 25 and Table 26, indicate how the tag has performed actions and whether corresponding fields are present or not.

Table 24 — Response format when Error_flag is set

Preamble	Flags	Error code	Terminator
	2 bits	Optional 4 bits See Table 27	'0' 1 bit

Table 25 — 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 26 — 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.

7.4.8.3 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 27.

If the tag does not support specific error code(s) listed in Table 27, it shall answer with the error code 'F' ("Error with no information given").

Table 27 — 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.

7.4.9 Tag states

7.4.9.1 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 16 and are specified in tables in each command description.

7.4.9.2 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 battery-assisted tags, it means that the level of RF excitation is insufficient to turn on the tag circuits.

7.4.9.3 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.

7.4.9.4 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.

7.4.9.5 Selected state

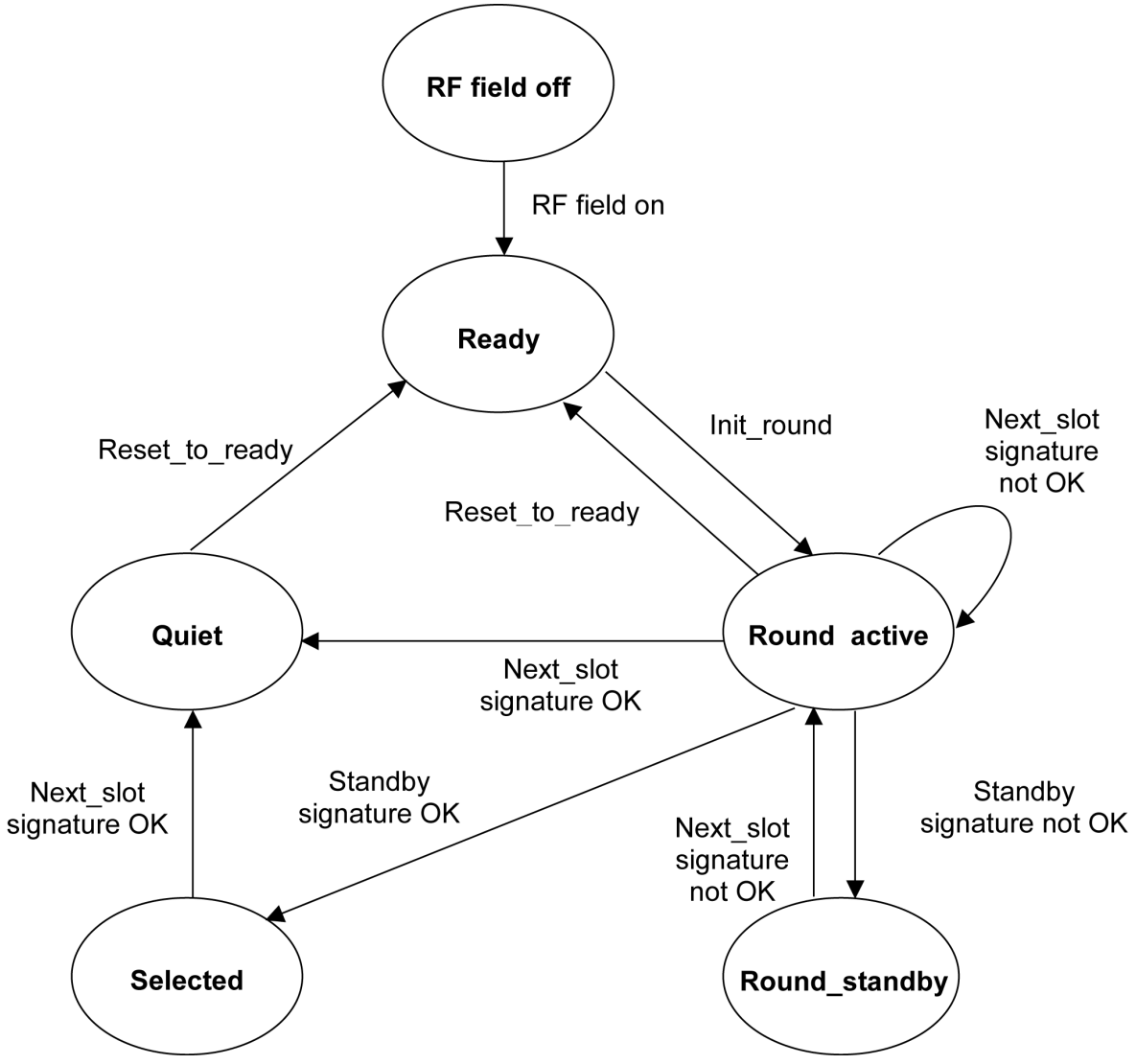
Only a tag in the Selected state shall process commands with the SUID flag set to 0, with the exception of Init_round, Init_round_all, Begin_round and New_round commands, which will be processed even if the SUID flag is set to 1.

7.4.9.6 Round_active state

When in the Round_active state, the tag shall participate in the collision arbitration described in clauses 7.4.10 and 7.4.11.

7.4.9.7 Round_standby state

When in the Round_standby state, the tag shall suspend its participation in the collision arbitration as described in clauses 7.4.10 and 7.4.11.



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

Figure 16 — Tag state transition diagram

7.4.10 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.

7.4.11 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 energising 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 clause 7.5.5.

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.

7.5 Timing specifications

7.5.1 Timing specifications general

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

7.5.2 Tag state storage

In the case of absent or insufficient energising 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.

7.5.3 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 17 and Table 28.

Table 28 — Forward link to return link handover timing

Minimum	Maximum
150µs	1150µs

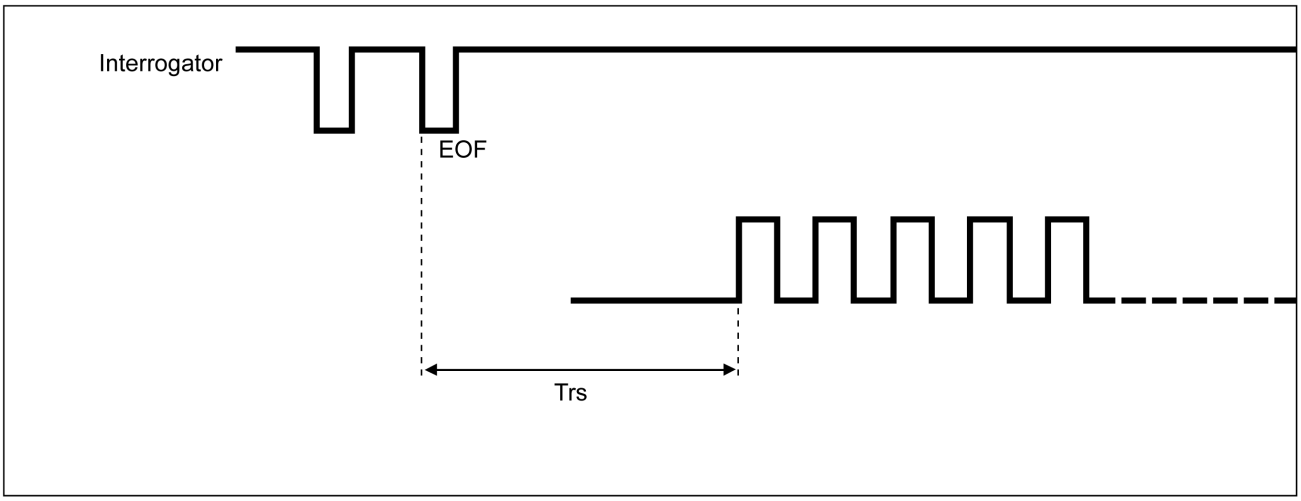


Figure 17 — Forward link to return link handover

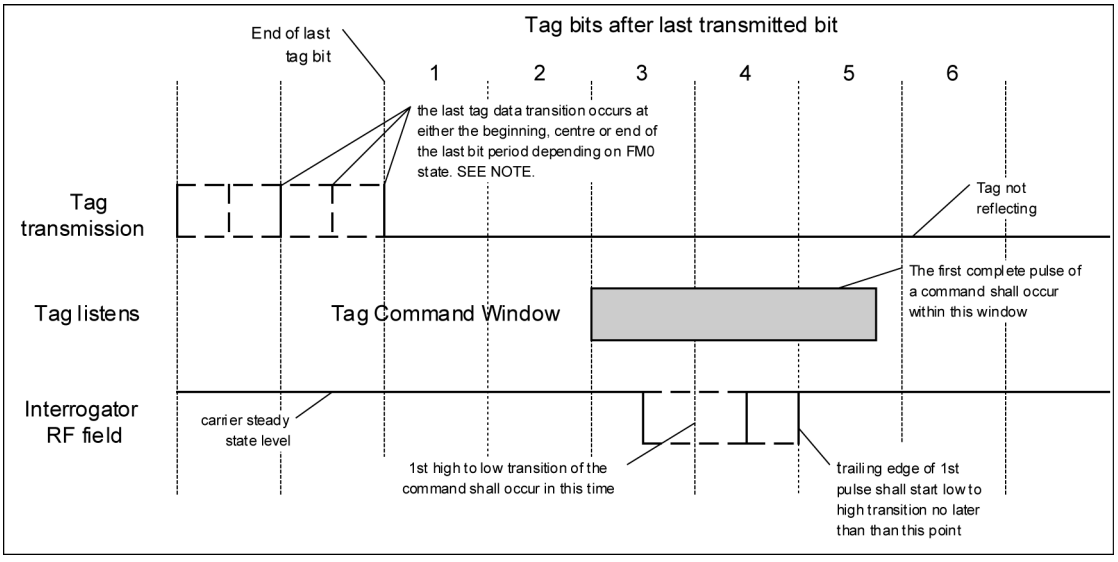
7.5.4 Return link to forward link handover

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

7.5.5 Acknowledgement time window

7.5.5.1 Acknowledgement time window general

The purpose of the acknowledgement time window, shown in Figure 18, 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.



NOTE The four cases of the FM0 state are detailed in Figure 19.

Figure 18 — Acknowledgement time window

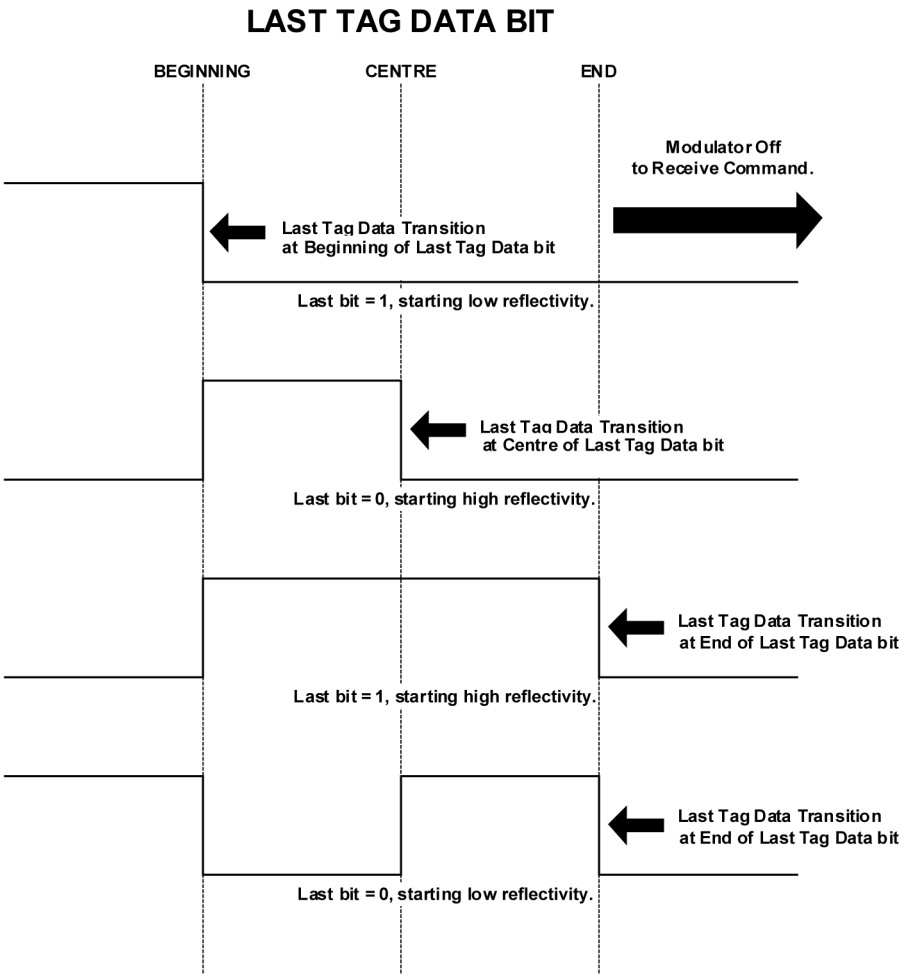


Figure 19 — Details of last FM0 tag data bit

7.5.5.2 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 18.

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’.

7.5.5.3 Tag

When a tag is in the Round active state, has just responded in the current slot and receives a Next_slot command, which was not received within the window specified in clause 7.5.5, 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 Clause 7.7.2 and Table 31.

When a tag is 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 clause 7.5.5, the tag shall not move to the selected state, but shall transition to the Round_standby state as specified in Clause 7.7.3 and Table 33.

7.6 Command format examples

For examples of the bit pattern for a short command, see 7.7.2 and for a long command see 7.8.6.

7.7 Mandatory commands

7.7.1 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.

7.7.2 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 29 shows the Next_slot command format.

Table 29 — Next_slot command format

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

Table 30 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 30 — 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 (eg = 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 31. It may either enter the Quiet state or switch to the next slot by incrementing its slot counter.

Table 31 — 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 clause 7.5.5	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 clause 7.5.5	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

7.7.3 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 Select 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. 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 32 shows the Standby_round command format.

Table 32 — Standby_round command format

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

There is no tag response to the Standby_round command. The tag shall perform the algorithm described in clauses 7.4.10 and 7.4.11. It may either enter the Select state or the Round_standby state, as shown in Table 33.

Table 33 — 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 clause 7.5.5	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 clause 7.5.5.	None	Round_standby
Round_standby	None	None	Round_standby

7.7.4 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 34 shows the Reset_to_ready command format.

Table 34 — 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 35.

Table 35 — 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

7.7.5 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 36 shows the Init_round_all command format.

Table 36 — 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 37 or Table 38 shall contain:

- The DSFID if the SUID flag is set in the command,
- The tag signature,
- The tag type (battery-less or battery-assisted),
- 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 37 — Init_round_all response format when the SUID flag is NOT set

Preamble	Flags see 7.4.8	Tag type see Table 39	Battery status see Table 40	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 clause 7.8.7.

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 UID, or a random number, etc.).

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

Table 38 — Init_round_all response format when the SUID flag is set

Preamble	Flags see 7.4.8	Tag type see Table 39	Battery status see Table 40	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 39 shows the Tag type field, while Table 40 shows the Battery Status field.

Table 39 — Tag type (battery-assisted or not)

Tag type	Meaning
0	Tag is NOT battery-assisted
1	Tag is battery-assisted

Table 40 — Battery status (for battery-assisted tag)

Battery status	Meaning
0	Battery is low. A non-battery-assisted tag shall set this bit to 0.
1	Battery is good.

The tag shall perform the actions specified in Table 41.

Table 41 — 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.

7.8 Optional commands

7.8.1 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

All other tags shall remain silent.

7.8.2 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 42 shows the Init_round command format.

Table 42 — 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 Clause 7.7.5, Table 37 and Table 38.

The tag shall perform the actions specified in Table 43.

Table 43 — 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.

7.8.3 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.

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 44 shows the Close_slot command format.

Table 44 — 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 45.

Table 45 — 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	The tag shall discard the command and transition to the Quiet state.	Quiet
Round_active	None	The tag shall increment its slot counter and send its response if its slot counter matches the chosen slot.	Round_active
Round_standby	None	The tag shall increment its slot counter and send its response if its slot counter matches the chosen slot.	Round_active

7.8.4 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 46 shows the New_round command format.

Table 46 — 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 37.

If the SUID = 1, the tag shall respond with the SUID as in Table 38.

The tag shall perform the actions specified in Table 47.

Table 47 — 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

7.8.5 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 48 shows the Select command format.

Table 48 — 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 24.

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 25.

All other tags shall remain silent.

The tag shall perform the actions specified in Table 49.

Table 49 — 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

7.8.6 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 50, and the other that doesn't specify the SUID, see Table 51, but is used to read blocks from the tag in the selected state.

Table 50 — 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 51 — 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 52 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 52 — 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 53.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 53.

All other tags shall remain silent.

Table 53 — Read_blocks response format when Error_flag is NOT set

Preamble	Flags see 7.4.8	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 17

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 clause 7.4.8.2

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 54.

Table 54 — 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	

7.8.7 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 55 and the other that doesn't specify the SUID, as shown in Table 56, but is used to get system information from the tag in the selected state.

Table 55 — 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 56 — 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 57.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 57.

All other tags shall remain silent.

Table 57 — Get_system_information response format when Error_flag is NOT set

Preamble	Flags see clause 7.4.8	Tag type see Table 39	Battery status see Table 40	Signature	UID see NOTE 1	DSFID see NOTE 2	AFI see NOTE 3	System information see Table 58	CRC-16
	2 bits	1 bit	1 bit	4 bits	64 bits	8 bits	8 bits	32 bits	16 bits

NOTE 1 The complete UID 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 clause 7.4.8.2

If Error_flag is not set, the response contains:

- UID (mandatory),
- DSFID if supported and present,
- AFI if supported and present, and
- System Information, see Table 58.

Table 58 — 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 59
		10	Tag sensitivity is S2. See Table 59
		11	Tag sensitivity is S3. See Table 59
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 60

Table 59 — Tag sensitivity class

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

Table 60 — 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 61.

Table 61 — 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	

7.8.8 Begin_round

Command code = '0B'

The Begin_round command, as shown in Table 62, 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 63, 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 64.

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 UID if present, unless there is no user data, in which case the tag returns the SUID

Table 62 — 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 (battery-less or battery-assisted),
- 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 63 — Begin_round response format

Preamble	Flags see 7.4.8.2	Tag type see Table 39	Battery status see Table 40	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 ie. after the 4 bit signature value starting with the field currently marked AFI.

Table 64 — 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.

7.8.9 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 65 and the other that doesn't specify the SUID, as shown in Table 66, but is used to write block data to the tag in the selected state.

Table 65 — 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 66 — 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 65 and Table 66 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 65 and Table 66 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 67.

Table 67 — 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	

7.8.10 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 68 and the other that doesn't specify the SUID, as shown in Table 69, but is used to write multiple blocks to the tag in the selected state.

Table 68 — 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 69 — 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 68 and Table 69 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) UID

First block number

Number of blocks

Block data (repeated as defined in the NOTE after Table 69).

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 70.

Table 70 — 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	

7.8.11 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, ie 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 71 and the other that doesn’t specify the SUID, as shown in Table 72, but is used to lock a block of the tag in the selected state.

Table 71— 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 72 — 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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

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 clauses 7.4.8.2, 7.4.8.3 and Table 27.

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 clause 7.4.8.2 and Table 24.

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 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 73.

Table 73 — Tag state transition for Lock_blocks

Command : Lock_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	

7.8.12 Write_AFI

Command code = '13'

On receiving the Write_AFI command, the tag shall write the AFI/ASF value into its memory.

There are two formats of the Write_AFI command, one specifying the SUID, as shown in Table 74 and the other that doesn't specify the SUID, as shown in Table 75, but is used to write the AFI/ASF of the tag in the selected state.

Table 74 — Write_AFI command format- with SUID

Protocol extension	Write_AFI	SUID flag = 1	RFU	CRC-5	SUID	AFI	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	8 bits	16 bits

Table 75 — Write_AFI command format- with no SUID

Protocol extension	Write_AFI	SUID flag = 0	RFU	CRC-5	AFI	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	16 bits

Command parameter:

(Optional) SUID

AFI

Command response:

If a tag in the selected state does not support the Write_AFI command it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Write_AFI command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Write_AFI command, and if the error flag is NOT set, the response shall be according to Table 25.

If the command specifies an SUID and the tag with the matching SUID does not support the Write_AFI command it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Write_AFI command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Write_AFI command, and if the error flag is NOT set, the response shall be according to Table 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 76.

Table 76 — Tag state transition for Write_AFI

Command : Write_AFI			
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	

7.8.13 Lock_AFI

Command code = '14'

On receiving the Lock_AFI command, the tag shall lock the AFI/ASF value permanently into its memory.

There are two formats of the Lock_AFI command, one specifying the SUID, as shown in Table 77 and the other that doesn't specify the SUID, as shown in Table 78, but is used to lock the AFI/ASF of the tag in the selected state.

Table 77 — Lock_AFI command format – with SUID

Protocol extension	Lock_AFI	SUID Flag = 1	RFU	CRC-5	SUID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	16 bits

Table 78 — Lock_AFI command format – with no SUID

Protocol extension	Lock_AFI	SUID Flag = 0	RFU	CRC-5
1 bit	6 bits	1 bit	3 bits	5 bits

Command parameter:

(Optional) SUID

Command response:

If a tag in the selected state does not support the Lock_AFI command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Lock_AFI command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Lock_AFI command, and if the error flag is NOT set, the response shall be according to Table 25.

If the command specifies an SUID and the tag with the matching SUID does not support the Lock_AFI command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_AFI command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_AFI command, and if the error flag is NOT set, the response shall be according to Table 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 79.

Table 79 — Tag state transition for Lock_AFI

Command : Lock_AFI			
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	

7.8.14 Write_DSFID command

Command code = '15'

On receiving the Write_DSFID command, the tag shall write the DSFID value into its memory.

There are two formats of the Write_DSFID command, one specifying the SUID, as shown in Table 80 and the other that doesn't specify the SUID, as shown in Table 81, but is used to write the DSFID of the tag in the selected state.

Table 80 — Write DSFID command format – with SUID

Protocol extension	Write_DSFID	SUID flag = 1	RFU	CRC-5	SUID	DSFID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	8 bits	16 bits

Table 81 — Write DSFID command format – with no SUID

Protocol extension	Write DSFID	SUID flag = 0	RFU	CRC-5	DSFID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	8 bits	16 bits

Command parameter:

(Optional) SUID

DSFID

Command response:

If a tag in the selected state does not support the Write_DSFID command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Write_DSFID command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Write_DSFID command, and if the error flag is NOT set, the response shall be according to Table 25.

If the command specifies an SUID and the tag with the matching SUID does not support the Write_DSFID command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Write_DSFID command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Write_DSFID command, and if the error flag is NOT set, the response shall be according to Table 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 82.

Table 82 — Tag state transition for Write DSFID

Command : Write_DSFID			
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	

7.8.15 Lock_DSFID

Command code = '16'

On receiving the Lock_DSFID command, the tag shall lock the DSFID value permanently into its memory.

There are two formats of the Lock_DSFID command, one specifying the SUID, as shown in Table 83 and the other that doesn't specify the SUID, as shown in Table 84, but is used to lock the DSFID of the tag in the selected state.

Table 83 — Lock DSFID command format – with SUID

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

Table 84 — Lock_DSFID command format – with no SUID

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

Command parameter:

(Optional) SUID

Command response:

If a tag in the selected state does not support the Lock_DSFID command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Lock_DSFID command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Lock_DSFID command, and if the error flag is NOT set, the response shall be according to Table 25.

If the command specifies an SUID and the tag with the matching SUID does not support the Lock_DSFID command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_DSFID command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Lock_DSFID command, and if the error flag is NOT set, the response shall be according to Table 25.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

The tag shall perform the actions specified in Table 85.

Table 85 — Tag state transition for Lock_DSFID

Command : Lock_DSFID			
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	

7.8.16 Get_blocks_lock_status

Command code = '17'

On receiving the Get_blocks_lock_status command, the tag shall send back the lock status of the requested blocks.

The Lock block number is numbered from '0 to '7', see Clause 7.8.11.

There are two formats of the Get_block_lock_status command, one specifying the SUID, as shown in Table 86 and the other that doesn't specify the SUID, as shown in Table 87, but is used to get the block lock status of the tag in the selected state.

Table 86 — Get_blocks_lock_status command format, with SUID

Protocol extension	Get_blocks_lock_status	SUID flag = 1	Lock block number	CRC-5	SUID	CRC-16
1 bit	6 bits	1 bit	3 bits	5 bits	40 bits	16 bits

Table 87 — Get_blocks_lock_status command format, with no SUID

Protocol extension	Get_blocks_lock_status	SUID flag = 0	Lock block number	CRC-5
1 bit	6 bits	1 bit	3 bits	5 bits

Command parameter:

(Optional) SUID

First block number

Number of blocks

Command response:

If a tag in the selected state does not support the Get_blocks_lock_status command it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Get_blocks_lock_status command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Get_blocks_lock_status command, and if the error flag is NOT set, the response shall be according to Table 88.

If the command specifies an SUID and the tag with the matching SUID does not support the Get_blocks_lock_status command it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Get_blocks_lock_status command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Get_blocks_lock_status command, and if the error flag is NOT set, the response shall be according to Table 88.

All other tags shall remain silent.

Response parameter:

Error_flag (and Error code if Error_flag is set), see clause 7.4.8.2

If Error_flag is not set

Table 88 — Get_blocks_lock_status response format when Error_flag is NOT set

Preamble	Flags see 7.4.8	Tag type see Table 39	Battery status see Table 40	Signature	SUID	User lock information see Table 89	Factory lock information see Table 89	CRC-16
	2 bits	1 bit	1 bit	4 bits	40 bits	32 bits	32 bits	16 bits

The Lock information comprises 2 bits per block, as defined in clause 7.3.3 and Table 17. Each 32 bit Lock information field contains the lock status of 32 consecutive blocks, starting from the first block number as defined in the command. If the memory does not contain the specified/requested block, the field is zero filled.

Table 89 — Lock information

MSB			**LSB**
b32	**b31**	**…**	**b1**
Lock info of 32nd block	Lock info of 31st block		Lock info of first block

The tag shall perform the actions specified in Table 90.

Table 90 — Tag state transition for Get-blocks_lock_status

Command: Get_blocks_lock_status			
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	

7.9 Custom commands

The format of custom command, as shown in Table 91, is generic and allows unambiguous determination of custom command codes by each tag IC manufacturer.

The custom command code is the combination of a custom command code and of the tag IC manufacturer code.

The custom command parameters definition and the state transition diagram are the responsibility of the tag manufacturer.

Table 91 — Custom command format

Protocol extension	Custom	Flags	IC mfg code	Custom command parameters
1 bit	6 bits	4 bits	8 bits	Custom defined

Command parameter:

IC manufacturer code according to ISO/IEC 7816-6.

Command response:

If a tag in the selected state does not support the Custom command, it shall return the error code "command is not supported", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If a tag in the selected state does support the Custom command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If a tag in the selected state does support the Custom command, and if the error flag is NOT set, the response shall be according to Table 92.

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", error code 1, see clauses 7.4.8.2, 7.4.8.3 and Table 27.

If the command specifies an SUID and the tag with the matching SUID does support the Custom command, and if the error flag is set, the response shall be according to clause 7.4.8.2 and Table 24.

If the command specifies an SUID and the tag with the matching SUID does support the Custom command, and if the error flag is NOT set, the response shall be according to Table 92.

All other tags shall remain silent.

Table 92 — Custom response format when Error_flag is NOT set

Flags see clause 7.4.8.2	Custom response parameters
2 bits	Custom defined

Response parameter:

Error_flag (and code if Error_flag is set), see clause 7.4.8.2

If Error_flag is not set

Custom parameters

7.10 Proprietary commands

The format of these commands is not defined in this part of ISO/IEC 18000.

8 Type B

8.1 Physical layer and data coding

8.1.1 Forward link

8.1.1.1 Carrier modulation

The data transmission from the interrogator to the tag is achieved by modulation of the carrier (ASK). The data coding is performed by generating pulses that create a Manchester coding, as shown in Figure 20, Figure 21, Table 93 and Table 94.

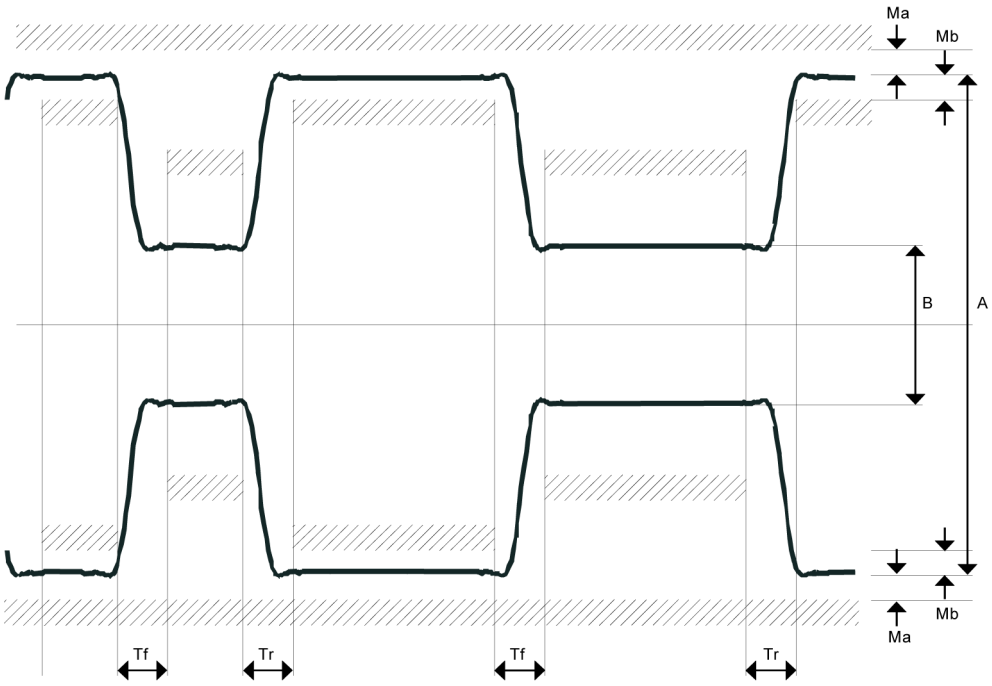


Figure 20 — 100 % modulation (example of 40 kbit/s signal)

Table 93 — Parameter for 100% modulation

Parameter	Minimum	Nominal	Maximum
M = (A-B)/(A+B)	90	100	100
Ma	0		0.03 (A-B)
Mb	0		0.03 (A-B)
Tr	0 µs	1.8 µs	0.1 / $f_{Datarate}$
Tf	0 µs	1.8 µs	0.1 / $f_{Datarate}$

NOTE Tr and Tf measured from 10% (A-B) to 90% (A-B)

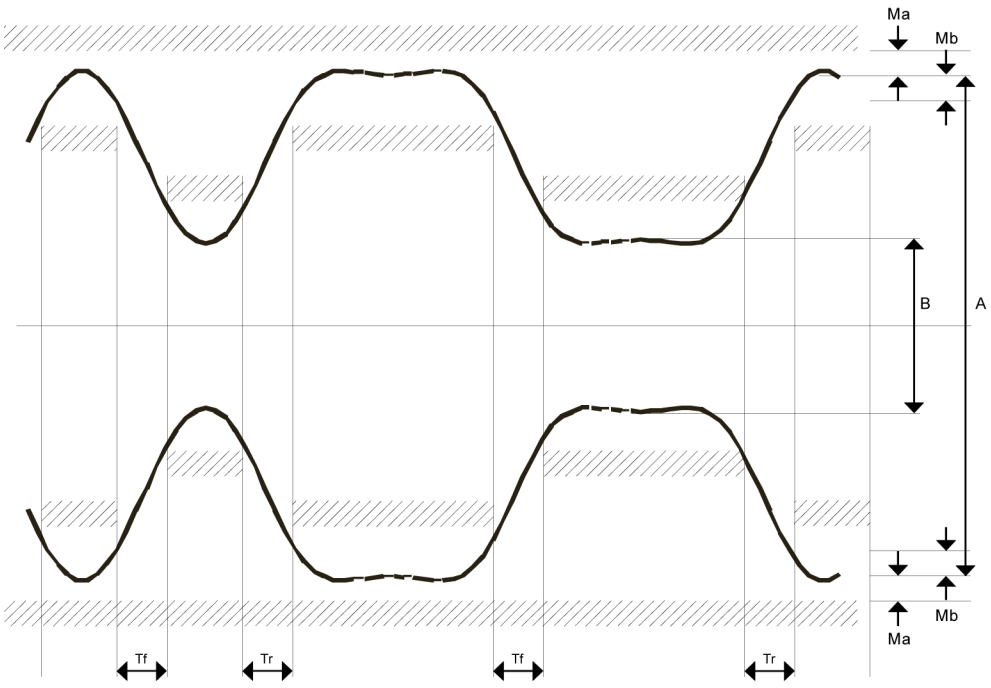


Figure 21 — 18% modulation (example of 8 kbit/s signal)

Table 94 — Parameter for 18% modulation

Parameter	Minimum	Nominal	Maximum
M = (A-B)/(A+B)	15%	18%	20%
Ma	0		0.05 (A-B)
Mb	0		0.05 (A-B)
Tr	0 µs		0.17 / $f_{Datarate}$
Tf	0 µs		0.17 / $f_{Datarate}$

NOTE Tr and Tf measured from 10% (A-B) to 90% (A-B)

8.1.1.2 Bit coding of forward link fields

Data is Manchester encoded as per Figure 22.

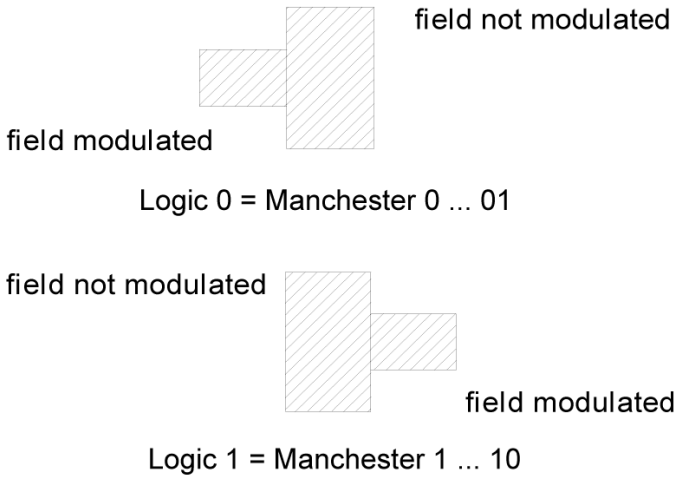


Figure 22 — Forward link bit coding

8.1.2 Return link

See clause 6.5.

8.1.3 Protocol concept

Data is encoded and presented in slightly different ways in the constituent fields. For interrogator-to-tag communication (forward link), data is sent using an on-off key format. The radio frequency field being on corresponds to 1, while the radio frequency field being off corresponds to 0. The modulation index specification is defined in clause 8.1.1.1. In the case of Manchester coding a Manchester 1 is a 1 to 0 transition, while a Manchester 0 is a 0 to 1 transition.

For tag-to-interrogator communication (return link), data is sent using backscatter techniques. This requires that the interrogator provide steady power to the tag during the return link. While the interrogator powers the tag, the tag shall change alternately the effective impedance of the tag front end and thus changing the overall radio frequency reflectivity of the tag as seen by the interrogator. During this time, the interrogator shall not modulate the carrier. During the WAIT field (when tags write data into their memory), the interrogator shall also provide steady power to the tag, and shall not modulate the carrier. The transmission protocol defines the mechanism to exchange instructions and data between the interrogator and the tag, in both directions.

It is based on the concept of "interrogator talks first".

This means that any tag shall not start transmitting (modulating) unless it has received and properly decoded an instruction sent by the interrogator.

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 clause 8.2.7

Each command and each response are contained in a frame. The respective frames are specified in clauses 8.1.4 and 8.1.5. Each command consists of the following fields:

- Preamble Detect (no modulation of the RF carrier),
- Preamble,
- Delimiter,

- Command code,
- Parameter fields (depending on the command),
- Application data fields (depending on the command), and
- CRC-16.

Each response consists of the following fields:

- Quiet (no modulation of the RF carrier),
- Return Preamble,
- Application data fields, and
- CRC-16.

The protocol is bit-oriented. The number of bits transmitted in a frame is a multiple of eight (8), i.e. an integer number of bytes. However, the frame itself is not based on an integer number of bytes, to support the frame detection.

In all byte fields, the MSB shall be transmitted first, proceeding to the LSB. In all word (8-byte) data fields, the MSByte shall be transmitted first.

The MSByte shall be the byte at the specified address. The LSByte shall be the byte at the specified address plus 7 (i.e., bytes are transmitted in incrementing address order).

The byte significance is relevant to data transmission and to the GROUP_SELECT and GROUP_UNSELECT greater than and less than comparisons.

The MSByte of the byte mask shall correspond to the most significant data byte, the byte at the specified address.

Word (8-byte) addresses are not required to be on an 8-word boundary and may be on any byte boundary.

RFU bits and bytes shall be set to zero (0).

8.1.4 Command format

8.1.4.1 Command format general

The command frame, as shown in Figure 23, consist of the following fields:

- Preamble Detect ,
- Preamble,
- Delimiter,
- Command,
- Parameter (and data fields), and
- CRC-16.

Preamble Detect	Preamble	Delimiter	Command	Parameter	Data	CRC-16

Figure 23 — General command format

8.1.4.2 PREAMBLE DETECT field

The preamble detect field consist of a steady carrier (no modulation) during a time of at least 400 µs. This corresponds to 16 bits for a communication rate of 40 kbit/s.

8.1.4.3 PREAMBLE

The preamble is equivalent to 9 bits of Manchester 0 in NRZ format.

010101010101010101

8.1.4.4 Delimiters

8.1.4.4.1 Delimiters general

Four delimiters are defined.

8.1.4.4.2 Start delimiter 1

In NRZ format; includes Manchester errors; spaces ignored

11 00 11 10 10 - Delimiter 1

8.1.4.4.3 Start delimiter 2

In NRZ format; includes Manchester errors; spaces ignored

01 01 11 00 11 - Delimiter 2

Reserved for future use

8.1.4.4.4 Start delimiter 3

In NRZ format; includes Manchester errors; spaces ignored

00 11 10 01 01 - Delimiter 3

Reserved for future use

8.1.4.4.5 Start delimiter 4

In NRZ format; includes Manchester errors; spaces ignored

11 01 11 00 10 1 - Delimiter 4

Delimiter4 supports all commands as delimiter 1, however the return data rate is 4 times the forward link data rate. The supported data rates are defined in clause 5.2.

8.1.4.5 CRC-16

See clauses 6.5.7.3, 6.5.7.4 and Annex A.

8.1.5 Response format

8.1.5.1 Response format general

The response, as shown in Figure 24, consists of the following fields:

- Quiet,
- Return Preamble,
- Data fields, and
- CRC-16.

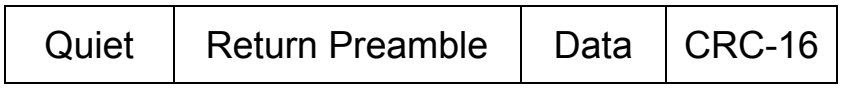


Figure 24 — General response format

The tag shall use a backscatter technique to communicate data to the interrogator. The interrogator shall be steadily powering the tag as well as listening to the tag response throughout the tag-to-interrogator (backscatter) communication. This applies to all fields in the return link.

8.1.5.2 QUIET

The tag shall not backscatter for 16 * $T_{\text{return data rate}}$ – 0,75 * $T_{\text{forward data rate}}$. The duration of the quiet time is determined by the communication speed of the return link.

8.1.5.3 CRC-16

See clauses 6.5.7.3, 6.5.7.4 and Annex A.

8.1.6 WAIT

When a tag receives a write command, it shall execute a write operation. (The details of the conditions under which a write will occur are described in clause 8.2.7.9.11. If a write operation is executed, the final field in the overall field sequence shall always be WAIT.

During the WAIT field, when the tag is writing data into the EEPROM, the interrogator must steadily power the tag. On-off key data shall not be sent during this time.

8.1.7 Examples of a command packet

Examples of command packets are given in Figure 25 and Figure 26.

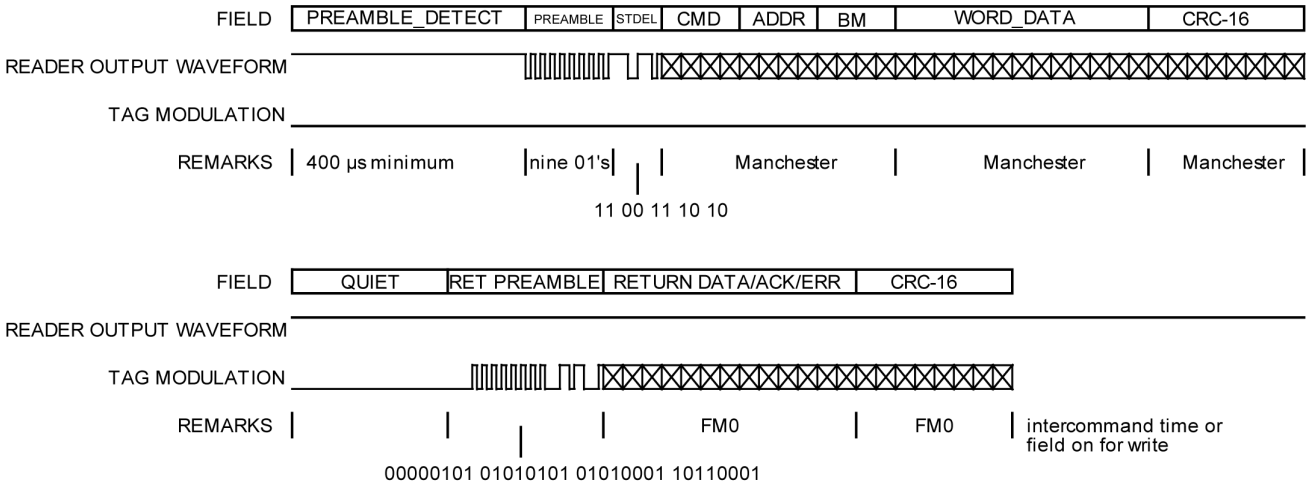


Figure 25 — Sample command/response packets for GROUP_SELECT (40 kbit/s on forward and return link)

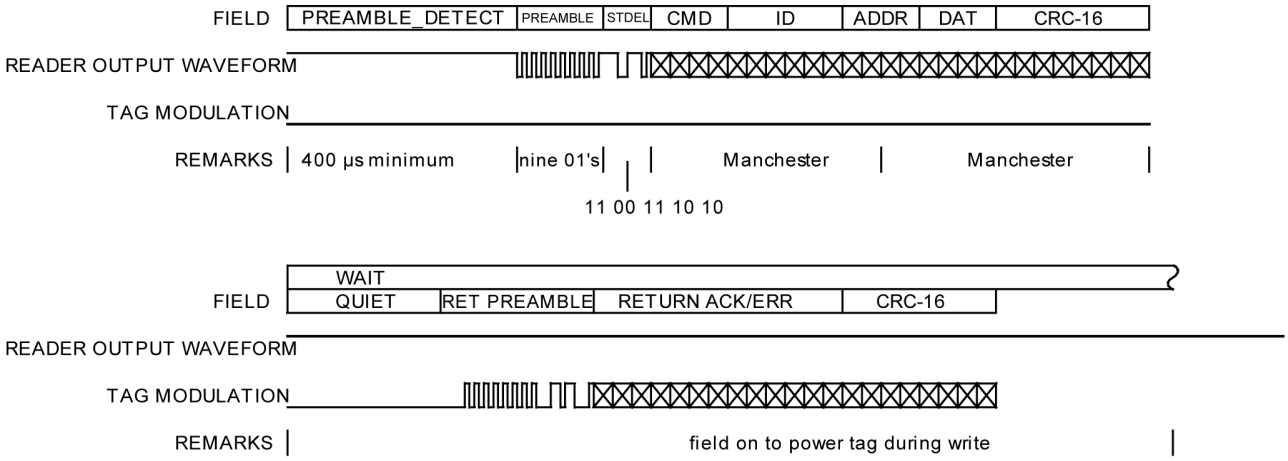


Figure 26 — Sample command/response packets for WRITE (40 kbit/s on forward and return link)

8.1.8 Communication sequences at packet level

Figure 27 and Figure 28 show examples of communication sequences at the packet level. Figure 27 depicts a packet sequence that includes a write command. The sequence includes a wait for write time, which provides the necessary time for the chip to complete its write operation. In addition, following the wait for write time, the interrogator issues a tag resync signal. This signal is composed of 10 consecutive 01 signals. The purpose of the tag resync signal is to initialise the tag data recovery circuitry. It is required after a write because the interrogator may output spurious edges during the wait for write time. Without the tag resync, tags may mis-calibrate as a result of the spurious signals that may be generated.

Figure 28 depicts a packet sequence in which a frequency hop between commands is included. The tag resync signal is again required after the hop because spurious signals may be generated during the hop time.

In order to ensure that tags do not get confused, frequency hops between command and response should be avoided.

Action	COMMAND	RESPONSE	WAIT FOR WRITE	TAG RESYNC	COMMAND	RESPONSE
Component execution action	Interrogator	Tag	Interrogator	Interrogator	Interrogator	Tag
Notes	---	---	15 ms minimum	ten 01's	---	---

Figure 27 — Command sequence (including a write) with no hopping

Action	COMMAND	RESPONSE	HOP	TAG RESYNC	COMMAND	RESPONSE
Component execution action	Interrogator	Tag	Interrogator	Interrogator	Interrogator	Tag
Notes	---	---	< 26 µs	ten 01's	---	---

Figure 28 — Command sequence with a hop between response and next command

8.2 Btree protocol and collision arbitration

8.2.1 Definition of data elements, bit and byte ordering

8.2.1.1 Unique ID

See Annex B and Annex C.

8.2.1.2 CRC-16

See Annex A.

8.2.1.3 FLAGS

8.2.1.3.1 FLAGS general

The tag shall support a field of 8 flags. This field is called FLAGS, and is shown in Table 95.

Table 95 — FLAGS

Bit	Name
FLAG1 (LSB)	DE_SB (Data_Exchange Status Bit)
FLAG2	WRITE_OK
FLAG3	BATTERY_POWERED
FLAG4	BATTERY_OK
FLAG5	0 (RFU)
FLAG6	0 (RFU)
FLAG7	0 (RFU)
FLAG8 (MSB)	0 (RFU)

8.2.1.3.2 Data exchange status bit (DE_SB)

The tag shall set this bit when the tag goes into the DATA_EXCHANGE state and keeps it set unless it moves into the POWER-OFF state.

When the DE_SB is set and the tag comes into the POWER-OFF state, then the tag shall trigger a timer that will reset the DE_SB bit after t_{DE_SB}.

t_{DE_SB} shall be at least 2 seconds in the temperature range –30 °C to 60 °C.

t_{DE_SB} shall be at least 4 seconds in the temperature range 0 °C to 50 °C.

When the tag receives the INITIALIZE command, then it shall reset the DE_SB immediately.

8.2.1.3.3 WRITE_OK

The WRITE_OK bit shall be set after a successful write access to the memory. (e.g., WRITE, LOCK)

The WRITE_OK bit is cleared after execution of the command following the write command.

8.2.1.3.4 BATTERY_POWERED

The BATTERY_POWERED bit shall be set when the tag should have a battery. It shall be cleared for passive tags.

8.2.1.3.5 BATTERY_OK

The BATTERY_OK bit shall be set when the battery has enough power to support the tag. It shall be cleared for passive tags.

NOTE BATTERY_POWERED indicates whether the tag should have a battery, while BATTERY_OK reports the status of the battery. BATTERY_POWERED could be hardwired.

8.2.2 Tag memory organisation

The functional memory shall be organised in blocks of one byte.

Up to 256 blocks of one byte can be addressed.

This leads to a maximum memory capacity of up to 2 kbits.

NOTE This structure allows future extensions of the maximum memory capacity, by the use of additional commands to be defined, when required.

8.2.3 Block security status

Each byte shall have a corresponding lock bit. The lock bits may be locked by use of the LOCK command. The status of the lock bit may be read by the QUERY_LOCK command. The tag shall not be allowed to reset any lock bit after leaving the final production site. In most cases this is the production site that defines the unique ID.

8.2.4 Overall protocol description, Btree protocol

8.2.4.1 Tag states

The tag has four major states, as shown in Figure 29:

POWER-OFF The tag is in the POWER-OFF state when the interrogator cannot activate it. (For battery-assisted tags, it means that the level of RF excitation is insufficient to turn on the tag circuits.)

READY The tag is in the READY state when the interrogator first powers it up.

ID The tag is in the ID state when it is trying to identify itself to the interrogator.

DATA_EXCHANGE The tag is in the DATA_EXCHANGE state, when it is known to the interrogator and was selected.

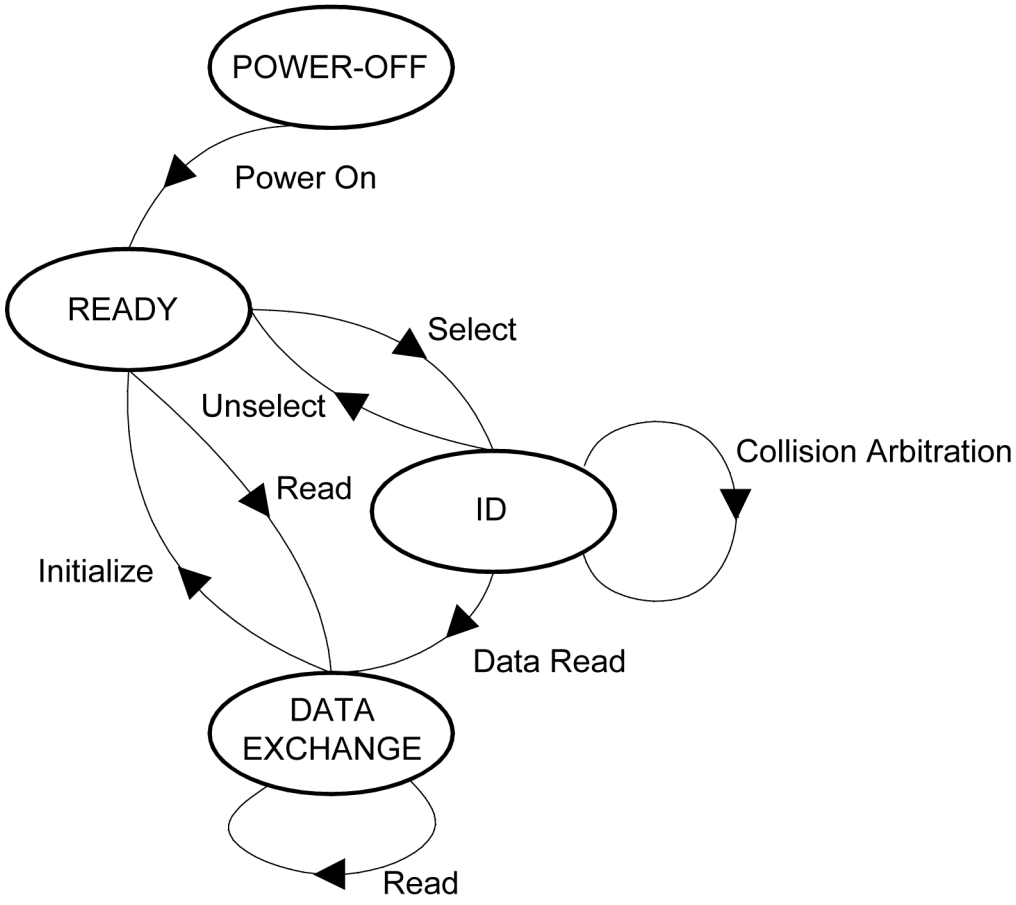


Figure 29 — State diagram

NOTE This diagram does not show that the tag goes into POWER-OFF, state from all other states when the interrogator field is off and the tag operation is no longer supported by the tag power buffer.

The state diagram only shows an overview of the possible transitions. Details are specified in Table 97.

Power-On:

State change when interrogator field is turned on.

Select:

State change due to selection of tag by GROUP_SELECT or READ commands

Unselect:

State change due to deselection of tag by GROUP_UNSELECT commands or INITIALIZE command

Collision_Arbitration:

No state change during collision arbitration until a single tag is identified.

Data_Read:

State change due to first read access in collision arbitration process

Read:

State change due to read access independent of collision arbitration process.

Initialize:

State change due to deselection of tag by INITIALIZE command

The transition between these states is specified in Table 97.

8.2.4.2 Detailed command processing

As shown in Table 96, commands shall be active in states marked with "X" and neither causes a state change, nor cause a response in the other states.

Table 96 — Detailed command processing

COMMAND	States		
	READY	ID	DATA EXCHANGE
GROUP_SELECT_EQ	X	X	
GROUP_SELECT_NE	X	X	
GROUP_SELECT_GT	X	X	
GROUP_SELECT_LT	X	X	
GROUP_SELECT_EQ_FLAGS	X	X	
GROUP_SELECT_NE_FLAGS	X	X	
GROUP_UNSELECT_EQ		X	
GROUP_UNSELECT_NE		X	
GROUP_UNSELECT_GT		X	
GROUP_UNSELECT_LT		X	
GROUP_UNSELECT_EQ_FLAGS		X	
GROUP_UNSELECT_NE_FLAGS		X	
MULTIPLE_UNSELECT		X	
FAIL		X	
SUCCESS		X	
RESEND		X	
INITIALIZE	X	X	X
READ	X	X	X
DATA_READ		X	X
READ_VERIFY	X	X	X
READ_VERIFY_4BYTE	X	X	X
WRITE	X	X	X
WRITE4BYTE	X	X	X
WRITE4BYTE_MULTIPLE		X	X
WRITE_MULTIPLE		X	X
LOCK			X
QUERY_LOCK	X	X	X
FAIL_O		X	
SUCCESS_O		X	
DATA_READ_O		X	X
READ_FLAGS	X	X	X
READ_VARIABLE	X	X	X
READ_PORT	X	X	X
READ_UNADDRESSED	X	X	X
RESEND_O		X	

Table 97 — State transition table

Current state	Command	Condition	New state
POWER-OFF	ANY COMMAND		POWER OFF
POWER-OFF	“Power up”		READY
READY	GROUP_SELECT_EQ	≠	READY
READY	GROUP_SELECT_NE	=	READY
READY	GROUP_SELECT_GT	≤	READY
READY	GROUP_SELECT_EQ_FLAGS	flag not set	READY
READY	GROUP_SELECT_NE_FLAGS	flag set	READY
READY	GROUP_SELECT_LT	≥	READY
READY	GROUP_SELECT_EQ	=	ID
READY	GROUP_SELECT_NE	≠	ID
READY	GROUP_SELECT_GT	>	ID
READY	GROUP_SELECT_LT	<	ID
READY	GROUP_SELECT_EQ_FLAGS	flag set	ID
READY	GROUP_SELECT_NE_FLAGS	flag not set	ID
READY	INITIALIZE		READY
READY	READ	ID no match	READY
READY	READ	ID match	DATA_EXCHANGE
READY	READ_VERIFY	ID no match or not WRITE_OK	READY
READY	READ_VERIFY	ID match and WRITE_OK	DATA_EXCHANGE
READY	READ_VERIFY_4BYTE	ID no match or not WRITE_OK	READY
READY	READ_VERIFY_4BYTE	ID match and WRITE_OK	DATA_EXCHANGE
READY	WRITE	ID no match	READY
READY	WRITE	ID match	DATA_EXCHANGE
READY	WRITE4BYTE	ID no match	READY
READY	WRITE4BYTE	ID match	DATA_EXCHANGE
READY	QUERY_LOCK	ID no match	READY
READY	QUERY_LOCK	ID match	DATA_EXCHANGE
READY	READ_FLAGS	ID no match	READY
READY	READ_FLAGS	ID match	DATA_EXCHANGE
READY	READ_VARIABLE	ID no match	READY
READY	READ_VARIABLE	ID match	DATA_EXCHANGE
READY	READ_PORT	ID no match	READY
READY	READ_PORT	ID match	DATA_EXCHANGE
READY	READ_UNADDRESSED		DATA_EXCHANGE
ID	GROUP_UNSELECT_EQ	≠	ID
ID	GROUP_UNSELECT_NE	=	ID
ID	GROUP_UNSELECT_GT	≤	ID

Table 97 *(continued)*

Current state	Command	Condition	New state
ID	GROUP_UNSELECT_LT	≥	ID
ID	GROUP_UNSELECT_EQ_FLAGS	flag not set	ID
ID	GROUP_UNSELECT_NE_FLAGS	flag set	ID
ID	GROUP_UNSELECT_EQ	=	READY
ID	GROUP_UNSELECT_NE	≠	READY
ID	GROUP_UNSELECT_GT	>	READY
ID	GROUP_UNSELECT_LT	<	READY
ID	GROUP_UNSELECT_EQ_FLAGS	flag set	READY
ID	GROUP_UNSELECT_NE_FLAGS	flag not set	READY
ID	MULTIPLE_UNSELECT	≠ or not WRITE_OK	ID
ID	MULTIPLE_UNSELECT	= and WRITE_OK	READY
ID	GROUP_SELECT_EQ		ID
ID	GROUP_SELECT_NE		ID
ID	GROUP_SELECT_GT		ID
ID	GROUP_SELECT_LT		ID
ID	GROUP_SELECT_EQ_FLAGS		ID
ID	GROUP_SELECT_NE_FLAGS		ID
ID	FAIL		ID
ID	SUCCESS		ID
ID	RESEND		ID
ID	INITIALIZE		READY
ID	READ	ID no match	ID
ID	READ	ID match	DATA_EXCHANGE
ID	DATA_READ	ID no match	ID
ID	DATA_READ	ID match	DATA_EXCHANGE
ID	READ_VERIFY	ID no match or not WRITE_OK	ID
ID	READ_VERIFY	ID match and WRITE_OK	DATA_EXCHANGE
ID	READ_VERIFY_4BYTE	ID no match or not WRITE_OK	ID
ID	READ_VERIFY_4BYTE	ID match and WRITE_OK	DATA_EXCHANGE
ID	WRITE	ID no match	ID
ID	WRITE	ID match	DATA_EXCHANGE
ID	WRITE4BYTE	ID no match	ID
ID	WRITE4BYTE	ID match	DATA_EXCHANGE
ID	WRITE_MULTIPLE		ID
ID	WRITE4BYTE_MULTIPLE		ID
ID	QUERY_LOCK	ID no match	ID
ID	QUERY_LOCK	ID match	DATA_EXCHANGE

Table 97 *(continued)*

Current state	Command	Condition	New state
ID	RESEND_O		ID
ID	READ_FLAGS	ID no match	ID
ID	READ_FLAGS	ID match	DATA_EXCHANGE
ID	READ_VARIABLE	ID no match	ID
ID	READ_VARIABLE	ID match	DATA_EXCHANGE
ID	READ_PORT	ID no match	ID
ID	READ_PORT	ID match	DATA_EXCHANGE
ID	FAIL_O		ID
ID	SUCCESS_O		ID
ID	DATA_READ_O	ID no match	ID
ID	DATA_READ_O	ID match	DATA_EXCHANGE
ID	READ_UNADDRESSED		DATA_EXCHANGE
DATA_EXCHANGE	INITIALIZE		READY
DATA_EXCHANGE	READ		DATA_EXCHANGE
DATA_EXCHANGE	DATA_READ		DATA_EXCHANGE
DATA_EXCHANGE	READ_VERIFY		DATA_EXCHANGE
DATA_EXCHANGE	READ_VERIFY_4BYTE		DATA_EXCHANGE
DATA_EXCHANGE	WRITE		DATA_EXCHANGE
DATA_EXCHANGE	WRITE4BYTE		DATA_EXCHANGE
DATA_EXCHANGE	WRITE4BYTE_MULTIPLE		DATA_EXCHANGE
DATA_EXCHANGE	WRITE_MULTIPLE		DATA_EXCHANGE
DATA_EXCHANGE	LOCK		DATA_EXCHANGE
DATA_EXCHANGE	QUERY_LOCK		DATA_EXCHANGE
DATA_EXCHANGE	READ_UNADDRESSED		DATA_EXCHANGE
DATA_EXCHANGE	DATA_READ_O		DATA_EXCHANGE
DATA_EXCHANGE	READ_FLAGS		DATA_EXCHANGE
DATA_EXCHANGE	READ_VARIABLE		DATA_EXCHANGE
DATA_EXCHANGE	READ_PORT		DATA_EXCHANGE

8.2.5 Collision arbitration

8.2.5.1 Collision arbitration general

The interrogator may use the GROUP_SELECT and GROUP_UNSELECT commands to define all or a subset of tags in the field to participate in the collision arbitration. It then may use the identification commands to run the collision arbitration algorithm.

For the collision arbitration the tag shall support two pieces of hardware on the tag:

- An 8-bit counter COUNT, and
- A random generator (with two possible values 0 or 1).

In the beginning, a group of tags are moved to the ID state by GROUP_SELECT commands and shall set their internal counters to 0. Subsets of the group may be unselected by GROUP_UNSELECT commands back to the READY state. Other groups can be selected before the identification process begins. Simulation results show no advantage in identifying one large group or a few smaller groups.

After above described selection, the following loop should be performed:

1) All tags in the ID state with the counter COUNT at 0 shall transmit their ID. This set initially includes all the selected tags.

2) If more than one tag transmitted, the interrogator receives an erroneous response. The FAIL command shall be sent.

3) All tags receiving a FAIL command with COUNT not equal to 0 shall increment COUNT. That is, they move further away from being able to transmit.

   All tags receiving FAIL command with a count of 0 (those that just transmitted) shall generate a random number. Those that roll a 1 shall increment COUNT and shall not transmit. Those that roll a zero shall keep COUNT at zero and shall send their UID again.

One of four possibilities now occurs:

4) If more than one tag transmits, the FAIL step 2 repeats. (Possibility 1)

5) If all tags roll a 1, none transmits. The interrogator receives nothing. It sends the SUCCESS command. All the counters decrement, and the tags with a count of 0 transmit. Typically, this returns to step 2. (Possibility 2)

6) If only one tag transmits and the ID is received correctly, the interrogator shall send the DATA_READ command with the ID. If the DATA_READ command is received correctly, that tag shall move to the DATA_EXCHANGE state and shall transmit its data.

   The interrogator shall sends SUCCESS. All tags in the ID state shall decrement COUNT.

7) If only one tag has a count of 1 and transmits, step 5 or 6 repeats. If more than one tag transmits, step 2 repeats. (Possibility 3)

8) If only one tag transmits and the ID is received with an error, the interrogator shall send the RESEND command. If the ID is received correctly, step 5 repeats. If the ID is received again some variable number of times (this number can be set based on the level of error handling desired for the system), it is assumed that more than one tag is transmitting, and step 2 repeats. (Possibility 4)

8.2.5.2 Special collision arbitration

In the case that parts of the user data are unique, or the probability of duplicated information is sufficient low, the commands FAIL_O, SUCCESS_O and DATA_READ_O may be used for collision arbitration. While the algorithm is equal to commands without _O, the ID used for the algorithm is either a 32 bit or 64 bit ID beginning at memory address ‘14’.

Additionally to the already mentioned use of GROUP_SELECT and GROUP_UNSELECT these commands may be used in combination with the 32-bit ID option of the _O command. This means in the case that a GROUP_SELECT only selects those tags that have all zeros in the upper 32 bits of a 64 bit UID and after handling those checks whether no tags with having not all zeros in the upper 32 bits of a 64 bit UID are still left.

8.2.6 Commands

Commands are divided into four functional groups:

- Selection commands,
- Identification commands,
- Data transfer commands, and
- Multiple commands.

Further, commands, as shown in Table 98, have one of the following types:

- Mandatory,
- Optional,
- Custom, and
- Proprietary.

Table 98 — Command classes

Code	Class	Number of possible codes
'00' - '0A', '0C', '15', '1E' - '3F'.	Mandatory	47
'0B', '0D' - '0F', '11' - '13', '17' - '1D', '40' - '9F'	Optional	110
'A0' - 'DF'	Custom	64
'10', '14', '16' 'E0' - 'FF'	Proprietary	35

8.2.7 Command types

8.2.7.1 Command types general

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

8.2.7.2 Mandatory

The command codes range from '00' to '0A', '0C', '15' and '1E' to '3F'.

A Mandatory command shall be supported by all tags that claim to be compliant. Interrogators which claim compliance shall support all mandatory commands. Mandatory commands shall be implemented as specified in this part of ISO/IEC 18000.

8.2.7.3 Optional

The command codes range from '0B', '0D' to '0F', '11' to '13', '17' to '1D' and from '40' to '9F'.

Optional commands are commands that are specified within this part of ISO/IEC 18000. Interrogators shall be technically capable of performing all optional commands that are specified this part of ISO/IEC 18000 (although need not be set up to do so). Tags may or may not support optional commands.

If an optional command is used, it shall be implemented in the manner specified in this part of ISO/IEC 18000.

If the tag does not support an optional command, it shall remain silent.

NOTE The command whose code ranges from ‘0B, 0D to 13 and '17' to '1D' are optional and not essential to operate the tag. However, their support by the tag is recommended for appropriate performance. To reflect this, they are reported as “recommended” in Table 99.

8.2.7.4 Custom

The command codes range from 'A0' to 'DF'.

Tags support them, at their option, to implement manufacturer specific functions. The only fields that can be customized are the parameters 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. Custom commands may be enabled by this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000.

Custom commands may be enabled by this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000dard. A custom 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.

If the tag does not support a custom command it shall remain silent.

8.2.7.5 Proprietary

The command codes are ‘10’, ’14’, ‘16’ and the range from 'E0' to 'FF'.

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. It is allowed that these commands are disabled after IC and/or tag manufacturing.

Proprietary commands may be enabled by this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000.

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.

8.2.7.6 Command codes and format

8.2.7.6.1 Command codes and format general

Command codes and formats are shown in Table 99.

Table 99 — Command codes and format

Command code	Type	Command name	Parameters		
'00'	Mandatory	GROUP_SELECT_EQ	ADDRESS	BYTE_MASK	WORD_DATA
'01'	Mandatory	GROUP_SELECT_NE	ADDRESS	BYTE_MASK	WORD_DATA
'02'	Mandatory	GROUP_SELECT_GT	ADDRESS	BYTE_MASK	WORD_DATA
'03'	Mandatory	GROUP_SELECT_LT	ADDRESS	BYTE_MASK	WORD_DATA
'04'	Mandatory	GROUP_UNSELECT_EQ	ADDRESS	BYTE_MASK	WORD_DATA
'05'	Mandatory	GROUP_UNSELECT_NE	ADDRESS	BYTE_MASK	WORD_DATA
'06'	Mandatory	GROUP_UNSELECT_GT	ADDRESS	BYTE_MASK	WORD_DATA
'07'	Mandatory	GROUP_UNSELECT_LT	ADDRESS	BYTE_MASK	WORD_DATA
'08'	Mandatory	FAIL	none	none	none
'09'	Mandatory	SUCCESS	none	none	none
'0A'	Mandatory	INITIALIZE	none	none	none
'0B'	Recommended	DATA_READ	ID	ADDRESS	none
'0C'	Mandatory	READ	ID	ADDRESS	none
'0D'	Recommended	WRITE	ID	ADDRESS	BYTE_DATA
'0E'	Recommended	WRITE_MULTIPLE	none	ADDRESS	BYTE_DATA
'0F'	Recommended	LOCK	ID	ADDRESS	none
'10'	Proprietary	IC manufacturer dependant			
'11'	Recommended	QUERY_LOCK	ID	ADDRESS	none
'12'	Recommended	READ_VERIFY	ID	ADDRESS	none
'13'	Recommended	MULTIPLE_UNSELECT	ADDRESS	BYTE_DATA	none
'14'	Proprietary	IC manufacturer dependant			
'15'	Mandatory	RESEND	none	none	none
'16'	Proprietary	IC manufacturer dependant			
'17'	Recommended	GROUP_SELECT_EQ_FLAGS	none	BYTE_MASK	BYTE_DATA
'18'	Recommended	GROUP_SELECT_NE_FLAGS	none	BYTE_MASK	BYTE_DATA
'19'	Recommended	GROUP_UNSELECT_EQ_FLAGS	none	BYTE_MASK	BYTE_DATA
'1A'	Recommended	GROUP_UNSELECT_NE_FLAGS	none	BYTE_MASK	BYTE_DATA
'1B'	Recommended	WRITE4BYTE	ID	ADDRESS	BYTE_MASK | 4BYTE_DATA
'1C'	Recommended	WRITE4BYTE_MULTIPLE	ADDRESS	BYTE_MASK	4BYTE_DATA
'1D'	Recommended	READ_VERIFY_4BYTE	ID	ADDRESS	none
'1E'-'3F'	Mandatory	RFU			
'40', '41'	Optional	FAIL_O	none	none	none
'42', '43'	Optional	SUCCESS_O	none	none	none
'44', '45'	Optional	DATA_READ_O	ID	ADDRESS	none
'46', '47'	Optional	RESEND_O	none	none	none

Table 99 *(continued)*

Command code	Type	Command name	Parameters		
'48' – '4F'	Optional	RFU			
'50'	Optional	READ_FLAGS	ID	ADDRESS	none
'51'	Optional	READ_VARIABLE	ID	ADDRESS	LENGTH
'52'	Optional	READ_PORT	ID	ADDRESS	none
'53'	Optional	READ_UNADDRESSED		ADDRESS	none
'54' – '9F'	Optional	RFU			
'A0' – 'DF'	Custom	IC Manufacturer dependent			
'E0' – 'FF'	Proprietary	IC Manufacturer dependent			

8.2.7.6.2 Command Fields

Command fields are shown in Table 100.

Table 100 — Command fields

Field name	Field size
COMMAND	1 byte
ADDRESS	1 byte
BYTE_MASK	1 byte
ID	8 bytes
WORD_DATA	8 bytes
BYTE_DATA	1 byte
4BYTE_DATA	4 bytes
LENGTH	1 byte
CRC-16	2 bytes

8.2.7.6.3 Tag responses

Tag responses are shown in Table 101.

Table 101 — Tag responses

Response code	Response name	Response size
'00'	ACKNOWLEDGE	1 byte
	ACKNOWLEDGE_NOK	1byte
'01'	ACKNOWLEDGE_OK	1byte
'FE'	ERROR_NOK	1byte
'FF'	ERROR	1byte
	ERROR_OK	1byte
n/a	WORD_DATA	8 bytes
N/a	VARIABLE DATA	LENGTH bytes
n/a	BYTE_DATA	1byte
	CRC-16	2 bytes
	ID	8 bytes

8.2.7.7 Selection commands

8.2.7.7.1 Selection commands general

Selection commands define a subset of tags in the field to be identified or written to and may be used as part of the collision arbitration.

8.2.7.7.2 Data comparison for selection command on memory

Each select command of the commands

- GROUP_SELECT_EQ,
- GROUP_SELECT_NE,
- GROUP_SELECT_GT,
- GROUP_SELECT_LT,
- GROUP_UNSELECT_EQ,
- GROUP_UNSELECT_NE,
- GROUP_UNSELECT_GT, or
- GROUP_UNSELECT_LT,

has 3 arguments (parameter and data)

- ADDRESS,
- BYTE_MASK, and
- WORD_DATA,

and the tag shall make one of 4 possible comparisons:

- EQ M EQUAL D,
- NE M NOT EQUAL D,
- GT M GREATER THAN D, or
- LT M LOWER THAN D.

The arguments of the comparison are shown in Table 102.

Table 102 — Comparison arguments

M7 (MSB)	M6	M5	M4	M3	M2	M1	M0 (LSB)
Tag memory content at ADDRESS+0	Tag memory content at ADDRESS+1	Tag memory content at ADDRESS+2	Tag memory content at ADDRESS+3	Tag memory content at ADDRESS+4	Tag memory content at ADDRESS+5	Tag memory content at ADDRESS+6	Tag memory content at ADDRESS+7

$M = M0 + M1 * 2^{8} + M2 * 2^{16} + M3 * 2^{24} + M4 * 2^{32} + M5 * 2^{40} + M6 * 2^{48} + M7 * 2^{56}$

and the arguments of the command, are shown in Table 103.

Table 103 — Command arguments

D7 (MSB)	D6	D5	D4	D3	D2	D1	D0 (LSB)
First byte after command	…						Last byte after command

$D = D0 + D1 * 2^{8} + D2 * 2^{16} + D3 * 2^{24} + D4 * 2^{32} + D5 * 2^{40} + D6 * 2^{48} + D7 * 2^{56}$

The argument BYTE_MASK defines what bytes to be considered for comparison, and are shown in Table 104.

Table 104 — Data masking for Group_Select and Group_Unselect commands

BYTE_MASK	WORD_DATA
Bit 7 (MSB) is set	Consider D7 and M7 for comparison
Bit 6 is set	Consider D6 and M6 for comparison
Bit 5 is set	Consider D5 and M5 for comparison
Bit 4 is set	Consider D4 and M4 for comparison
Bit 3 is set	Consider D3 and M3 for comparison
Bit 2 is set	Consider D2 and M2 for comparison
Bit 1 is set	Consider D1 and M1 for comparison
Bit 0 (LSB) is set	Consider D0 and M0 for comparison
Bit 7 (MSB) is cleared	Ignore D7 and M7 for comparison
Bit 6 is cleared	Ignore D6 and M6 for comparison
Bit 5 is cleared	Ignore D5 and M5 for comparison
Bit 4 is cleared	Ignore D4 and M4 for comparison
Bit 3 is cleared	Ignore D3 and M3 for comparison
Bit 2 is cleared	Ignore D2 and M2 for comparison
Bit 1 is cleared	Ignore D1 and M1 for comparison
Bit 0 (LSB) is cleared	Ignore D0 and M0 for comparison

8.2.7.7.3 Data comparison for selection command on flags

Each select command of the following commands have two arguments (parameter and data):

- GROUP_SELECT_EQ_FLAGS,
- GROUP_SELECT_NE_FLAGS,
- GROUP_UNSELECT_EQ_FLAGS, or
- GROUP_UNSELECT_NE_FLAGS.

The two arguments (parameter and data) are:

- BYTE_MASK, or
- BYTE_DATA.

The tag shall make two possible comparisons:

- EQ FLAGS EQUAL D, or
- NE FLAGS NOT EQUAL D.

The arguments of the comparison are FLAGS, as defined in clause 8.2.1.3 and the argument of the command D, consisting of the bits D7 (MSB) to D0 (LSB).

The argument BYTE_MASK defines what bits to be considered for comparison, as shown in Table 105.

Table 105 — Data masking for Group_Select_Flags and Group_Unselect_Flags

BYTE_MASK	BYTE_DATA
Bit 7 (MSB) is set	Consider D7 and FLAG7 for comparison
Bit 6 is set	Consider D6 and FLAG6 for comparison
Bit 5 is set	Consider D5 and FLAG5 for comparison
Bit 4 is set	Consider D4 and FLAG4 for comparison
Bit 3 is set	Consider D3 and FLAG3 for comparison
Bit 2 is set	Consider D2 and FLAG2 for comparison
Bit 1 is set	Consider D1 and FLAG1 for comparison
Bit 0 (LSB) is set	Consider D0 and FLAG0 for comparison
Bit 7 (MSB) is cleared	Ignore D7 and FLAG7 for comparison
Bit 6 is cleared	Ignore D6 and FLAG6 for comparison
Bit 5 is cleared	Ignore D5 and FLAG5 for comparison
Bit 4 is cleared	Ignore D4 and FLAG4 for comparison
Bit 3 is cleared	Ignore D3 and FLAG3 for comparison
Bit 2 is cleared	Ignore D2 and FLAG2 for comparison
Bit 1 is cleared	Ignore D1 and FLAG1 for comparison
Bit 0 (LSB) is cleared	Ignore D0 and FLAG0 for comparison

NOTE In the formulae below the following symbols are used, = …. is equal to, != … is not equal to, ! … Boolean not

Formula describing the EQUAL function:

The EQUAL comparison passes, if (!B7+(D7=FLAG7)) * (!B6+(D6=FLAG6)) * (!B5+(D5=FLAG5)) * (!B4+(D4=FLAG4)) * (!B3+(D3=FLAG3)) * (!B2+(D2=FLAG2)) * (!B1+(D1=FLAG1)) * (!B0+(D0=FLAG0)) is true.

Formula describing the UNEQUAL function:

The UNEQUAL comparison passes, if B7*(D7!=FLAG7) + B6*(D6!=FLAG6) + B5*(D5!=FLAG5) + B4*(D4!=FLAG4) + B3*(D3!=FLAG3) + B2*(D2!=FLAG2) + B1*(D1!=FLAG1) + B0*(D0!=FLAG0) is true.

8.2.7.7.4 GROUP_SELECT_EQ

Command code = ‘00’

On receiving a GROUP_SELECT_EQ command, as shown in Table 106, a tag that is in the READY state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is equal to WORD_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 107, and go into the state ID.

On receiving a GROUP_SELECT_EQ command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 106 — GROUP_SELECT_EQ command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 107 — GROUP_SELECT_EQ response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

NOTE If the byte mask is zero, GROUP_SELECT_EQ selects all tags.

8.2.7.7.5 GROUP_SELECT_NE

Command code = '01'

On receiving a GROUP_SELECT_NE command, as shown in Table 108, a tag that is in the READY state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is not equal to WORD_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 109, and go into the state ID.

On receiving a GROUP_SELECT_NE command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 108 — GROUP_SELECT_NE command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 109 — GROUP_SELECT_NE response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.6 GROUP_SELECT_GT

Command code = '02'

On receiving a GROUP_SELECT_GT command, as shown in Table 110, a tag that is in the READY state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is greater than WORD_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 111, and go into the state ID.

On receiving a GROUP_SELECT_GT command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 110 — GROUP_SELECT_GT command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 111 — GROUP_SELECT_GT response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.7 GROUP_SELECT_LT

Command code = '03'

On receiving a GROUP_SELECT_LT command, as shown in Table 112, a tag that is in the READY state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is lower than WORD_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 113, and go into the state ID.

On receiving a GROUP_SELECT_LT command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 112 — GROUP_SELECT_LT command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 113 — GROUP_SELECT_LT response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.8 GROUP_UNSELECT_EQ

Command code = '04'

On receiving a GROUP_UNSELECT_EQ command, as shown in Table 114, a tag that is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is equal to WORD_DATA the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 115.

In all other cases the tag shall not send a reply.

Table 114 — GROUP_UNSELECT_EQ command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 115 — GROUP_UNSELECT_EQ response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

NOTE If the byte mask is zero, GROUP_UNSELECT_EQ unselects all tags.

8.2.7.7.9 GROUP_UNSELECT_NE

Command code = ‘05’

On receiving a GROUP_UNSELECT_NE command, as shown in Table 116, a tag that is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is not equal to WORD_DATA the tag shall go into the state READY and not send any reply. In the case the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID as shown in Table 117.

In all other cases the tag shall not send a reply.

Table 116 — GROUP_UNSELECT_NE command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 117 — GROUP_UNSELECT_NE response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.10 GROUP_UNSELECT_GT

Command code = ‘06’

On receiving a GROUP_UNSELECT_GT command, as shown in Table 118, a tag that is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is greater than to WORD_DATA the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 119.

In all other cases the tag shall not send a reply.

Table 118 — GROUP_UNSELECT_GT command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 119 — GROUP_UNSELECT_GT response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.11 GROUP_UNSELECT_LT

Command code = '07'

On receiving a GROUP_UNSELECT_LT command, as shown in Table 120, a tag that is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the WORD_DATA sent by the interrogator. In the case that the memory content is lower than to WORD_DATA the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 121.

In all other cases the tag shall not send a reply.

Table 120 — GROUP_UNSELECT_LT command

Preamble	Delimiter	Command	Address	Mask	WORD_DATA	CRC-16
		8 bits	8 bits	8 bits	64 bits	16 bits

Table 121 — GROUP_UNSELECT_LT response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.12 MULTIPLE_UNSELECT

Command code = '13'

On receiving a MULTIPLE_UNSELECT command, as shown in Table 122, a tag that is in the ID state shall read the 1-byte memory content beginning at the specified address and compare it with the BYTE_DATA sent by the interrogator. In the case that the memory content is equal to BYTE_DATA and the flag WRITE_OK is set, then the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 123.

In all other cases the tag shall not send a reply.

Table 122 — MULTIPLE_UNSELECT command

Preamble	Delimiter	Command	ADDRESS	BYTE_DATA	CRC-16
		8 bits	8 bits	8 bits	16 bits

Table 123 — MULTIPLE_UNSELECT response in the case that WRITE_OK is reset or BYTE_DATA is not equal to memory content at ADDRESS

Preamble	ID	CRC-16
	64 bits	16 bits

This command may be used to unselect all tags that had a successful write, while tags that had a weak write or write problems stay selected.

8.2.7.7.13 GROUP_SELECT_EQ_FLAGS

Command code = '17'

On receiving a GROUP_SELECT_EQ_FLAGS command, as shown in Table 124, a tag that is in the READY state shall compare the FLAGS with the BYTE_DATA sent by the interrogator. In the case that the FLAGS are equal to BYTE_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 125, and go into the state ID.

On receiving a GROUP_SELECT_EQ_FLAGS command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 124 — GROUP_SELECT_EQ_FLAGS command

Preamble	Delimiter	Command	Mask	BYTE_DATA	CRC-16
		8 bits	8 bits	8 bits	16 bits

Table 125 — GROUP_SELECT_EQ_FLAGS response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

NOTE If the byte mask is zero, GROUP_SELECT_EQ_FLAGS selects all tags.

8.2.7.7.14 GROUP_SELECT_NE_FLAGS

Command code = '18'

On receiving a GROUP_SELECT_NE_FLAGS command, as shown in Table 126, a tag that is in the READY state shall compare the FLAGS with the BYTE_DATA sent by the interrogator. In the case that the FLAGS are not equal to BYTE_DATA the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 127, and go into the state ID.

On receiving a GROUP_SELECT_NE_FLAGS command, a tag that is in the ID state shall set its internal counter COUNT to 0, read its UID and send back the UID and stay in the ID state.

In all other cases the tag shall not send a reply.

Table 126 — GROUP_SELECT_NE_FLAGS command

Preamble	Delimiter	Command	Mask	BYTE_DATA	CRC-16
		8 bits	8 bits	8 bits	16 bits

Table 127 — GROUP_SELECT_NE_FLAGS response in the case of NO error

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.7.15 GROUP_UNSELECT_EQ_FLAGS

Command code = '19'

On receiving a GROUP_UNSELECT_EQ_FLAGS command, as shown in Table 128, a tag that is in the ID state shall compare the FLAGS with the BYTE_DATA sent by the interrogator. In the case that the FLAGS are equal to BYTE_DATA the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 129.

In all other cases the tag shall not send a reply.

Table 128 — GROUP_UNSELECT_EQ_FLAGS command

Preamble	Delimiter	Command	Mask	BYTE_DATA	CRC-16
		8 bits	8 bits	8 bits	16 bits

Table 129 — GROUP_UNSELECT_EQ_FLAGS response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

NOTE If the byte mask is zero, GROUP_UNSELECT_EQ_FLAGS unselects all tags.

8.2.7.7.16 GROUP_UNSELECT_NE_FLAGS

Command code = '1A'

On receiving a GROUP_UNSELECT_NE_FLAGS command, as shown in Table 130, a tag that is in the ID state shall compare the FLAGS with the BYTE_DATA sent by the interrogator. In the case that the FLAGS are not equal to BYTE_DATA the tag shall go into the state READY and not send any reply. In the case that the comparison fails, the tag shall set its internal counter COUNT to 0, read its UID and send back the UID, as shown in Table 131.

In all other cases the tag shall not send a reply.

Table 130 — GROUP_UNSELECT_NE_FLAGS command

Preamble	Delimiter	Command	Mask	BYTE_DATA	CRC-16
		8 bits	8 bits	8 bits	16 bits

Table 131 — GROUP_UNSELECT_NE_FLAGS response in the case of NO error and comparison fails

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.8 Identification commands

8.2.7.8.1.1 Identification commands general

Identification commands are used to perform the multiple tag identification protocol.

8.2.7.8.1.2 FAIL

Command code = ‘08’

The identification algorithm uses FAIL when more than one tag tried to identify itself at the same time. Some tags back off and some tags retransmit.

A tag shall only accept a FAIL command, as shown in Table 132, if it is in the ID state. In the case that its internal counter COUNT is not zero or the random generator result is 1, then COUNT shall be increased by 1, unless it is FF. If the count is at FF, then the count remains unchanged for further FAIL commands.

If the resulting COUNT value is 0, then the tag shall read its UID and sent back it in the response, as shown in Table 133.

Table 132 — FAIL command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 133 — FAIL response in the case that COUNT stays zero

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.8.1.3 SUCCESS

Command code = ‘09’

SUCCESS initiates identification of the next set of tags. It is used in two cases:

- When all tags receiving FAIL backed off and did not transmit, SUCCESS causes those same tags to transmit again, and
- After a DATA_READ moves an identified tag to DATA_EXCHANGE, SUCCESS causes the next subset of selected but unidentified tags to transmit.

A tag shall only accept a SUCCESS command, as shown in Table 134, if it is in the ID state. In the case that the tag's internal counter COUNT is not zero, the internal counter shall be decreased by 1.

If the resulting COUNT value is 0, then the tag shall read its UID and sent back it in the response, as shown in Table 135.

Table 134 — SUCCESS command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 135 — SUCCESS response in the case that COUNT is zero

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.8.1.4 RESEND

Command code = ‘15’

The identification algorithm uses RESEND when only one tag transmitted but the UID was received in error. The tag that transmitted resends its UID.

A tag shall only accept a RESEND command, as shown in Table 136, if it is in the ID state. If the COUNT value is 0, then the tag shall read its UID and sent back it in the response, as shown in Table 137.

Table 136 — RESEND command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 137 — RESEND response in the case that COUNT is zero

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.8.1.5 INITIALIZE

Command code = ‘0A’

On receiving a INITIALIZE command, as shown in Table 138 a tag shall go into the READY state and reset the Data_Exchange_Status_Bit.

It shall not send any response.

Table 138 — INITIALIZE command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

8.2.7.8.1.6 FAIL_O

Command code = ‘40’ or ’41’

The identification algorithm uses FAIL_O when more than one tag tried to identify itself at the same time. Some tags back off and some tags retransmit.

A tag shall only accept a FAIL_O command, as shown in Table 139, if it is in the ID state. In the case that the tag's internal counter COUNT is not zero, or the random generator result is 1, the internal counter shall be increased by 1, unless COUNT is FF.

If the resulting COUNT value is 0, then the tag shall read the memory at address ‘14’ sent back either 32 bit (for command code ‘40’) or 64 bit (for command code ‘41’) in the response, as shown in Table 140.

Table 139 — FAIL_O command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 140 — FAIL_O response in the case that COUNT stays zero

Preamble	WORD_DATA	CRC-16
	32 or 64 bits	16 bits

8.2.7.8.1.7 SUCCESS_O

Command code = ‘42’ or ‘43’

SUCCESS_O initiates identification of the next set of tags. It is used in two cases:

- When all tags receiving FAIL_O backed off and did not transmit, SUCCESS_O causes those same tags to transmit again, and
- After a DATA_READ_O moves an identified tag to DATA_EXCHANGE, SUCCESS_O causes the next subset of selected but unidentified tags to transmit.

A tag shall only accept a SUCCESS_O command, as shown in Table 141 if it is in the ID state. In the case that the tag's internal counter COUNT is not zero it shall be decreased by 1.

If the resulting COUNT value is 0, then the tag shall read the memory at address ‘14’ sent back either 32 bit (for command code ‘40’) or 64 bit (for command code ‘41’) in the response, as shown in Table 142.

Table 141 — SUCCESS_O command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 142 — SUCCESS_O response in the case that COUNT is zero

Preamble	WORD_DATA	CRC-16
	64 bits	16 bits

8.2.7.8.1.8 RESEND_O

Command code = ’46’ or ‘47’

The identification algorithm uses RESEND_O when only one tag transmitted but the ID was received in error. The tag that transmitted resends its ID.

A tag shall only accept a RESEND_O command, as shown in Table 143, if it is in the ID state. If the COUNT value is 0, then the tag shall read the memory at address ‘10’ sent back either 32 bit (for command code ‘46’) or 64 bit (for command code ‘47’) in the response, as shown in Table 144.

Table 143 — RESEND_O command

Preamble	Delimiter	Command	CRC-16
		8 bits	16 bits

Table 144 — RESEND_O response in the case that COUNT is zero

Preamble	ID	CRC-16
	64 bits	16 bits

8.2.7.9 Data transfer commands

8.2.7.9.1 Data transfer commands general

Data Transfer commands are used to read or write data from or to the memory.

8.2.7.9.2 READ

Command code = ‘0C’

On receiving the READ command, as shown in Table 145, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state move to the DATA_EXCHANGE state, read the 8 byte memory content beginning at the specified address and send back its content in the response, as shown in Table 146. Further, the tag shall mark the byte at ADDRESS lockable. In the case that ID is not equal to UID or any other error the tag shall not send a reply.

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 145 — Read command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 146 — Read response in the case of NO error

Preamble	WORD_DATA	CRC-16
	64 bits	16 bits

8.2.7.9.3 DATA_READ

Command code = ‘0B’

On receiving the DATA_READ command, as shown in Table 147, the tag shall only if it is in either the state ID or the state DATA_EXCHANGE compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state except READY move to the DATA_EXCHANGE state, read the 8 byte memory content beginning at the specified address and send back its content in the response, as shown in Table 148. Further, the tag shall mark the byte at ADDRESS lockable. In the case that the tag is not in the state READY, or ID is not equal to UID or any other error the tag shall not send a reply.

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 147 — DATA_READ command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 148 — DATA_READ response in the case of NO error

Preamble	WORD_DATA	CRC-16
	64 bits	16 bits

8.2.7.9.4 DATA_READ_O

Command code = ‘44’, or ‘45’

On receiving the DATA_READ_O command, as shown in Table 149, the tag shall compare the sent DATA with its memory contents starting at address ‘14’ if the tag is in either the ID state or the DATA_EXCHANGE state. The amount of data compared shall be determined by the command code. If the received command code is '44', the tag shall logically compare the 32 bits. If command code is '45', the tag shall logically compare 64 bits. If the data strings (32 or 64 bits) are equal, then the tag shall transition from any state except READY to the DATA_EXCHANGE state, read the 8 byte memory content beginning at the specified address and send back its contents in the response, as shown in Table 150. Further, the tag shall mark the byte at ADDRESS lockable. If the tag is in the READY state, or the data strings are not equal, or any other error the tag shall not send a reply.

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 149 — DATA_READ_O command

Preamble	Delimiter	Command	DATA	Address	CRC-16
		8 bits	32 or 64 bits	8 bits	16 bits

Table 150 — DATA_READ_O response in the case of NO error

Preamble	WORD_DATA	CRC-16
	64 bits	16 bits

8.2.7.9.5 READ_FLAGS

Command code = ‘50’

On receiving the READ_FLAGS command, as shown in Table 151, the tag shall from any state move to the DATA_EXCHANGE state, read the FLAGS and send back its content in the response, as shown in Table 152.

Table 151 — READ_FLAGS command

Preamble	Delimiter	Command	ID	CRC-16
		8 bits	64 bits	16 bits

Table 152 — READ_FLAGS in the case of NO error

Preamble	BYTE_DATA	CRC-16
	8 bits	16 bits

8.2.7.9.6 READ_VARIABLE

Command code = ‘51’

On receiving the READ_VARIABLE command, as shown in Table 153, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state move to the DATA_EXCHANGE state, read the specified length of the memory content beginning at the specified address and send back its content in the response, as shown in Table 154. In the case that ID is not equal to UID or any other error the tag shall not send a reply.

The address is numbered from ‘00’ to ‘FF’ (0 to 255). Length is numbered from ‘00’ to ‘FF’.

Table 153 — READ_VARIABLE command

Preamble	Delimiter	Command	ID	Address	Length	CRC-16
		8 bits	64 bits	8 bits	8 bits	16 bits

Table 154 — READ_VARIABLE response in the case of NO error

Preamble	Length * BYTE_DATA	CRC-16
	Length * 8 bits	16 bits

8.2.7.9.7 READ_PORT

Command code = '52'

On receiving the READ_PORT command, as shown in Table 155, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state move to the DATA_EXCHANGE, read the 8 bit memory content beginning at the specified port address and send back the memory content in the response, as shown in Table 156. In the case that ID is not equal to UID or any other error the tag shall not send a reply.

The address is numbered from '00' to 'FF' (0 to 255). Length is numbered from '00' to 'FF'

Table 155 — READ_PORT command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 156 — READ_PORT response in the case of NO error

Preamble	BYTE_DATA	CRC-16
	8 bits	16 bits

Selecting port 0 shall select the flags as specified in clause 8.2.1.3, and Table 95

Ports 1, 2, … to 'FF' are reserved for future use.

8.2.7.9.8 READ_UNADDRESSED

Command code = '53'

On receiving the READ_UNADDRESSED command, as shown in Table 157, the tag shall from any state move to the DATA_EXCHANGE state, read the 16-byte memory content beginning at the specified address and send back its content in the response, as shown in Table 158.

The address is numbered from '00' to 'FF' (0 to 255).

Table 157 — READ_UNADDRESSED command

Preamble	Delimiter	Command	Address	CRC-16
		8 bits	8 bits	16 bits

Table 158 — READ_UNADDRESSED response in the case of NO error

Preamble	WORD_DATA	WORD_DATA	CRC-16
	64 bits	64 bits	16 bits

8.2.7.9.9 READ_VERIFY

Command code = '12'

On receiving the READ_VERIFY command, as shown in Table 159 the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID and the WRITE_OK flag is set, the tag shall from any state move to the DATA_EXCHANGE state, read the 1-byte memory content beginning at the specified address and send back its content in the response, as shown in Table 160. Further, the tag shall mark the byte at ADDRESS lockable. In the case that ID is not equal to UID, WRITE_OK is not set, or any other error the tag shall not send a reply.

The address is numbered from '00' to 'FF' (0 to 255).

Table 159 — READ_VERIFY command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 160 — READ_VERIFY response in the case of NO error

Preamble	BYTE_DATA	CRC-16
	8 bits	16 bits

8.2.7.9.10 READ_VERIFY_4BYTE

Command code = '1D'

On receiving the READ_VERIFY_4BYTE command, as shown in Table 161, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID and the WRITE_OK flag is set, the tag shall from any state move to the DATA_EXCHANGE state, read the 4-byte memory content beginning at the specified address and send back its content in the response, as shown in Table 162. Further, the tag shall mark the byte at ADDRESS lockable. Bytes at ADDRESS+1, ADDRESS+2 and ADDRESS+3 shall not be marked lockable.

In the case that ID is not equal to UID, WRITE_OK is not set, or any other error the tag shall not send a reply.

BYTE_MASK of the command

ADDRESS bit of BYTE_MASK to select whether byte should be written

[ADDR+0] B7

[ADDR+1] B6

[ADDR+2] B5

[ADDR+3] B4

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 161 — READ_VERIFY_4BYTE command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 162 — READ_VERIFY_4BYTE response in the case of NO error

Preamble	BYTE_DATA	CRC-16
	8 bits	16 bits

8.2.7.9.11 WRITE

Command code = ‘0D’

On receiving the WRITE command, as shown in Table 163, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state move to the DATA_EXCHANGE state, read the lock information for the byte on the specified memory content beginning at the specified address. In the case that the memory is locked, it shall send back the ERROR response, as shown in Table 165. Further, the tag shall mark the byte at ADDRESS lockable. In the case that the memory is unlocked, it shall send back the ACKNOWLEDGE, as shown in Table 164, and program the data into the specified memory address. In all other cases the tag shall not send a reply.

In the case that the write access was successful, the tag shall set the WRITE_OK bit. Otherwise it shall reset it.

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 163 — Write command

Preamble	Delimiter	Command	ID	Address	BYTE_ DATA	CRC-16
		8 bits	64 bits	8 bits	8 bits	16 bits

Table 164 — WRITE response in the case of NO error

Preamble	ACKNOWLEDGE	CRC-16
	8 bits	16 bits

Table 165 — WRITE response in the case of locked memory

Preamble	ERROR	CRC-16
	8 bits	16 bits

8.2.7.9.12 WRITE4BYTE

Command code = ‘1B’

On receiving the WRITE4BYTE command, as shown in Table 166, the tag shall compare the sent ID with its UID. In the case that the ID is equal to the UID, the tag shall from any state move to the DATA_EXCHANGE state, read the lock information for the 4 bytes on the specified memory content beginning at the specified address. In the case that one of the memory bytes is locked, it shall send back the ERROR response, as shown in Table 168. In the case that all bytes are unlocked, it shall send back the ACKNOWLEDGE, as shown in Table 167, and program the data into the specified memory. In all other cases the tag shall not send a reply.

Executing WRITE4BYTE a tags shall only write those bytes that are selected by the BYTE_MASK, which means that write could be done to 1 to 4 bytes(using the mask bits in the BYTE_MASK field).

In the case that the write access was successful, the tag shall set the WRITE_OK bit. Otherwise it shall reset it.

BYTE_MASK of the command

ADDRESS bit of BYTE_MASK to select whether byte should be written

[ADDR+0] B7

[ADDR+1] B6

[ADDR+2] B5

[ADDR+3] B4

The address is numbered from ‘00’ to ‘FF’ (0 to 255). The starting address for the WRITE4BYTE command must be on a 4-byte page boundary.

Table 166 — WRITE4BYTE command

Preamble	Delimiter	Command	ID	Address	Byte_Mask	Data	CRC-16
		8 bits	64 bits	8 bits	8 bits	32 bits	16 bits

Table 167 — WRITE4BYTE response in the case of NO error

Preamble	ACKNOWLEDGE	CRC-16
	8 bits	16 bits

Table 168 — WRITE4BYTE response in the case that of locked memory

Preamble	ERROR	CRC-16
	8 bits	16 bits

8.2.7.9.13 LOCK

Command code = ‘0F’

On receiving a LOCK command, as shown in Table 169, a tag that is in the DATA_EXCHANGE state shall read its UID and compare it with the ID sent by the interrogator. In the case that the UID is equal to ID, the ADDRESS is within the valid address range and the byte at ADDRESS is marked lockable, then the tag shall send back the ACKNOWLEDGE, as shown in Table 170 and program the lock bit of the specified memory address. In case the ADDRESS is not in the valid range, or it is not marked as lockable, then the tag shall send back the ACKNOWLEDGE_NOK, as shown in Table 171.

In all other cases the tag shall not send a reply.

In the case that the write access was successful, the tag shall set the WRITE_OK bit. Otherwise it shall reset it.

The address is numbered from '00' to 'FF' (0 to 255).

Table 169 — LOCK command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 170 — LOCK response in the case that locking was possible

Preamble	ACKNOWLEDGE	CRC-16
	8 bits	16 bits

Table 171— LOCK response in the case that locking was not possible

Preamble	ERROR	CRC-16
	8 bits	16 bits

8.2.7.9.14 QUERY_LOCK

Command code = '11'

On receiving a QUERY_LOCK command, as shown in Table 172, a tag shall read its UID and compare it with the ID sent by the interrogator. In the case that the UID is equal to ID and the ADDRESS is within the valid address range the tag shall move into the DATA_EXCHANGE state. Further, the tag shall read the lock bit for the memory byte at ADDRESS. Further, the tag shall mark the byte at ADDRESS lockable. In the case that this memory is not locked, then it shall either respond with ACKNOWLEDGE_OK, as shown in Table 173, if WRITE_OK is set, or ACKNOWLEDGE_NOK, as shown in Table 174 if WRITE_OK is cleared. In the case that this memory is locked, then it shall either respond with ERROR_OK, as shown in Table 175 if WRITE_OK is set, or ERROR_NOK, as shown in Table 176, if WRITE_OK is cleared.

In all other cases the tag shall not send a reply.

The address is numbered from '00' to 'FF' (0 to 255).

Table 172 — QUERY_LOCK command

Preamble	Delimiter	Command	ID	Address	CRC-16
		8 bits	64 bits	8 bits	16 bits

Table 173 — QUERY_LOCK response in memory address is not locked and WRITE_OK is set

Preamble	ACKNOWLEDGE_OK	CRC-16
	8 bits	16 bits

Table 174 — QUERY_LOCK response in memory address is not locked and WRITE_OK is cleared

Preamble	ACKNOWLEDGE_NOK	CRC-16
	8 bits	16 bits

Table 175 — QUERY_LOCK response in memory address is locked and WRITE_OK is set

Preamble	ERROR_OK	CRC-16
	8 bits	16 bits

Table 176 — QUERY_LOCK response in memory address is locked and WRITE_OK is cleared

Preamble	ERROR_NOK	CRC-16
	8 bits	16 bits

8.2.7.9.15 WRITE_MULTIPLE

Command code = ‘0E’

Write Multiple commands, as shown in Table 177, are used to write to and to verify multiple tags in parallel.

On receiving the WRITE_MULTIPLE command, a tag that is in the ID state or the DATA_EXCHANGE state shall read the lock information for the byte on the specified memory content beginning at the specified address. In the case that the memory is locked, it shall do nothing. In the case that unlocked, it shall program the data into the specified memory.

In the case that the write access was successful, the tag shall set the WRITE_OK bit. Otherwise it shall reset it.

Further, the tag shall mark the byte at ADDRESS lockable.

The address is numbered from ‘00’ to ‘FF’ (0 to 255).

Table 177 — WRITE_MULTIPLE command

Preamble	Delimiter	Command	Address	Data	CRC-16
		8 bits	8 bits	8 bits	16 bits

8.2.7.9.16 WRITE4BYTE_MULTIPLE

Command code = '1C'

Write Multiple commands, as shown in Table 178, are used to write to and to verify multiple tags in parallel.

On receiving the WRITE4BYTE_MULTIPLE command, a tag that is in the ID state or the DATA_EXCHANGE state shall read the lock information for the 4 bytes on the specified memory content beginning at the specified address. In the case that one of the 4-byte block is locked, it shall do nothing. In the case that all bytes are unlocked, it shall program the data into the specified memory.

Executing WRITE4BYTE a tags shall only write those bytes that are selected by the BYTE_MASK, which means that write could be done to 1 to 4 bytes (using the mask bits in the byte_mask field).

In the case that the write access was successful, the tag shall set the WRITE_OK bit. Otherwise it shall reset it.

BYTE_MASK of the command

ADDRESS bit of BYTE_MASK to select whether byte should be written

[ADDR+0] B7

[ADDR+1] B6

[ADDR+2] B5

[ADDR+3] B4

The address is numbered from '00' to 'FF' (0 to 255). The starting address for the WRITE4BYTE command must be on a 4-byte page boundary.

Table 178 — WRITE4BYTE_MULTIPLE command

Preamble	Delimiter	Command	Address	Byte_Mask	Data	CRC-16
		8 bits	8 bits	8 bits	32 bits	16 bits

8.2.7.10 Response description (binary tree protocol type)

8.2.7.10.1 ACKNOWLEDGE

ACKNOWLEDGE indicates a successful acceptance of the WRITE or LOCK.

8.2.7.10.2 ERROR

ERROR indicates an error in the WRITE. e.g. write to locked memory area

8.2.7.10.3 ACKNOWLEDGE_OK

ACKNOWLEDGE_OK is the response to a QUERY_LOCK and indicates an unlocked memory byte and a successful proceeding write command.

8.2.7.10.4 ACKNOWLEDGE_NOK

ACKNOWLEDGE_NOK is the response to a QUERY_LOCK and indicates an unlocked memory byte and an unsuccessful proceeding write command.

8.2.7.10.5 ERROR_OK

ERROR_OK is the response to a QUERY_LOCK and indicates a locked memory byte and a successful proceeding write command.

8.2.7.10.6 ERROR_NOK

ERROR_NOK is the response to a QUERY_LOCK and indicates as locked memory byte and an unsuccessful proceeding write command.

8.2.7.10.7 WORD_DATA

WORD_DATA is 8 bytes returned in response to a READ, DATA_READ, or DATA_READ_O command.

8.2.7.10.8 BYTE_DATA

BYTE_DATA is 1 byte returned in response to the READ_VERIFY and READ_PORT command.

8.2.7.10.9 4BYTEDATA

4BYTEDATA are 4 bytes used as argument for commands WRITE4BYTE, WRITE4BYTE_MULTIPLE and READ_VERIFY4BYTE.

8.2.8 Transmission errors

There are two types of transmission errors: modulation coding errors (detectable per bit) and CRC errors (detectable per command). Both errors cause any command to be aborted. The tag does not respond.

For all CRC errors, the tag returns to the ready state.

For all coding errors, the tag returns to the READY state if a valid start delimiter had been detected. Otherwise it maintains its current state.

Annex A
(informative)

Cyclic redundancy check (CRC)

A.1 Interrogator to tag CRC-5

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

A possible implementation is using a 5-bit shift register as defined in Figure 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

Table A.1 — 5 bit CRC register pre-load values

Register	Pre-load value
Q0	1
Q1	0
Q2	0
Q3	1
Q4	0

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.

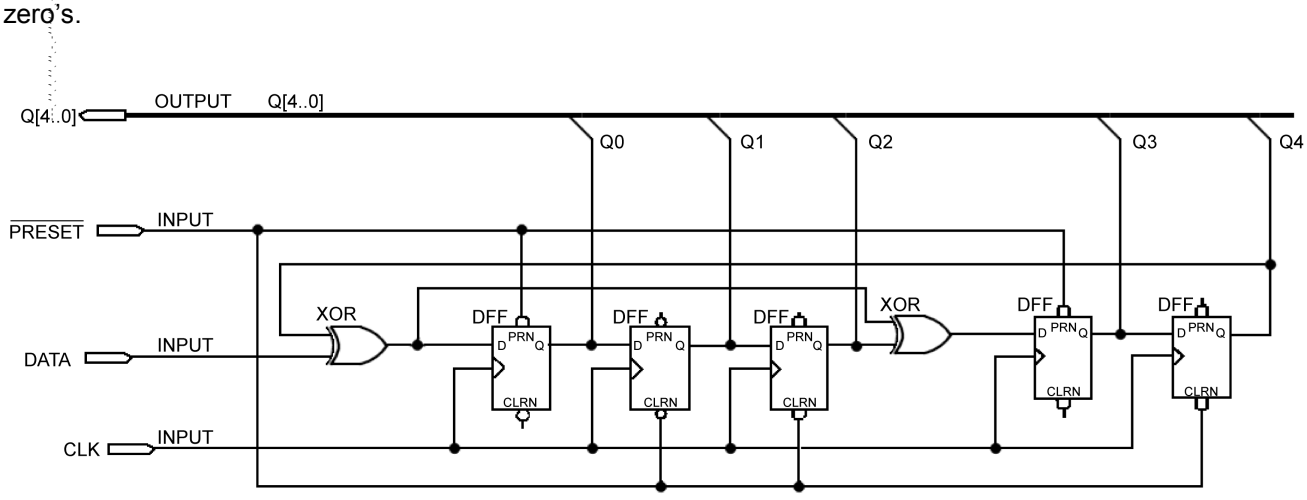


Figure A.1 — CRC-5

A.2 Interrogator to tag and tag to interrogator CRC-16

A.2.1 CRC-16 general

The polynomial used to calculate the CRC-16 is $X^{16}+X^{12}+X^5+1$, which is the CRC-CCITT International Standard. The Cyclic Redundancy Check (CRC) is calculated on all data contained in a message, from the start of the command through to the end of data. This CRC-16 is used from interrogator to tag and from tag to interrogator. The CRC-16 is defined in Table A.2

Table A.2 — CRC-16 definition

CRC definition					
CRC type	**Length**	**Polynomial**	**Direction**	**Preset**	**Residue**
CRC-CCITT	16 bits	$X^{16} + X^{12} + X^5 + 1$	Forward	0xFFFF(HEX)	'0'

The CRC algorithm is as follows:

For computing the CRC: (see example Table A.4)

- initialize the CRC accumulator to all ones – 0xFFFF (HEX),
- accumulate, MSB first, data using the polynomial $X^{16} + X^{12} + X^5 + 1$,
- invert the resulting CRC value, and
- attach the inverted CRC-16 to the end of the packet and transmit it MSB first.

For checking the CRC: (See example Table A.5)

- compute the CRC on the incoming packet,
- invert the received CRC data bits,
- accumulate the “inverted CRC bits” in the CRC registers, and
- verify that the accumulator is all zeroes.

An alternative can be used to check the CRC bits, which does not involve inverting the received CRC bits. The resulting CRC value will be 0x1D0F(HEX), as defined in CRC-CCITT. (See example Table A.6):

- compute the CRC on the incoming packet,
- accumulate the 16 “received CRC” in the CRC registers, and
- verify that the accumulator contains the value 1D0F.

Example A.1 — Example C code to generate the CRC bits for the Type B success command.

```
unsigned int Calc_CRC (unsigned int CRCacc, unsigned int cword)
{
/* Routine to calculate CRC for 1 byte (lower 8 bits of cword) */
/* Initially, CRCacc should have been set to 0xffff */
int i;
unsigned int xorval;
printf("\n");
for (i=0; i<8; i++)
 {
 xorval = ((CRCacc>>8) ^ (cword << i)) & 0x0080;
 CRCacc = (CRCacc << 1) & 0xfffe;
 if (xorval)
 CRCacc ^= 0x1021;
 printf("%04x\n",CRCacc);
 }
return (CRCacc);
}
main()
{
unsigned int CRCacc = 0xffff;
int i;
unsigned char test_str[2];
test_str[0] = 0x09; /* Success Command */
test_str[1] = '\0';
for (i =0; i < strlen(test_str); i++)
 CRCacc = Calc_CRC(CRCacc, test_str[i]);
printf("\nCRC = %04x\n",CRCacc);
}
```

A possible implementation is using a 16-bit shift register as defined in Figure A.2.

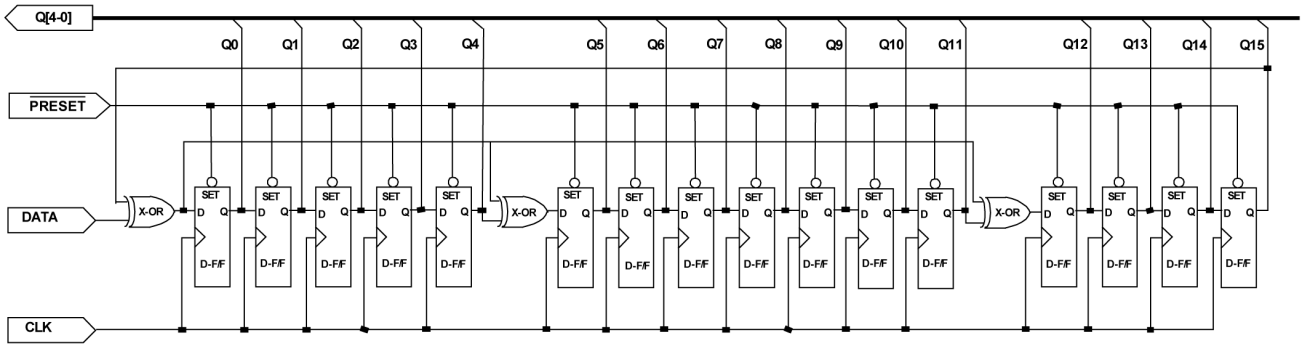


Figure A.2 — Shift register implementation of CRC-16 accumulator

A.2.2 CRC calculation examples

This example refers to a Type B interrogator sending a SUCCESS command.

Command code. '09'

The packet, as shown in Table A.3, sent from the interrogator to the tag consists of the following blocks, but only the SUCCESS command (09h), shown shaded, is used in the CRC calculation.

Table A.3 — CRC example Type B success command

Preamble detect	Preamble	Start delimiter	SUCCESS command	CRC-16
2 bytes field high	nine Manchester 0's	11 00 11 10 10	'09'	CRC-16

The CRC is calculated on the SUCCESS command as the field is transmitted MSB.

The following example, as shown in Table A.4, shows the values of the 16 CRC registers as the data is shifted through the CRC registers.

SUCCESS command 0 0 0 0 1 0 0 1 ('09')

Table A.4 — Practical example of CRC calculation for a 'SUCCESS' command in the interrogator

Step	Input (SUCCESS command)	Calculated CRC in interrogator (HEX)
1	0	'EFDF'
2	0	'CF9F'
3	0	'8F1F'
4	0	'0E1F'
5	1	'0C1F'
6	0	'183E'
7	0	'307C'
8	1	'70D9'

The CRC bits transmitted by the interrogator are inverted, 0x8F26 (HEX), MSB first.

The tag then inverts the received data bits and accumulates those 16 bits (i.e. 0x70D9 (HEX), as shown in Table A.5.

Table A.5 — Practical example of CRC checking for a 'SUCCESS' command in the tag

Step	Input (sent CRC-16)	Calculated CRC in tag (HEX)
0		'70D9'
1	0	'E1B2'
2	1	'C364'
3	1	'86C8'
4	1	'0D90'
5	0	'1B20'
6	0	'3640'
7	0	'6C80'
8	0	'D900'
9	1	'B200'
10	1	'6400'
11	0	'C800'
12	1	'9000'
13	1	'2000'
14	0	'4000'
15	0	'8000'
16	1	'0000'

Alternatively, the tag does not invert the received data bits and accumulates those 16 bits (i.e. 0x70D9 (HEX), as shown in Table A.6

Table A.6 — Practical example of CRC checking for a 'SUCCESS' command in the tag, without inversion of the received CRC bits, i.e. the received CRC bits 0x8F26 (HEX) are used.

Step	Input (sent CRC-16)	Calculated CRC in tag (HEX)
0		'70D9'
1	1	'F193'
2	0	'F307'
3	0	'F62F'
4	0	'FC7F'
5	1	'F8FE'
6	1	'F1FC'
7	1	'E3F8'
8	1	'C7F0'
9	0	'9FC1'
10	0	'2FA3'
11	1	'4F67'
12	0	'9ECE'
13	0	'2DBD'
14	1	'4B5B'
15	1	'8697'
16	0	'1D0F'

Annex B
(normative)

Memory mapping for Type B

B.1 Unique identifier (normative)

B.1.1 Unique identifier general

The tag serial number shall either be in accordance to clause B.1.2 and Table B.1, or clause B.1.3.

Differentiation is done by the leading bits in byte 0, which are 111 for unique identifiers as defined in chapter B.1.2 and 000 for unique identifier as defined in chapter B.1.3.

B.1.2 Unique identifier format

B.1.2.1 Unique identifier format general

Table B.1 — UID format

MSB							**LSB**
Byte 0 M L	**Byte 1** M L	**Byte 2** M L	**Byte 3** M L	**Byte 4** M L	**Byte 5** M L	**Byte 6** M L	**Byte 7** M L
'E0' 8 bits	IC Mfg code acc. ISO/IEC 7816-6	Chip Manufacturer Assigned 48 bits					

B.1.2.2 'E0' (byte 0)

E0 is the header for unique identifier followed by the manufacturer code according ISO/IEC 7816-6.

B.1.2.3 IC Mfg code according ISO/IEC 7816-6 (byte 1)

ISO/IEC 7816-6 defines an 8 bit code for chip manufacturers.

B.1.2.4 Chip manufacturer assigned (bytes 2 – 7)

This is a 48-bit field that is defined and managed by the chip manufacturer. Different chip manufacturers will have different manufacturer codes (see below), thus eliminating the potential for duplicated collision arbitration data (Tag UIDs). The numbering system employed by the chip manufacturer must ensure that all tags produced will have a unique and unambiguous number (used by the collision arbitration algorithm). This unique number will be "locked" prior to use. Maximum value for this field is 2^{48} –1.

Responsibility for ensuring uniqueness and for locking this unique number prior to use shall rest with the chip manufacturer.

B.1.3 Unique identifier according to ANSI 256

Table B.2 depicts bytes 0 to 7 of the tag serial number.

Table B.2 — Layout of tag serial number (bytes 0-7)

MSB							**LSB**
Byte 0 **MSB LSB**	**Byte 1** **MSB LSB**	**Byte 2** **MSB LSB**	**Byte 3** **MSB LSB**	**Byte 4** **MSB LSB**	**Byte 5** **MSB LSB**	**Byte 6** **MSB LSB**	**Byte 7** **MSB LSB**

000 zero 3bits	Chip Manufacturer Assigned 47 bits **MSB** **LSB**	Manufacturer ID 8 bits see NOTE 1 **MSB … LSB**	FAB 4 bits see NOTE 2 **MSB ... LSB**	CK 2 bits see NOTE 3 **MSB … LSB**

NOTE 1 Manufacturer ID field is the same as the IC Manufacturer.

NOTE 2 FAB is allocated by the IC manufacturer, such that when combined with the IC manufacturer code and serial number the resulting number is unique.

NOTE 3 CK represents the truncated sum of the bits set to '1' of the 62 bits preceding the checksum, valid values are 0, 1,2 and 3.

The Tag Serial Number shall be programmed and locked at the factory with a unique number for each tag.

B.1.4 Remaining system memory

B.1.4.1 Manufacturer ID (Bytes 8,9)

These two bytes, as shown in Table B.3, have been reserved for encoding the Tag Manufacturer ID to provide some conformance to anticipated RFID International Standards. These fields will initially be encoded with the following codes depending on the manufacturer of the tags.

Table B.3 — Manufacturer codes

Manufacturer	ASCII representation	Hexadecimal code
Reserved	'AT'	4154
Reserved	'HT'	4854
Reserved	'AA'	4141
Reserved	'AS'	4153
Reserved	'AN'	414E

B.1.4.2 Tag hardware type (bytes 10,11)

This is a 2-byte hexadecimal representation of the tag hardware design. This number shall be different for each type of hardware change made to the tag design that affects the function of the tag. This does not include differences in tag packaging, or colour, nor does it include differences in RF operational frequency. This field will be used to distinguish differences in commands or command structure, block size, and data capacity. It will also be used to distinguish differences in data protocol or optional features such as audio, or visual indicators.

B.1.4.3 Embedded application code (Byte 12)

B.1.4.3.1 Embedded application code general

This is the top-level hierarchy of tag memory layout. This field, in conjunction with the Tag Memory Allocation allows an application to determine the format and content of the user data. This field can be used to represent various formats of the tag data content.

Current hexadecimal assignments of the Embedded Application Code are shown in Table B.4:

Table B.4 — Embedded application codes

Embedded application code	Description of embedded application codes
00, FF	Unformatted, programmed to “FF” at manufacturer.
01	Reserved
02	Customer Specific Memory Allocation
03	File Allocation Table (Long Directory) – TBD in future
04	Check Tag
05	RFID Interrogator Configuration Tag
06	Reserved for future use
07	Reserved for Engineering Development
08	Reserved for future use
09	Reserved for future use
0A	ISO 15962 Compliant Data Format
0B	ANSI MH10.8.4 Compliant Data Format
0C – 0E	Reserved for future used
0F	UCC.EAN GTAG Compliant Data Format (see NOTE)
0C – FE	To be allocated and registered by application based needs.

NOTE GTAG compliant data format is selected by the command WAKEUPGTAG:

GROUP_SELECT_EQ (ADDRESS = ‘12’, BYTE_MASK = ‘01’, WORD_DATA = ’0F 00 00 00 00 00 00 00’

(0x00 0x12 0x01 0x0F 0x00 0x00 0x00 0x00 0x00 0x00 0x00)

B.1.4.3.2 Embedded application code '01' - reserved

All other Tag Memory Allocation Map combinations for Embedded Applications Code “01” are currently considered undefined. These remaining characters may be used to further specify functions and/or applications that have been appended to the primary application.

B.1.4.3.3 Embedded application code '02' - customer specific memory allocation

Customer specific data and file allocation tables are not detailed in this specification. However, it is envisioned that customers must register a Customer Specific Memory Allocation Code (CSMAC) with the manufacturer’s marketing and obtain a configuration string to write their two byte CSMAC value as well as an embedded application code of “02” in tag memory location byte 12.

Table B.5 provides examples of specific tag memory allocation fields for Customer Specific Tag Memory applications already being used by customers. It should be noted that this table is not inclusive of all customer specific memory allocations, nor does it represent all the codes registered. It is recommended that the manufacturer be contacted for obtaining a current listing and registering any additional codes.

Table B.5 — Customer Specific memory allocation codes bytes 13 & 14

Hexadecimal code	ASCII representation
4141	'AA'
414E	'AN'
4143	'AS'
4154	'AT'
4650	'GP'
4854	'HT'
4D44	'MD'
5046	'PF'
5354	'ST'

Other Customer Specific Tag Memory applications that may be added in the future include (but are not limited to):

- Theft Detection and Deterrence, or
- Emergency Warning Systems.

B.1.4.3.4 Embedded application code '03' - file allocation table (long directory)

The Tag Memory Allocation for Embedded Applications Code '03', File Allocation Table, has not yet been defined, and is only a placeholder at current. This format will allow a user to view the tag memory similar to that of a floppy drive on a computer.

B.1.4.3.5 Embedded application code '04' - check tag

The Check Tag architecture is made available so the interrogator may selectively read the check tag to verify the type of antenna attached and determines duty cycle and/or output power. This check tag function may also be used for providing end-to-end system diagnostic operational verification. These bytes are programmed and locked at the factory.

B.1.4.3.6 Embedded application code '05' - RFID interrogator configuration tag

The Tag Memory Allocation for Embedded Applications Code '05', RFID Interrogator Configuration Tag, has been reserved to indicate a tag used for configuring a interrogator by means of a special diagnostic configuration mode. This tag can be used for setting various configuration parameters in a interrogator or group of interrogators simply by reading the tag in the configuration mode. Format of this data is to be defined by the interrogator specification or may be added in a future version of this document as an appendix.

B.1.4.3.7 Embedded application codes '06' through '09'

Reserved for future use.

B.1.4.3.8 Embedded applications code '0A' – ISO/IEC 15961 and ISO/IEC 15962 compliant data format

The Tag Memory Allocation for Embedded Applications Code '0A', ISO/IEC 15961 and ISO/IEC 15962 Compliant Data Format, has been reserved to indicate a tag used in a fashion as defined by this part of ISO/IEC 18000. Under the current ISO/IEC 15961 and ISO/IEC 15962 definition byte 13 (Tag Memory Allocation Map) is allocated to retain the Application Family Identifier (AFI) information. Byte 14 is allocated to retain the Data Storage Format (DSF).

B.1.4.3.9 Embedded application codes '0B' – ANSI MH10.8.4 compliant data format

The Tag Memory Allocation for Embedded Applications Code '0B', ANSI MH10.8.4 Compliant Data Format, has been reserved to indicate a tag used in a fashion as defined by that application standard.

B.1.4.3.10 Embedded application codes '0C through 0E'

Reserved for future use.

B.1.4.3.11 Embedded application codes '0F' – EAN.UCC GTAG compliant data format

The Tag Memory Allocation for Embedded Applications Code '0F', EAN.UCC GTAG Compliant Data Format, has been reserved to indicate a tag used in a fashion as defined by that application standard. Under this definition byte 13 (Tag Memory Allocation Map) is allocated to retain the Application Family Identifier (AFI) information

B.1.4.3.12 Embedded application codes '10' through 'FF'

Reserved for future use.

B.1.4.3.13 Tag memory allocation map (Bytes 13 through 17)

This field is based on a structured hierarchy that allows for an application to determine the format and content of the user data in conjunction with the Embedded Application Code in byte 12 defined above.

B.1.4.3.14 Tag application memory (bytes 18 and above)

These bytes represent the general application data storage area in tag memory.

Annex C
(informative)

Tag Memory Map for Type B

C.1 Tag memory map

Table C.1 describes the overall memory map of the tags.

Table C.1 — Layout of tag memory map

Bytes	Field name	Written	Locked
0-7	Tag ID	Manufacturing	Manufacturing
8,9	Tag Manufacturer	Manufacturing	Manufacturing
10,11	Tag Hardware Type	Manufacturing	Manufacturing
12-17	Tag Memory Layout	Manufacturing or Application	As Required by Application
18 and above	User Data*	Application	As Required

* Definition and Format of User Data determined by Tag Memory Layout

The first eight bytes of the tag memory shall be programmed with unique Tag ID numbers. This field is hard-coded in the ASIC to allow tag sort algorithms to function properly, therefore it is important that these Tag Serial Numbers be unique.

Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

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

[2] ISO/IEC TR 18047-6, *Information technology — Radio frequency identification device conformance test methods — Part 6: Test methods for air interface communications at 860 MHz to 930 MHz*

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

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

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

ICS 35.040

Price based on 134 pages