ISO/IEC JTC 1/SC 31/WG 4/SG 3 N311

Date: 2002-05-04

ISO/IEC CD 18000-6

ISO/IEC JTC 1/SC 31/WG 4/SG 3

Secretariat: ANSI

Information Technology — Radio Frequency Identification (RFID) for Item Management — Part 6: Parameters for Air Interface Communications at 860-930 MHz

Technologie de l'information — Identification par radio-fréquence (RFID) pour la gestion d'objets — Partie 6: Paramètres de communication à 860-930MHz

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (30) Committee
Document language: E

D:\RFID 资 料 文 档 \ 标 准 法 规 \ISO_standard_18000\577_18000_6_CD_200205\N0577_18000_6_CD_200205.doc STD Version 2.0

Contents

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 3.

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

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 18000 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 18000-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 31.

ISO/IEC 18000 consists of the following parts, under the general title *Information Technology — Radio Frequency Identification (RFID) for Item Management*:

— *Air Interface, Part 1 – Generic Parameters for Air Interface Communication for Globally Accepted Frequencies*

— *Air Interface, Part 2 – Parameters for Air Interface Communications below 135 kHz*

— *Air Interface, Part 3 – Parameters for Air Interface Communications at 13.56 MHz*

— *Air Interface, Part 4 – Parameters for Air Interface Communications at 2.45 GHz*

— *Air Interface, Part 5 – Parameters for Air Interface Communications at 5.8 GHz*

— *Air Interface, Part 6 – Parameters for Air Interface Communications at 860-930 MHz*

Introduction

This standard describes a passive backscatter RFID system that supports the following system capabilities:

- System protocol
- 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

In this RFID system, the interrogator powers and communicates with the tags that are within their range. Tags receive data as on-off key amplitude modulation of the power/data signal from the reader. 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.

Information Technology — Radio Frequency Identification (RFID) for Item Management — Part 6: Parameters for Air Interface Communications at 860-930 MHz

1 Scope

This standard describes:

- the physical interactions between the interrogator and the tag
- the protocols and the commands,
- the collision arbitration schemes

2 Normative references

ISO/IEC 18000-1 - Air Interface, Part 1 – Generic Parameters for Air Interface Communication for Globally Accepted Frequencies

3 Definitions, abbreviations and symbols

3.1 Definitions

3.1.1 Collision arbitration loop

Algorithm used to prepare for and handle a dialogue between an interrogator and a tag. This also known as collision arbitration.

3.1.2 Byte

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

3.2 Abbreviations

AFI	Application family identifier
CRC	Cyclic redundancy check
DSFID	Data storage format identifier
EOF	End of frame
LSB	Least significant bit
MSB	Most significant bit
PIE	Pulse interval encoding

RFU Reserved for future use

SOF Start of frame

SUID Sub unique identifier

TEL Tag excitation level

UID Unique identifier

3.3 Symbols

f_c Frequency of operating field (carrier frequency)

Tcr Carrier rise time

Tcs Carrier steady time

Tcf Carrier fall time

Tfhr Carrier FHSS rise time

Tfhs Carrier FHSS steady time

Tfhf Carrier FHSS fall time

Trlb Return link bit time

Tapw Type A pulse width

Tapf Type A pulse fall time

Tapr Type A pulse rise time

Tari Type A reference interval time

Taq Type A quiet time

Tbmr Type B Manchester rise time

Tbmf Type B Manchester fall time

Cht Carrier high level tolerance

Clt Carrier low level tolerance

Mut Modulation upper tolerance

Mlt Modulation lower tolerance

4 Overview

This standard specifies two types, Type A and Type B.

For the forward link, Type A uses Pulse interval encoding, Type B uses bi-phase modulation and Manchester encoding.

For the collision arbitration, Type A uses an Aloha-based mechanism, Type B uses an adaptive binary tree mechanism.

Both types uses the same bi-phase space FM0 return link encoding. See the parameter tables in clauses 4.1 and 6.4.

Figure 1, Figure 2 and Figure 3 show their respective architecture.

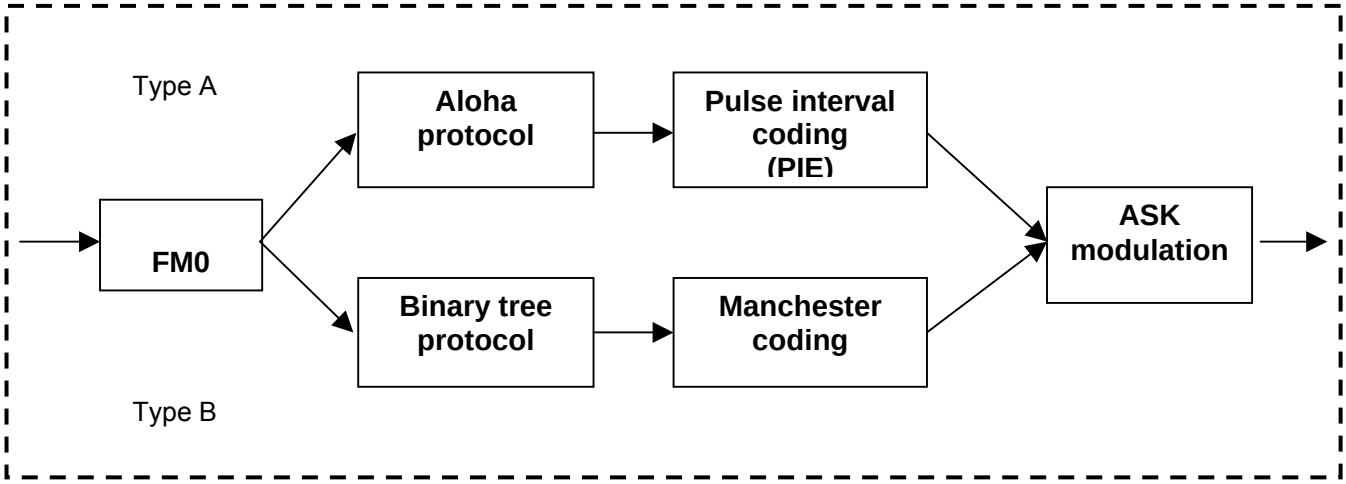


Figure 1 — Interrogator architecture

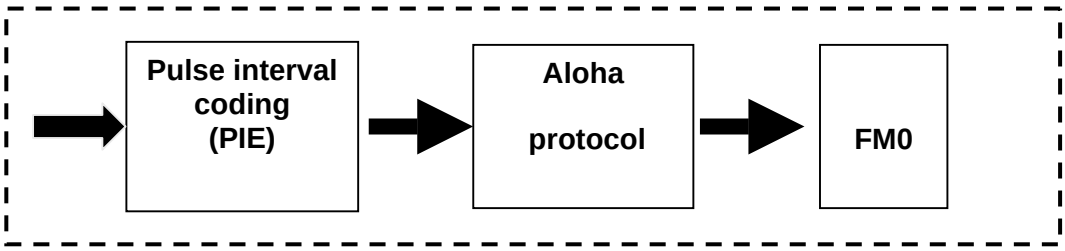


Figure 2 — Type A tag architecture

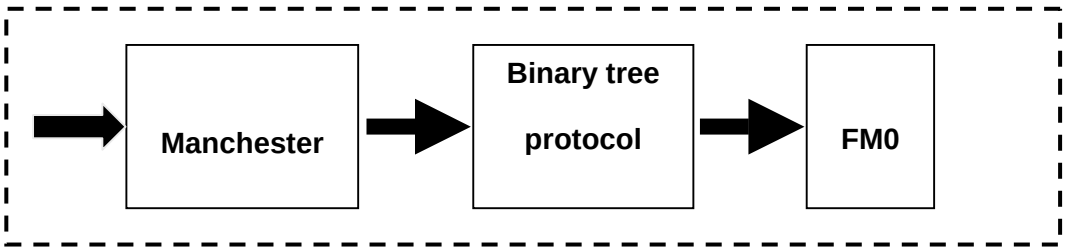


Figure 3— Type B tag architecture

4.1 Parameter tables

The following tables contain the parameter for both types A and B in accordance to CD 18000-1. Detailed description of the operating modes and parameters are specified in the subsequent clauses.

Table 1 – Interrogator to tag link parameters

Interrogator to Tag	Parameter Name	Description
Int: 1	Operating Frequency Range	860 – 930 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 (for Adaptive Frequency Agile and Spread Spectrum systems)	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 Hopping Rate (for Frequency Hopping [FHSS] systems)	Not applicable for single fixed frequency or channelised Adaptive Frequency Agile systems. Where FHSS is permitted, the hop rate shall be in accordance with the local radio regulations.
Int: 1e	Frequency Hopping 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 which 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, but shall in no event exceed 4 Watts EIRP.

Interrogator to Tag	Parameter Name	Description
Int: 4	Interrogator Transmitter Spurious Emissions	In accordance with the local regulations.
Int: 4a	Interrogator Transmitter Spurious Emissions, In-Band	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	Minimum 100μs.
Int: 6b	Receive to Transmit Turn Around Time	As determined by the communication protocol – refer Tag: 6a.
Int: 6c	Interrogator Transmitter RF Power up and settling time from standby	Maximum 1.5ms.
Int: 6d	Decay time of Interrogator Transmitter Power Down ramp	Maximum 1ms.
Int: 7	Modulation	Amplitude Modulation.
Int: 7a	Spreading Sequence (for Spread Spectrum 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 Depth	**Type A**: Nominal 30%. **Type B**: Nominal 11% or 99%
Int: 7e	Duty Cycle	In accordance with local regulations.
Int: 7f	FM Deviation	Not applicable.
Int: 7g	Transmitter modulation pulse negative and positive slopes	The modulation pulse fall and rise times shall conform to the detailed specifications in subsequent clauses.

Interrogator to Tag	Parameter Name	Description
Int: 8	Data Encoding of the modulation	**Type A**: Pulse Interval Encoding **Type B**: Manchester bi-phase
Int: 9	Bit Rate	**Type A**: 33 kbps as constrained by the local radio regulations **Type B**: 10 kbps or 40 kbps 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	Yes
Int:11a	Preamble Length	**Type A**: 4 bits. See relevant clause. **Type B**: 9 bits. See relevant clause.
Int:11b	Preamble Waveform	**Type A**: see relevant clause. **Type B**: see relevant clause.
Int: 11c	Bit Sync Sequence	**Type A**: not applicable. **Type B**: Yes.
Int: 11d	Frame Sync Sequence	Yes
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	Antenna Polarization	Interrogator dependent. Not defined in this standard.

Table 2 — Tag to interrogator link parameters

Tag to Interrogator	Parameter Name	Description
Tag:1	Operating Frequency Range	860 MHz – 930 MHz

Tag to Interrogator	Parameter Name	Description
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	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	200kHz.
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	Minimum 100 µs
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 Spread Spectrum systems)	Not applicable.

Tag to Interrogator	Parameter Name	Description
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 – Tag modulation depth – backscatter modulation	The tag shall have an equivalent Delta RCS (Varying Radar Cross Sectional area) of not less than 0.005m2.
Tag:7e	Sub-carrier Frequency	Not applicable.
Tag:7f	Sub-carrier Frequency Accuracy	Not applicable.
Tag:7g	Sub-Carrier Modulation	Not applicable.
Tag:7h	Duty Cycle	The tag shall transmit its response when commanded to do so by the interrogator.
Tag:7i	FM Deviation	Not applicable
Tag:8	Data Encoding	Bi-phase space (FM0)
Tag:9	Bit Rate	40 kbps
Tag:9a	Bit Rate Accuracy	+/- 15%
Tag:10	Tag Transmit Modulation Accuracy	Not applicable.
Tag:11	Preamble	Yes.
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	Yes. Included in the preamble.
Tag:11d	Frame Sync Sequence	Yes. 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	Antenna Polarization	Product design feature. Not defined in this standard.
Tag:16	Minimum Tag Receiver Bandwidth	860 – 930 MHz

Table 3 - Protocol Parameters

Ref.	Parameter Name	Description
P:1	Who talks first	Interrogator talks first.
P:2	Tag addressing capability	Yes.
P:3	Tag UID	Contained in tag memory and accessible by means of a command.
P:3a	UID Length	**Type A**: 64 bits. A SUID of 40 bits is used during census transactions. **Type B**: 64 bits
P:3b	UID Format	The UID format is different for types A and B. See relevant clauses.
P:4	Read size	**Type A**: Addressable in blocks of 32 bits. **Type B**: Addressable in byte blocks.
P:5	Write Size	**Type A**: Addressable in blocks of 32 bits. **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 (to read the first 128 bits of user memory)	A single tag can typically be identified and have its first 128 bits of user memory read in less than 10ms. 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 20ms. 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**: 11 bit commands: CRC-5. Commands of more than 11 bits, write commands and lock commands: CRC-16. **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 Physical memory (user)	**Type A**: Minimum of 256 bits of user memory arranged in blocks of 32 bits. Accessible by means of read and write commands as defined in relevant clauses. **Type B:** Minimum memory size of 64 bits. Recommended minimum memory size of 128bits.
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 4 — Anti Collision Parameters

Ref.	Parameter Name	Description
A:1	Type (Probabilistic or Deterministic)	**Type A**: Probabilistic using deterministic slot management. **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.

5 Conformance

5.1 Interrogator conformance and obligations

To conform to this standard, the interrogator shall support both 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%.

NOTE Interrogators shall be set up to operate within local regulations.

5.2 Tag conformance and obligations

To conform to this standard, 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 930 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 860-930 MHz band.

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

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

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

6.1 Interrogator power-up waveform

The interrogator power-up waveform shall comply with the mask specified in Figure 4 and Table 5.

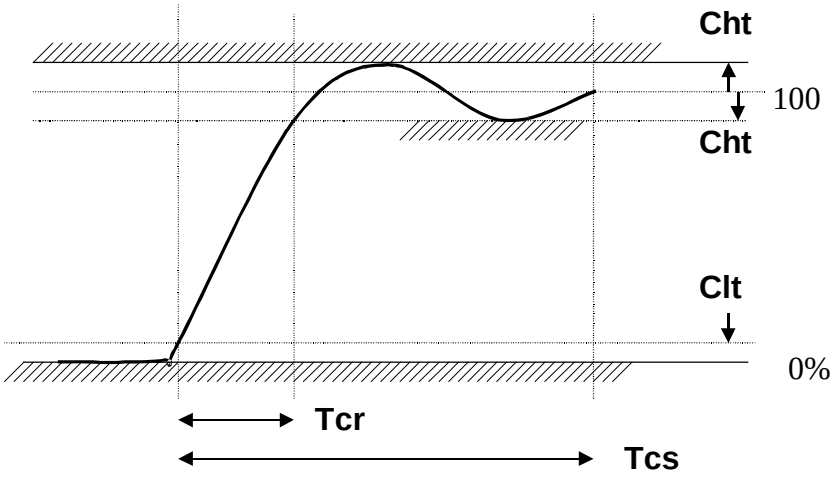


Figure 4 —Interrogator power-up wave form

Table 5 – Interrogator power-up waveform parameter values

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

6.2 Interrogator power-down

Once the carrier level has dropped below the ripple limit Cht, power down shall be monotonic and of duration Tcf, as specified in Figure 5 and Table 6.

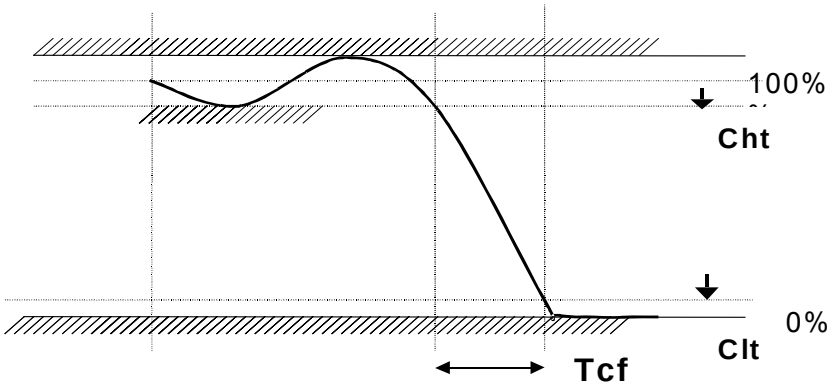


Figure 5 — Interrogator power-down waveform

Table 6 — Interrogator power-down timings

Parameter	Min	Max
Tcf	1 µs	500 µs
Cht		10%
Clt		1%

6.3 Frequency hopping carrier rise and fall times

When the interrogator operates in the frequency operating hopping spread spectrum mode (FHSS), the carrier rise and fall times shall conform to the characteristics specified in Figure 6 and Table 7.

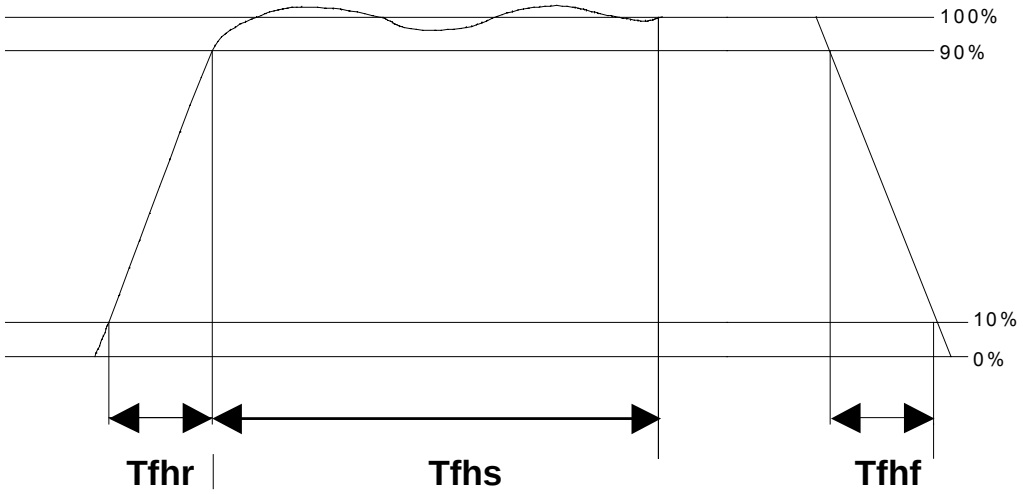


Figure 6 — FHSS carrier rise and fall characteristics

Table 7 – FHSS carrier rise and fall parameters

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

6.4 FM0 return link

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

6.4.1 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.

6.4.2 Data Rate

The data rate is 40kbps.

6.4.3 Data coding

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

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

Table 8 – Return link parameters

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

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

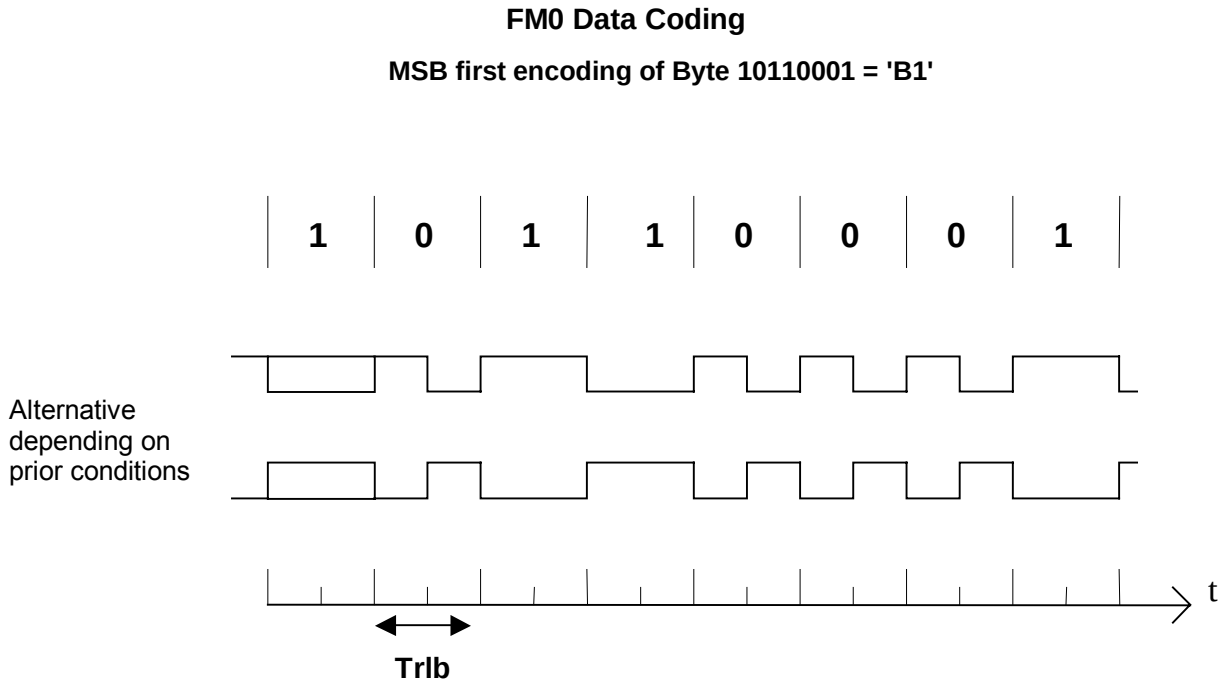


Figure 7 — Tag to interrogator data coding

6.4.4 Message Format

A Return Link Message consists of n data bits preceded by the Preamble and followed by the tag data. 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 the diagram. There are multiple code violations (sequence not conforming to FM0 rules) that act as a frame marker for the transition from Preamble to Data.

6.4.5 Return preamble

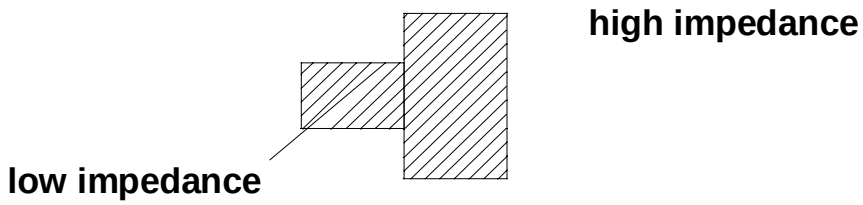
The return preamble is a sequence of backscatter modulation specified in Figure 8.

00 00 01 01 01 01 01 01 01 01 00 01 10 11 00 01

Figure 8 – Return preamble

Data ‘0’ is represented by the tag’s modulator being in the high impedance state, Data ‘1’ is represented by the tag’s modulator switching to the low impedance state, thereby causing a change in the incident energy to be back-scattered.

The tag shall execute backscatter, a half-bit 0 and half-bit 1 sent by the tag defined as follows:



NOTE 1 = low impedance (backscatter), 0 = high impedance (no backscatter),

Figure 9 — Return Link Preamble

6.4.6 Cyclic redundancy check (CRC)

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.

6.4.6.1 Interrogator to tag 5 bit CRC

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^2 +1.

The 5-bit register shall be preloaded with '12' (LSB to MSB is ‘10010’).

The 5 CRC bits shall be sent MSB first. After the MSB 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.

6.4.6.2 Interrogator to tag 16 bit CRC

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.

After the LSB bit is clocked into the tag, the 16 bit CRC register should contain all zero's.

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

6.4.6.3 Tag to interrogator 16 bit CRC

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 '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.

After the LSB bit is clocked into the interrogator, the 16 bit CRC register should contain all zero's.

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.

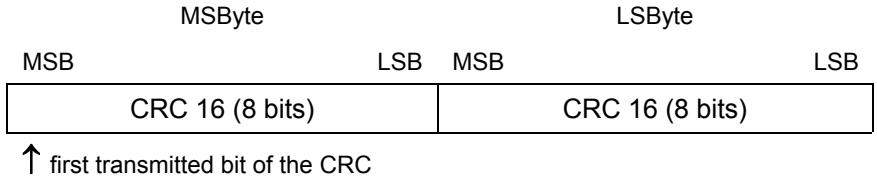


Figure 10 — CRC 16 bits and bytes transmission rules

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 11.

The carrier modulation pulses are negative-going raised cosine shaped, their characteristics are specified in Figure 12.

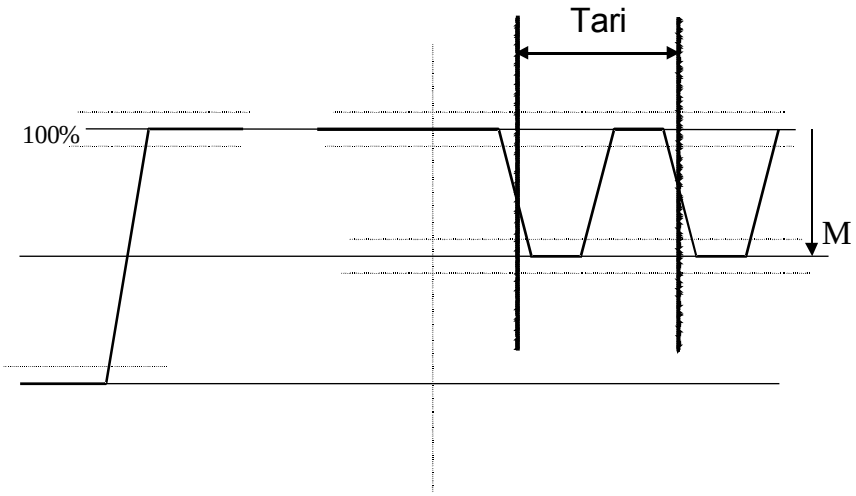


Figure 11 — Inter-pulse mechanism

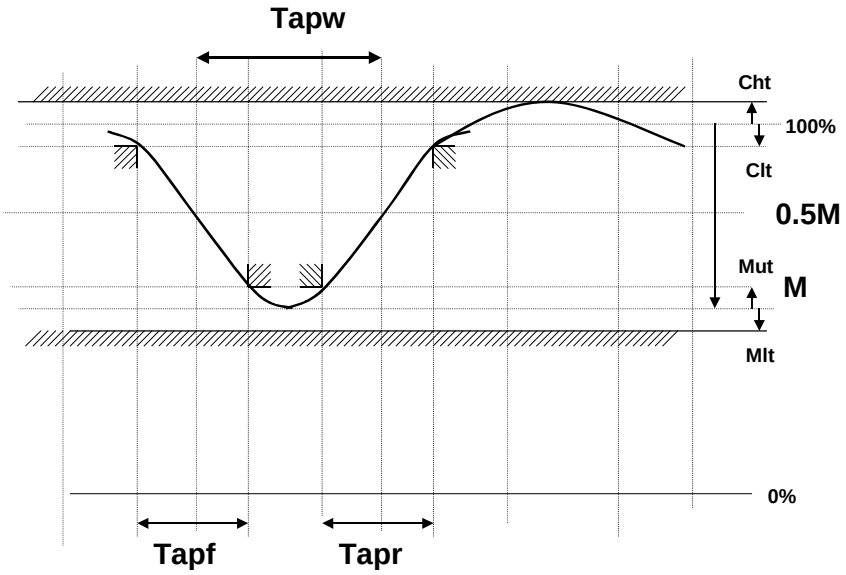


Figure 12 — Modulation shaping

Table 9 — Modulation parameters

Parameter	Min	Nominal	Max
Tapw		10μs	
M	27%	30%	33%
Tapf		4μs	
Tapr		4μs	
Cht			0.1 M
Clt			0.1 M

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

NOTE The min and max modulation depth values are the absolute limits over the interrogator's operating temperature range.

7.1.1.2 Data coding and framing

The time Tari specifies the reference interval time between two pulses, its value is specified in Table 10.

Table 10 — Reference interval timing

Tari	Tolerance
20 μs	±100 ppm

Four symbols encoding are specified, ‘0’, ‘1’ , SOF and EOF. Their meaning is defined by the relative time interval as shown in Table 11 and Figure 13.

Table 11 — PIE symbols

Symbol	Coding duration	Tolerance
0	1 Tari	±100 ppm
1	2 Tari	±100 ppm
SOF	1 Tari followed by 3 Tari	±100 ppm
EOF	4 Tari	±100 ppm

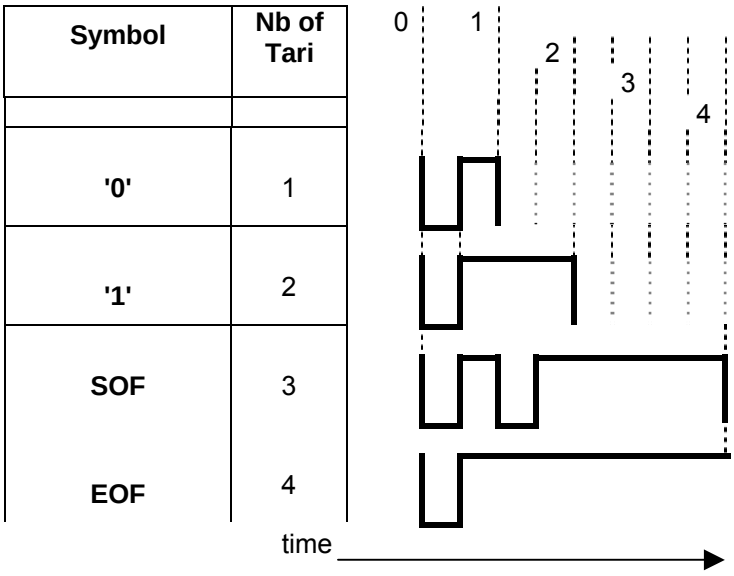


Figure 13 — PIE symbols

7.1.1.3 Frame format

The bits transmitted by the interrogator to the tag are embedded in a frame as specified in Figure 14.

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, immediately followed by the data bits and terminated by an End of frame. 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.

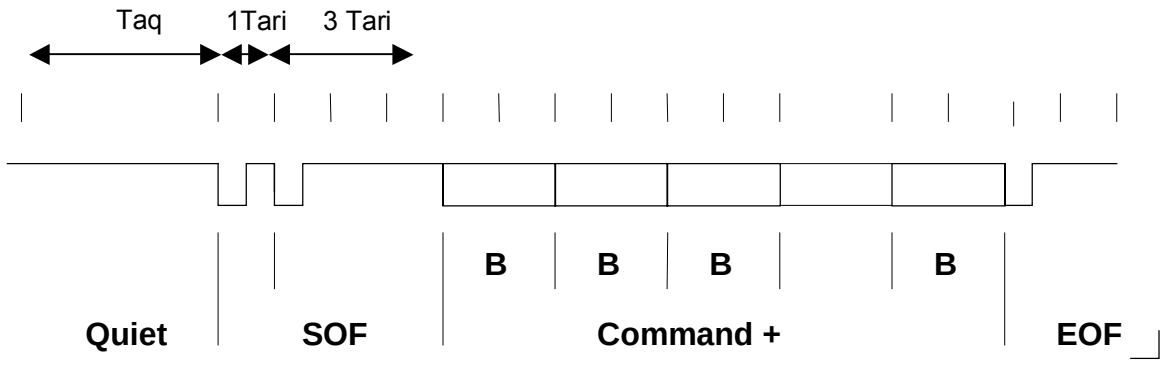


Figure 14 – Forward link frame format

7.1.1.4 Data decoding

The tag shall decode the symbols specified in Table 11.

The tag shall be capable of demodulating an interrogation signal having a modulation depth in the range of 25% to 40%.

If the tag detects an invalid code, it shall discard the frame and wait for an unmodulated carrier of 3Tari duration.

7.1.1.5 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 15.

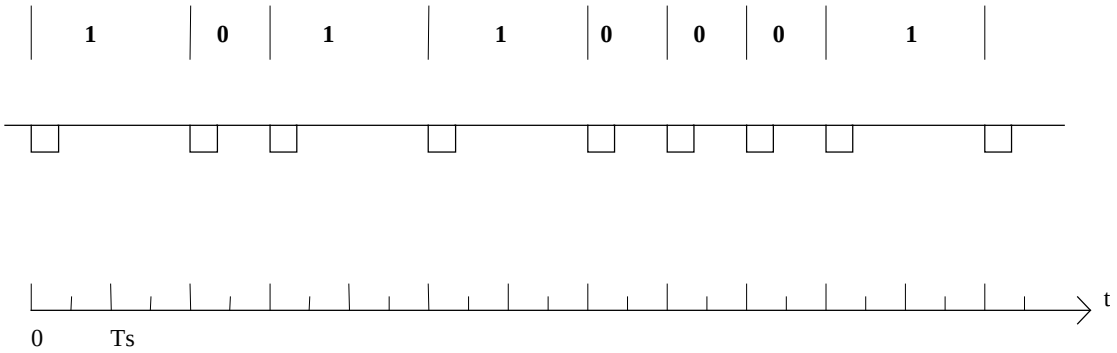


Figure 15 — Example of PIE byte encoding for 'B1'

7.2 Data elements

7.2.1 Unique identifier (UID)

The tags are uniquely identified by a 64 bit unique identifier (UID). This is used for addressing each tag uniquely and individually, during the collision arbitration loop and for one-to-one exchange between an interrogator and a tag.

The UID shall be set permanently by the IC manufacturer in accordance with Figure 16 and in compliance with ISO/IEC DTR 15963.

MSB LSB

64 57	56 49	48 1
'E0'	IC Mfg code	IC manufacturer serial number

Figure 16 — UID format

The UID shall comprise

- The 8 MSB bits defined as 'E0',
- The IC manufacturer code of 8 bits according to ISO/IEC 7816-6/AM1,
- 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 7.6.9.

The SUID consists in 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.

The mapping of the 64 UID to the transmitted 40 bits and back is described in

Figure 17.

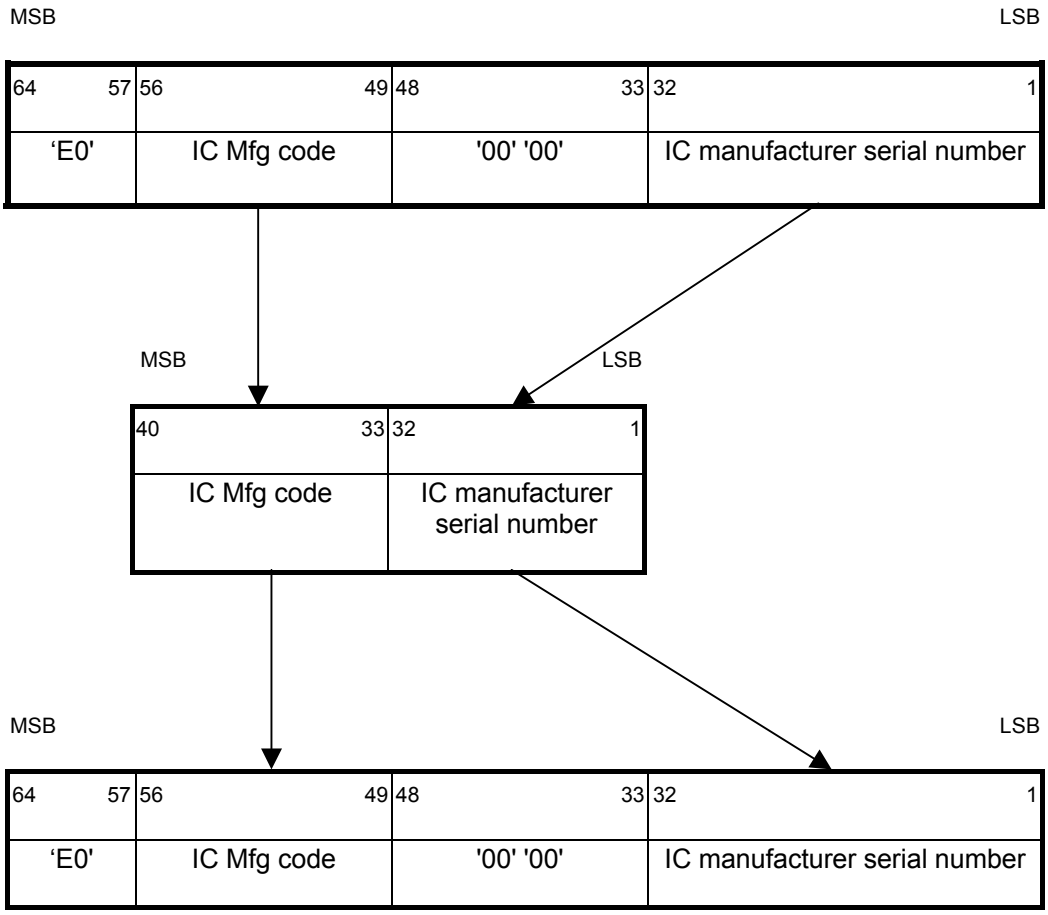


Figure 17 — UID/SUID mapping between 64 to 40

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 17 — UID/SUID mapping between 64 to 40Figure 17.

7.2.3 Application family identifier (AFI)

The Application family identifier comprises one byte of two nibbles of 4 bits each.

The most significant nibble of AFI is used to code one specific or all application families, as specified in Table 12.

The least significant nibble of AFI is used as the Application Sub Family to code one specific or all application sub-families. Sub-family codes other than 0 are allocated by the body identified by the AFI

The AFI (Application Family Identifier) represents the type of application targeted by the interrogator and is used to extract from all the tags present only those tags meeting the required application criteria.

The AFI may be programmed and locked respectively by the Write AFI and the Lock AFI commands.

Table 12 — AFI coding

AFI most significant nibble	AFI least significant nibble	Meaning tags respond from	Examples / note
'0'	'0'	All families and sub-families	No applicative preselection
X	'0'	All sub-families of family X	Wide applicative preselection
X	Y	Only the Yth sub-family of family X	
'0'	Y	Proprietary sub-family Y only	
'1'	'0', Y	Transport	Mass transit, Bus, Airline
'2'	'0', Y	Financial	IEP, Banking, Retail
'3'	'0', Y	Identification	Access control
'4'	'0', Y	Telecommunication	Public telephony, GSM
'5'	'0', Y	Medical	
'6'	'0', Y	Multimedia	Internet services
'7'	'0', Y	Gaming	
'8'	'0', Y	Data storage	Portable files
'9'	'0', Y	EAN.UCC	The sub-family shall be registered by EAN.UCC.
'A'	'0', Y	Express parcels	
'B'	'0', Y	Postal services	
'C'	'0', Y	Airline bags	
'D'	'0', Y	RFU	
'E'	'0', Y	RFU	
'F'	'0', Y	RFU	

NOTE X = '1' to 'F', Y = '1' to 'F'

The support of AFI by the tag is optional.

If AFI is not supported by the tag and if the AFI flag is set, the tag shall not answer whatever the AFI value is in the command.

If AFI is supported by the tag, it shall answer according to the matching rules described in Table 12.

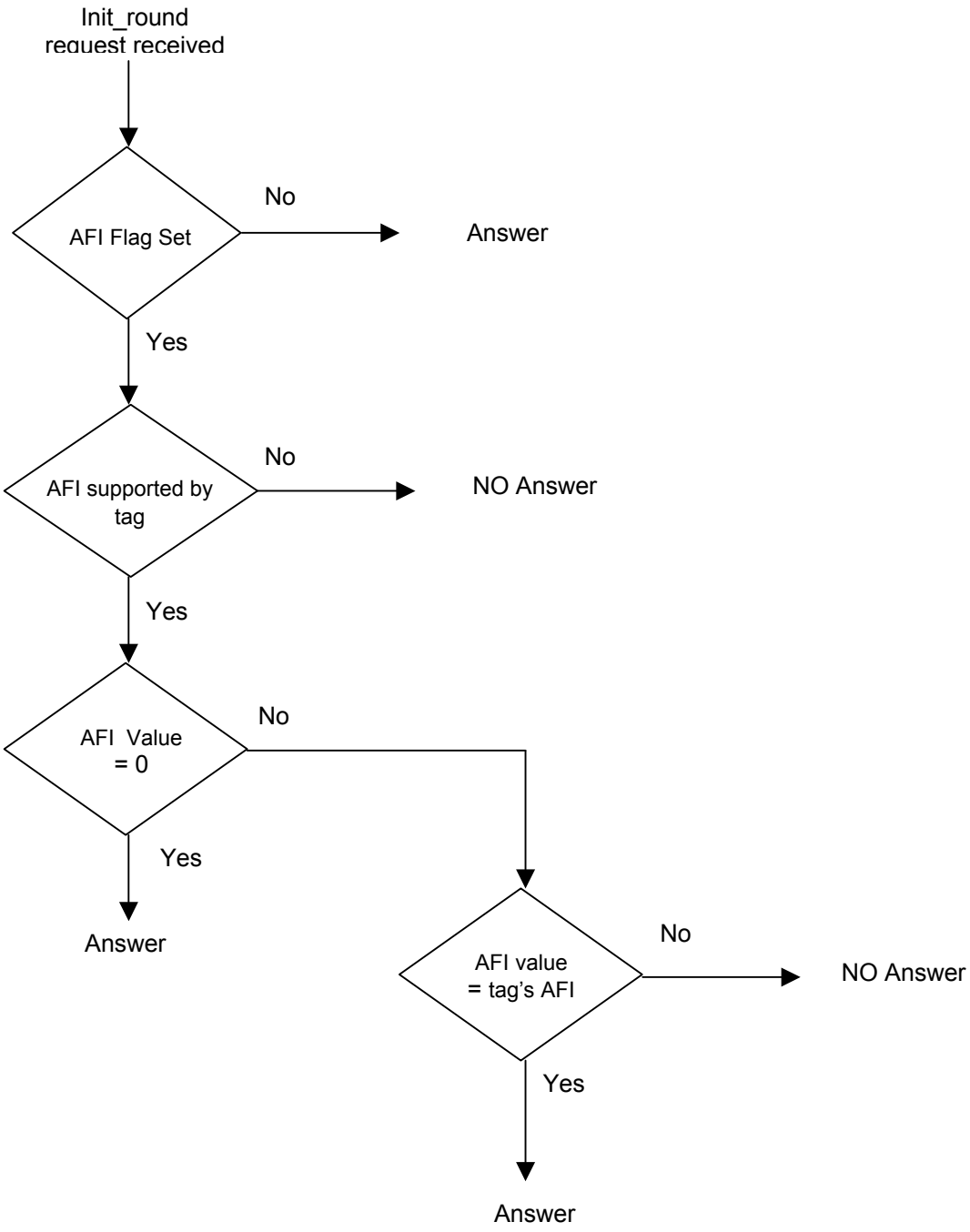


Figure 18 — Tag decision tree for AFI

NOTE "Answer" means that the tag shall answer to the Init_round command.

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 zero ('00').

7.3 Protocol elements

7.3.1 Tag memory organization

The commands specified in this version of this standard assume that the physical memory is organized in blocks of a fixed size.

- Up to 256 blocks can be addressed.
- Block size can be of up to 256 bits.
- This 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 future versions of this Standard.

The commands described in this standard 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 future revision(s) of the standard, or in custom commands).

7.3.2 Support of battery-assisted tags

This standard makes provision for the support of battery-assisted tags.

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

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

a) The type of tag and its sensitivity level are returned in the answer to the Get system information command. See clause 7.6.9

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 7.6.1.

c) When a collision arbitration sequence is initiated, it is possible to specify whether all tags or only passive tags shall participate. See clause 7.6.1.

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 7.7.8 It is coded using two bits.

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_block command.

Factory locking may be performed by a proprietary command.

Table 13— 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 which 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 standard. .

Whatever the chosen implementation, the signature shall be pseudo-random at the system level.

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. If the signatures match, the tag executes the command. If the signatures do not match, the tag shall behave in accordance with the state diagram.

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 from the interrogator a command, correctly decoded, before transmitting.

a) The protocol is based on an exchange of

- A command from the interrogator to the tag
- A response from the tag(s) to the interrogator

The conditions under which the tag sends a response are defined in clause 10.

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

- A command code

- Flags
- Mandatory and optional parameter fields, depending on the command
- Application data fields
- Checksum or CRC

c) Each response consists of the following fields:

- Flags
- Mandatory and optional parameter fields, depending on the command
- Application data fields
- CRC

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

The command consists of the following fields:

- One RFU bit, reserved for protocol extension
- A command code (see clause 7.4.10) on 6 bits
- Command flags (see clause 7.4.3) on 4 bits
- Parameter and data fields
- A CRC-16 or a CRC-5 depending on the number of bits of the command (see clause 7.4.10)

RFU flags	Command code	Command flags	Parameters	Data	CRC-5 or CRC-16

Figure 19 — General command format

7.4.3 Command flags

In a command, the "Command flags" field specifies the actions to be performed by the tag and those fields that are or not present.

It consists of four bits. The meaning of bits b2, b3 and b4 depends of the value of the Census flag (b1).

Table 14 — Command flag 1 definition

Bit	Flag name	Value	Description
b1	Census_flag	0	Command shall be executed outside of the collision arbitration scheme
		1	Command shall be executed within the collision arbitration scheme

7.4.3.1 Census flag

Table 15 — Command flags 2 to 4 definition when the Census flag is NOT set

Bit	Flag name	Value	Description
b2	Select_flag	0	Command shall be executed by any tag according to the setting of Address_flag
		1	Command shall be executed only by tag in selected state. The Address_flag shall be set to 0 and the SUID field shall not be included in the command.
b3	Address_flag	0	Command is not addressed. SUID field is not included. It shall be executed by any tag.
		1	Command is addressed. SUID field is included. It shall be executed only by the tag whose SUID matches the SUID specified in the command.
b4	RFU	0	RFU. Shall be set to 0
		1	RFU

Table 16 — Command flags 2 to 4 when the Census_flag is set

Bit	Flag name	Value	Description
b2	Slot_delay_flag	0	The tag shall answer immediately after the beginning of the slot
		1	The tag shall answer after a delay following the beginning of the slot
b3	AFI_flag	0	AFI field is not present
		1	AFI field is present
b4	SUID_flag	0	The tag shall not include the SUID in its response and shall return the first 128 bits of its memory.
		1	The tag shall include the SUID in its response.

When the Census_flag is set ('1'), the round size shall always be present as the first parameter of the interrogator command. The coding of the round size is specified in clause 7.4.5.

7.4.3.2 Select_flag

When the Select_flag is set to 1 (select mode), the command shall not contain a tag SUID.

The tag in the selected state receiving a command with the Select_flag set to 1 shall execute this command (if possible) and shall return a response to the interrogator as specified by the command description.

Only the tag in the selected state shall answer to a command having the select flag set to 1.

7.4.3.3 Address_flag

When the Address_flag is set to 1 (addressed mode), the command shall contain the SUID of the addressed tag.

On receiving a command with the Address_flag set to 1 the tag shall compare the received SUID (address) to its own SUID.

If the SUID matches, the tag shall execute the command (if possible) and return the requested response to the interrogator.

If the SUID does not match, the tag shall remain silent.

When the Address_flag is set to 0 (non-addressed mode), the command shall not contain a SUID.

On receiving a command with the Address_flag set to 0 the tag shall execute the command (if possible) and shall return the requested response to the interrogator.

7.4.3.4 Slot_delay_flag

If the Slot_delay flag is not set ('0'), the tag shall answer at the beginning of the slot it has selected. See clause 7.5.2.

If the Slot_delay flag is set, the tag shall delay the transmission of its response for a pseudo-random time that shall be computed as a finite number of bits in the range of 0 to 7 bits.

Therefore the average delay is equivalent to the transmission time of 3.5 bits.

NOTE 1 Due to the tag internal clock tolerance of the tag, two tags sending their response after the same number of bits (e.g. 6) will actually not transmit exactly at the same time.

NOTE 2 This standard does not specify the mechanism for generating the random delay before the transmission of the tag‘s response. It only requires that this mechanism must allow a response of the tag in one of the slots, in a pseudo-random manner. For instance, the tag may use its TEL (if supported), the last LSBs of its UID, or a pseudo random generator.

7.4.3.5 SUID_flag

If the SUID_flag is not set ('0'), the tag shall return the first 128 bits of its user memory. It shall not include the SUID in its response.

If the SUID_flag is set ('1'), the tag shall return the SUID.

See details of the response format in clause 7.6.1.

7.4.4 Battery_flag

This flag is used only by the Init round command.

If the Battery_flag is set by the interrogator in the Init_round command, the tag shall process the command whether it is battery-assisted or battery-less.

If the Battery_flag is not set, the tag shall process the command only if it is battery-less, otherwise it shall discard it and not respond.

7.4.5 Repeat_round flag

This flag is used only by the Init round, New round and Close slot commands.

If the Repeat-round flag is set in the received Init_round command, the tag shall commence the collision arbitration process by selecting at random a slot in which to transmit its response. At the end of the round (i.e. when all slots have elapsed) , if the tag is in the Round_active state, it shall automatically enter a further round. The round size shall remain the same. The tag shall select at random a new slot in which to respond.

If the Repeat-round flag is NOT set in the received Init_round command, the tag shall commence the collision arbitration process by selecting at random a slot in which to transmit its response. If, having transmitted its response in that slot, the tag receives a Next_slot command with matching signature, it shall move to the Quiet state. If it does not receive a Next_slot command with matching signature, it shall move to the Ready state at the end of the current round. The tag shall remain in the Ready state until it receives a New_round or Init_round command at which time it shall select a new slot and enter a new round.

7.4.6 Round size

The round size is coded using 3 bits according to Table 17.

Table 17 — 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

7.4.7 Commands

7.4.8 Command code definition and structure

The size of the command code is 6 bits.

7.4.9 Command classes

Four sets of commands are defined: mandatory, optional, custom, proprietary.

Table 18 — Command classes

Code	Class	Number of possible codes
'00' –'0F'	Mandatory	16
'10 – '27'	Optional	24
'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.

7.4.9.1 Mandatory

The command codes range from '00' to '0F'.

All tags shall support them.

7.4.9.2 Optional

The command codes range from '10' to '27'. The tag may support optional commands, at its option. If supported, command and response formats shall comply with the definition given in this standard.

If the tag does not support an optional command and if the Address_flag or the Select_flag is set, it may return an error code ("Not supported") or remain silent. If neither the Address_flag nor the Select_flag is set, the tag shall remain silent.

7.4.9.3 Custom

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

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

If the tag does not support a custom command and if the Address_flag or the Select_flag is set, it may return an error code ("Not supported") or remain silent. If neither the Address_flag nor the Select_flag is set, the tag shall remain silent.

7.4.9.4 Proprietary

The command codes range from '38' to '3F'.

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 standard. The IC manufacturer may at its option document them or not. This standard allows these commands to be disabled after IC and/or tag manufacturing.

7.4.10 Command codes and CRC

Table 19 specifies the command codes and the CRC-5 or CRC-16 that the interrogator shall append to each command.

Table 19 — Command codes

Command code	Type	Function	CRC-5 (clause 6.4.6.1) or CRC-16 (clause 6.4.6.2)
'00'	Mandatory	RFU	RFU
'01'	Mandatory	Init_round	CRC 16 bits
'02'	Mandatory	Next_slot	CRC 5 bits
'03'	Mandatory	Close_slot	CRC 5 bits
'04'	Mandatory	Standby_round	CRC 5 bits
'05'	Mandatory	New_round	CRC 5 bits
'06'	Mandatory	Reset_to_ready	CRC 5 bits
'07'	Mandatory	Select (by SUID)	CRC 16 bits
'08'	Mandatory	Read blocks	CRC 16 bits
'09'	Mandatory	Get system information	CRC 16 bits
'0A' – '0F'	Mandatory	RFU	RFU
'10'	Optional	Write block	CRC 16 bits
'11'	Optional	Write multiple blocks	CRC 16 bits
'12'	Optional	Lock_block	CRC 16 bits
'13'	Optional	Write AFI	CRC 16 bits
'14'	Optional	Lock AFI	CRC 16 bits
'15'	Optional	Write DSFID	CRC 16 bits
'16'	Optional	Lock DSFID	CRC 16 bits
'17'	Optional	Get block lock status	CRC 16 bits
'18' – '27'	Optional	RFU	RFU
'28' – '37'	Custom	IC Mfg dependent	IC Mfg dependent
'38' – '3F'	Proprietary	IC Mfg dependent	IC Mfg dependent

7.4.11 Response format

The response from the tag consists of the following fields:

- Flags
- One or more parameter fields
- Data
- CRC

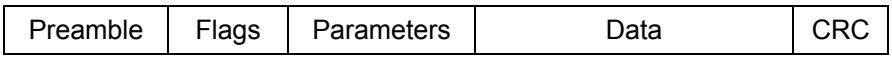


Figure 20 — General response format

7.4.12 Response flags

In a response, two flags indicate how actions have been performed by the tag and whether corresponding fields are present or not.

Table 20 — Response flags 1 to 2 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.

7.4.13 Response error code

The error code comprises four bits.

When the Error_flag is set by the tag, the error code field shall be included and provides information about the error that occurred. Error codes are specified in Table 21.

If the tag does not support specific error code(s) listed in table 7, it shall answer with the error code 'F' ("Error with no information given").

Table 21 — Response error codes

Code	Description
'0'	RFU
'1'	The command is not supported, i.e. the command code is not recognized
'2'	The command is not recognized, 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.

7.4.14 Tag states

A tag can be in one of the following states:

- RF field off
- Ready
- Quiet
- Selected
- Round_active
- Round_standby

The transition between these states is specified in tables in each command description.

7.4.14.1 RF field off

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.

7.4.14.2 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.

7.4.14.3 Quiet state

When in the Quiet state, the tag shall process any command where the Census_flag is not set and where the Address_flag is set.

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 round.

7.4.14.4 Selected state

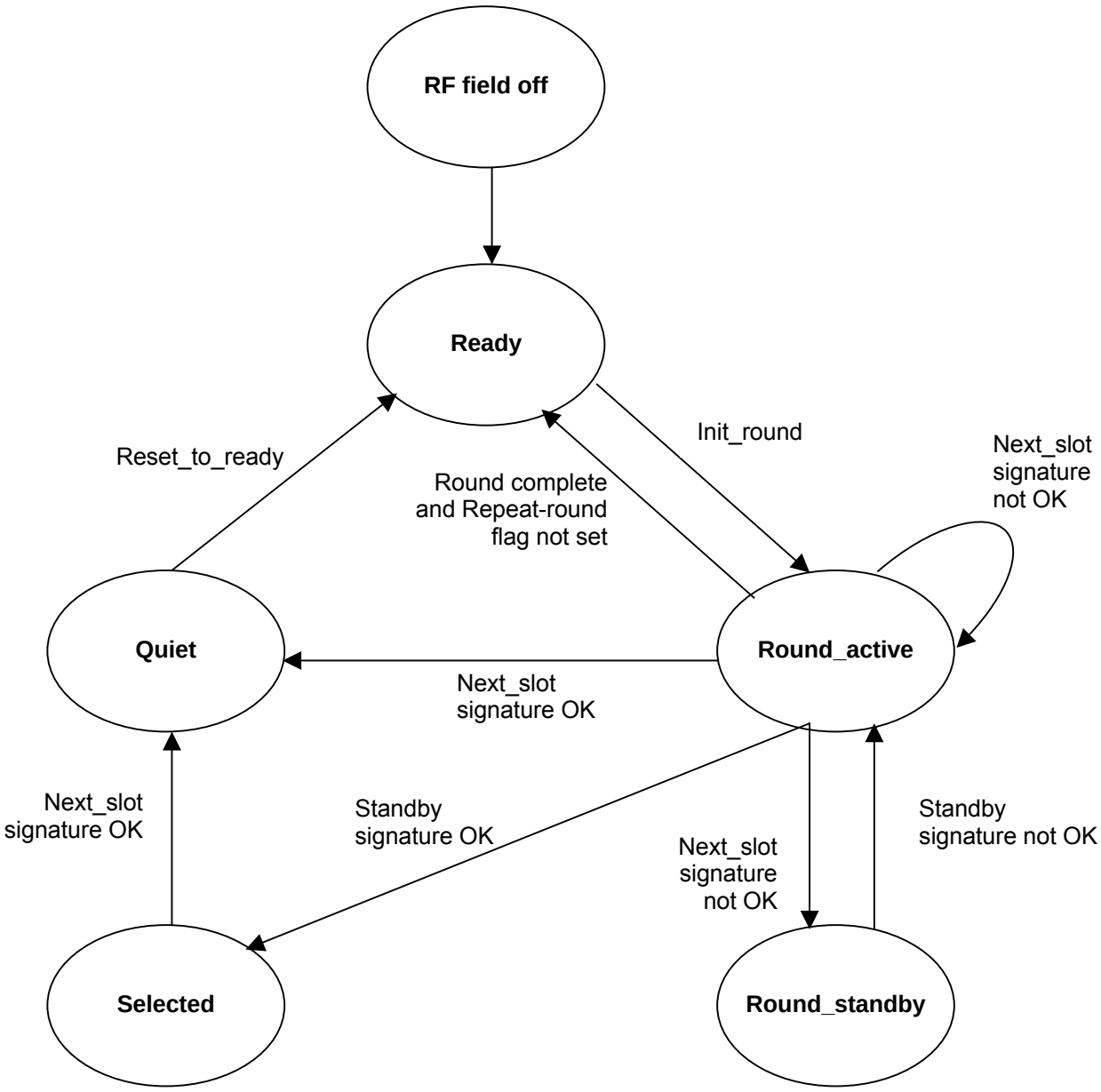
Only a tag in the Selected state shall process commands having the Select_flag set.

7.4.14.5 Round_active

When in the Round_active state, the tag shall participate in the collision arbitration described in clauses 7.4.15 and 7.4.16.

7.4.14.6 Round_standby

When in the Round_standby state, the tag shall suspend its participation to the collision arbitration as described in clauses 7.4.15 and 7.4.16.



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

Figure 21 — Tag state transition diagram

7.4.15 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 flags 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.

7.4.16 General explanation of the collision arbitration mechanism

The collision arbitration uses a mechanism which 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 actual duration of a slot is determined by the interrogator.

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 process by sending an Init_round 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.

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 a tag has selected the first slot and the slot_delay flag is not set, it will transmit its response immediately.

If the Slot_delay flag is set, then it will wait for a pseudo-random time delay equal to a time of between 0 and 7 tag bit 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 which 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
- verifies that the Next_slot command has been received within the time window specified in clause 7.5.4

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 Round_standby 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.

The interrogator keeps track of the slot count each time it issues a Close_slot or Next_slot command. When the number of slots used equals the Round_size issued in the Init_round command, the round has terminated. Where the Repeat_round flag was not previously set in the Init_round command, the interrogator may initiate a new round by issuing an Init_round or New_round command.

Where the Repeat-round flag was previously set in the Init_round command, the tag may continue the round as specified in clause 7.4.5.

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

7.5 Timing specifications

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

7.5.1 Tag state storage

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

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

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

7.5.2 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 22.

If the Slot_delay flag is set, this time is increased as specified in clause 7.4.3.4.

Table 22 — Forward link to return link handover timing

Slot delay flag	Minimum	Maximum
Not set	150µs	450µs
Set	150µs	650µs

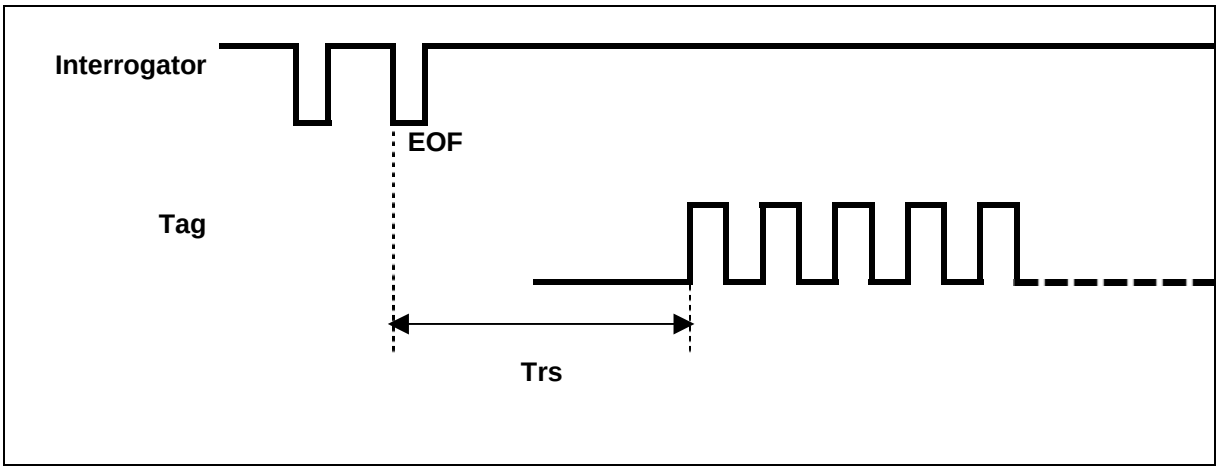


Figure 22 - Forward link to return link handover

7.5.3 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.

7.5.4 Acknowledgement time window

The purpose of the acknowledgement time window is to ensure that only those tags which 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.

7.5.4.1 Interrogator

The interrogator shall send a Next_slot or a Round_standby 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. This window shall open from the end of the last tag bit period. See Figure 23.

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’.

7.5.4.2 Tag

When a tag is in the Round active state and has responded in the current slot, it shall discard any Next slot and Round standby commands if they are not received within the window specified in clause above.

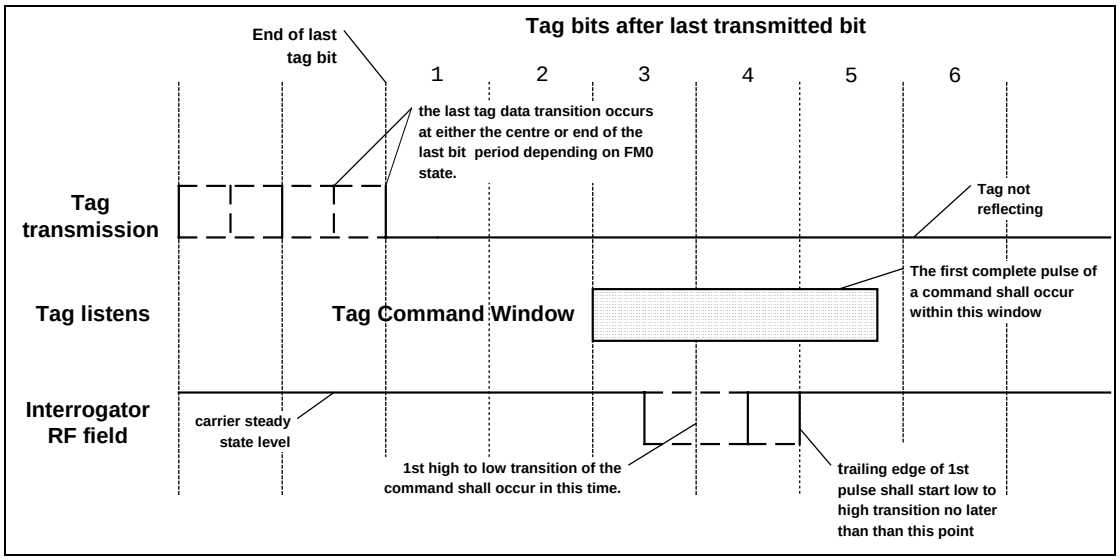


Figure 23 — Acknowledgement time window

7.6 Mandatory commands

7.6.1 Init_round

Command code = '01'

The command contains:

- The protocol extension flag of 1 bit
- The Init_round command code of 6 bits
- The command flags of 4 bits
- The Battery flag of 1 bit
- The Repeat round flag
- The Round size of 3 bits
- The AFI if the AFI flag is set
- The CRC-16

The Census_flag shall be set to 1.

The meaning of the flags is specified in clause 7.4.3.

NOTE The setting of the Repeat_round flag determines the action to be performed by the tag at the end of the round. If the Repeat_round flag is set, a tag which is in the Round_active state at the end of a round shall automatically enter a new round, otherwise it shall return to the Ready_state.

See clause 7.4.5.

Protocol extension	Init_round	Flags	Battery flag	Repeat round flag	Round size	Optional AFI	CRC-16
1 bit	6 bits	4 bits	1 bit	1 bit	3 bits	8 bits	16 bits

Figure 24 — Init_round command format

The response 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
- The first 128 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.

Flags	Signature	Tag type	Battery status	Random number	First 128 bits of memory
2 bits	4 bits	1 bit	1 bit	6 bits	128 bits

Figure 25 — Init_round response format when the SUID flag is NOT set

This standard does not specify how the 6 bit random number shall be generated (It could for instance be the last 6 significant bits of the UID, or generated by a random generated number, etc..).

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

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 10 bits random number.

NOTE 2 The DSFID field is not included since in situations where the first 128 bits of memory are read, it is expected that no further exchange with the tag will take place. The DSFID can nevertheless be determined using other commands.

Flags	Signature	Tag type	Battery status	DSFID	SUID
2 bits	4 bits	1 bit	1 bit	8 bits	40 bits

Figure 26 — Init_round response format when the SUID flag is set

NOTE The 6 bits random number is not required when a SUID is sent back since the collisions will be detected using the combination of the signature and the SUID.

Table 23 – Tag type (battery-assisted or not)

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

Table 24 – 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.

Table 25 – Tag state transition for Init_round

Command : Init_round			
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 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**
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.

7.6.2 Next_slot

Command code = '02'

The Next_slot command has two functions:

- It acknowledges the tag which has been identified.
- 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
- The tag signature

No flags are used by this command.

Protocol extension	Next_slot	Tag signature
1 bit	6 bits	4 bits

Figure 27 — Next_slot command format

There is no tag response to the Next_slot command. The tag shall perform the actions specified in Table 26. It may either enter the Quiet state or switch to the next slot by incrementing their slot counter.

Table 26 – 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 7.5.4	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 7.5.4	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**

7.6.3 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

It shall be sent by the interrogator when no tag response has been received in the slot or when a collision has been detected.

The command contains:

- The Close_slot command code
- The Repeat_round flag
- 3 RFU bits. These bits shall be set to 0

Protocol extension	Close_slot	Repeat round	RFU
1 bit	6 bits	1 bit	3 bits

Figure 28 — Close_slot command format

There is no 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 their slot counter and send its response if its slot counter matches the chosen slot.

Table 27 – Tag state transition for Close_slot

Command : Close_slot			
Current state	**Criteria**	**Action**	**New state**
Ready	None	None	Ready
Quiet	None	None	Quiet
Selected	None	None	Selected
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**

7.6.4 Round_standby

Command code = '04'

The Round_standby 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 by setting the Select flag.
- It instructs all other tags in the Round_active state to enter the Standby_round state. For these tags, their participation in the round is suspended. The round will be resumed by a Next_slot command, a Close_slot command or a New_round command.

The command contains:

- The Round_standby command code
- The tag signature

No flags are used by this command.

Protocol extension	Round_ standby	Tag signature
1 bit	6 bits	4 bits

Figure 29 — Round_standby command format

There is no tag response to the Round_standby command. The tag shall perform the algorithm described in clauses 7.4.15 and 7.4.16. It may either enter the Select state or the Round_standby state.

Table 28 – Tag state transition for Round_standby

Command : Round_standby			
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 7.5.4	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 7.5.4.	None	**Round_standby**
Round_standby	None	None	Round_standby

7.6.5 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.
- It instructs the tag in the Selected state to enter the Quiet state. Tags in the Quiet state shall remain in this state.

The command contains:

- The New_round command code
- The Repeat_round flag
- The new round size

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

Protocol extension	New_round	Repeat round flag	Round size
1 bit	6 bits	1 bit	3 bits

Figure 30 — New_round command format

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

Table 29 – 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**

7.6.6 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
- Four RFU bits. These bits shall be set to zero.

No flags are used by this command.

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.

Protocol extension	Reset_to_ ready	RFU bits
1 bit	6 bits	4 bits

Figure 31 — Reset_to_ready command format

There is no response to the Reset_to_ready.

Table 30 – 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**

7.6.7 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. By setting the Select_flag in individual commands, they can be then be sent to this tag (such as Read and Write).
- 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 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.
    - 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. It shall not send a response. The Select command shall always be executed in Addressed mode. (The Select_flag is set to 0. The Address_flag is set to 1.)

Command parameter:

- SUID (mandatory)

Protocol extension	Select	SUID
1 bit	6 bits	40 bits

Figure 32 — Select command format

Flags	Error Code
2 bits	4 bits

Figure 33 — Select response format when Error_flag is set

Flags
2 bits

Figure 34 — Select block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 31 – 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**

7.6.8 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.

E.g. 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.

Protocol extension	Read multiple block	Flags	SUID	First block number	Number of blocks
1 bit	6 bits	4 bits	40 bits	8 bits	8 bits

Figure 35 — Read blocks command format

Command parameter:

(Optional) SUID (if the Address_flag is set)

First block number

Number of blocks

Flags	Error code
2 bits	4 bits

Figure 36 — Read multiple blocks response format when Error_flag is set

Flags	Block security status	Data
2 bits	2 bits	Block length
	Repeated as needed	

Figure 37 — Read multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

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.

Table 32 – Tag state transition for Read blocks

Command: Read multiple 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.6.9 Get system information

Command code = '09'

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

Protocol extension	Get system info	Flags	SUID
1 bit	6 bits	4 bits	40 bits

Figure 38 — Get system information command format

Command parameter:

(Optional) SUID

Flags	Error code
2bits	4 bits

Figure 39 — Get system information response when Error_flag is set

Flags	Info flags	UID	DSFID	AFI	Other fields
2 bits	10 bits	64 bits	8 bits	8 bits	See Table 33 and Table 34

Figure 40 — Get system information response format when Error_flag is NOT set

NOTE The complete UID on 64 bits shall be returned.

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set

Information flag

UID (mandatory)

Information fields, in the order of their corresponding flag, as defined in Table 33 and Table 34 if their corresponding flag is set.

Table 33 — Information flags 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 Annex C.
		10	Tag sensitivity is S2. See Annex C.
		11	Tag sensitivity is S3. See Annex C.
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

Table 34 — Tag memory size information

MSB LSB

16 14	13 9	8 1
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.

Table 35 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7 Optional commands

The following commands are optionally supported by the tag. If the tag does not support them, it shall return the error code "not supported".

7.7.1 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.

Protocol extension	Write single block	Flags	SUID	Block number	Data
1 bit	6 bits	4 bits	40 bits	8 bits	Block data

Figure 41 — Write single block command format

Command parameter:

(Optional) SUID

Block number

Data

Flags	Error code
2 bits	4 bits

Figure 42 — Write single block response format when Error_flag is set

Flags
2 bits

Figure 43 — Write single block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 36 – Tag state transition for Write_block

Command: Write 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.2 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.

E.g. 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.

Write multiple block	Flags	SUID	First block number	Number of blocks	Data
6 bits	4 bits	40 bits	8 bits	8 bits	Block length
					Repeated as needed

Figure 44 — Write multiple blocks command format

Command parameter:

(Optional) UID

First block number

Number of blocks

Block data (repeated as defined in Figure 44)

Flags	Error code
2 bits	4 bits

Figure 45 — Write multiple blocks response format when Error_flag is set

Flags
2 bits

Figure 46 — Write multiple block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 37 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.3 Lock single block

Command code = '12'

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

Protocol extension	Lock block	Flags	SUID	Block number
1 bit	6 bits	4 bits	40 bits	8 bits

Figure 47 — Lock single block command format

Command parameter:

(Optional) SUID

Block number

Flags	Error code
2 bits	4 bits

Figure 48 — Lock block response format when Error_flag is set

Flags
2 bits

Figure 49 — Lock block response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 38 – Tag state transition for Lock single block

Command : Lock 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.4 Write AFI

Command code = '13'

On receiving the Write AFI command, the tag shall write the AFI value into its memory.

Protocol extension	Write AFI	Flags	SUID	AFI
1 bit	6 bits	4 bits	40 bits	8 bits

Figure 50 - Write AFI command format

Command parameter:

(Optional) SUID

AFI

Flags	Error code
2 bits	4 bits

Figure 51— Write AFI response format when Error_flag is set

Flags
2 bits

Figure 52 — Write AFI response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 39 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.5 Lock AFI

Command code = '14'

On receiving the Lock AFI command, the tag shall lock the AFI value permanently into its memory.

Protocol extension	Lock AFI	Flags	SUID
1 bit	6 bits	4 bits	40 bits

Figure 53 — Lock AFI command format

Command parameter:

(Optional) SUID

Flags	Error Code
2 bits	4 bits

Figure 54 — Lock AFI response format when Error_flag is set

Flags
2 bits

Figure 55 — Lock AFI response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 40 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.6 Write DSFID command

Command code = '15'

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

Protocol extension	Write DSFID	Flags	SUID	DSFID
1 bit	6 bits	4 bits	40 bits	8 bits

Figure 56 — Write DSFID command format

Command parameter:

(Optional) SUID

DSFID

Flags	Error code
2 bits	4 bits

Figure 57 — Write DSFID response format when Error_flag is set

Flags
2 bits

Figure 58 — Write DSFID response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 41 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

7.7.7 Lock DSFID

Command code = '16'

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

Protocol extension	Lock DSFID	Flags	SUID
1 bit	6 bits	4 bits	40 bits

Figure 59 — Lock DSFID command format

Command parameter:

(Optional) SUID

Flags	Error code
2 bits	4 bits

Figure 60 — Lock DSFID response format when Error_flag is set

Flags
2 bits

Figure 61 — Lock DSFID response format when Error_flag is NOT set

Response parameter:

Error_flag (and Error code if Error_flag is set)

Table 42 – 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 Select_flag is not set	none	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	none	none	Round_active
Round_standby	none	none	Round_standby

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

The number of requested blocks shall be one less than the number of block lock status that the tag shall return in its response.

E.g. a value of '06' in the "Number of blocks" field requests to return 7 Block lock status. A value of '00' in the "Number of blocks" field requests to return a single Block lock status.

Protocol extension	Get block lock status	Flags	SUID	First block number	Number of blocks
1 bit	6 bits	4 bits	40 bits	8 bits	8 bits

Figure 62 — Get blocks lock status command format

Command parameter:

(Optional) SUID (if the Address_flag is set)

First block number

Number of blocks

Flags	Block lock status	CRC16
2 bits	2 bits	16 bits
	Repeated as needed	

Figure 63 — Get blocks lock status response format when Error_flag is NOT set

Flags	Error code
2 bits	4 bits

Figure 64 — Get blocks lock status format when Error_flag is set

Response parameter:

Error_flag (and Error code if Error_flag is set)

if Error_flag is not set

Block lock status (repeated as per)

Table 43 – Tag state transition for Get blocks lock status

Command : Get multiple block security 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 Select_flag is not set	None	Selected
	The Select_flag is set	The tag shall process the command and send back its response	
Round_active	None	None	Round_active
Round_standby	None	None	Round_standby

7.8 Custom commands

The format of custom command 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.

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

Figure 65 — Custom command format

Command parameter:

IC manufacturer code according to ISO/IEC 7816-6/AM1.

Flags	Error code
2 bits	4 bits

Figure 66 — Custom response format when Error_flag is set

Flags	Custom response parameters
2 bits	Custom defined

Figure 67 — Custom response format when Error_flag is NOT set

Response parameter:

Error_flag (and code if Error_flag is set)

if Error_flag is not set

Custom parameters

7.9 Proprietary commands

The format of these commands is not defined in this standard

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.

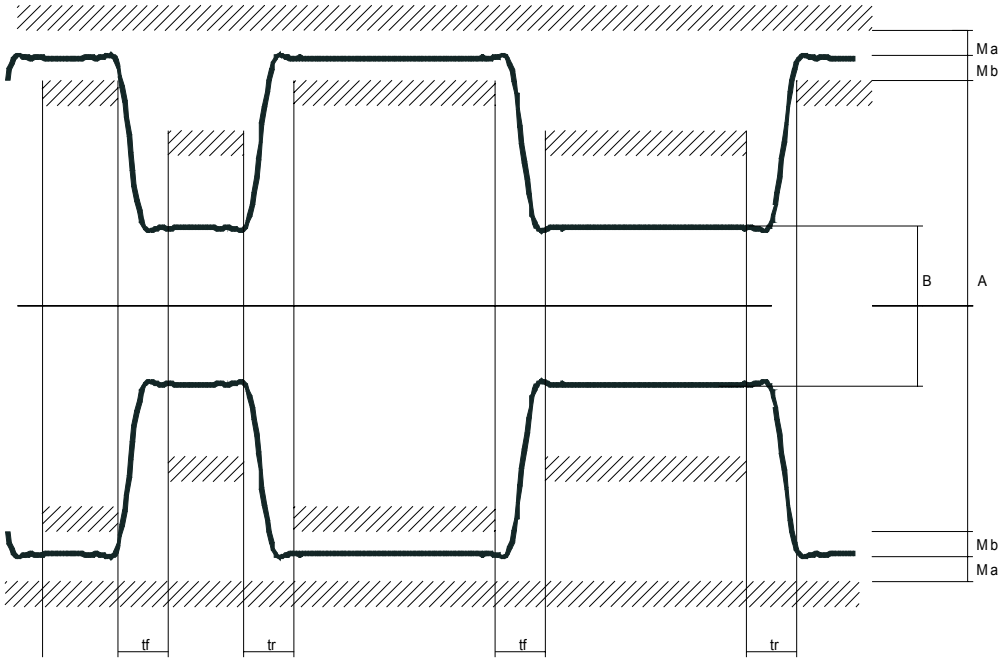


Figure 68 — 99% Modulation (Example of 40 kbps signal)

Table 44 — Parameter for 99% Modulation

Parameter	Minimum	Nominal	Maximum
M = (A-B)/(A+B)	90	99	100
Ma	0		0.03 (A-B)
Mb	0		0.03 (A-B)
tr	0 µs	1.8 µs	35 µs
tf	0 µs	1.8 µs	35 µs

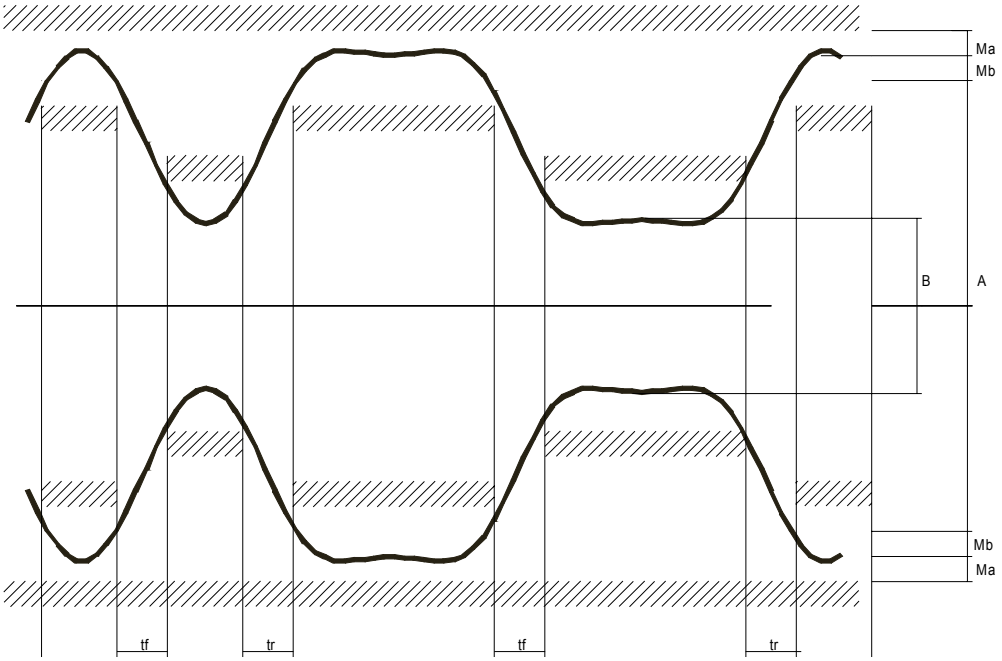


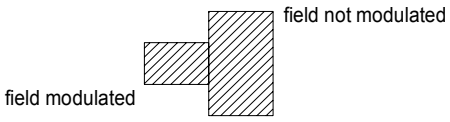
Figure 69 — 11% Modulation (Example of 8 kbps signal)

Table 45 — Parameter for 11% Modulation

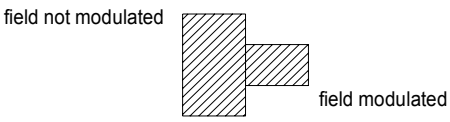
Parameter	Minimum	Nominal	Maximum
M = (A-B)/(A+B)	8 %	11 %	15 %
Ma	0		0.03 (A-B)
Mb	0		0.03 (A-B)
tr	0 µs		35 µs
tf	0 µs		35 µs

8.1.1.2 Bit coding of forward link fields

Data is Manchester encoded as per Figure 70.



Logic 0 = Manchester 0 … 01



Logic 1 = Manchester 1 … 10

Figure 70 — Forward link bit coding

8.1.2 Return link

See clause 6.4.

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 on-off ratio 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 8.2.6.

Each command and each response are contained in a frame. The frame is specified in clause 8.1.4.

Each command consists of the following fields:

- Preamble
- Delimiter
- Command code
- Parameter fields, depending on the command
- Application data fields, depending on the command
- CRC

Each response consists of the following fields:

- Return Preamble
- Application data fields
- CRC

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.

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

The MSB shall be the byte at the specified address. The LSB 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 MSB 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

The command consist of the following fields:

- Preamble
- Delimiter
- Command
- Parameter and data files
- CRC

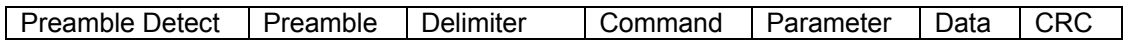


Figure 71 - General command format

8.1.4.1 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 kbps.

8.1.4.2 PREAMBLE

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

010101010101010101

8.1.4.3 Delimiters

Four delimiters are defined.

8.1.4.3.1 Start delimiter 1

In NRZ format; includes Manchester errors; spaces ignored

11 00 11 10 10 - Delimiter 1

8.1.4.3.2 Start delimiter 2

In NRZ format; includes Manchester errors; spaces ignored

01 01 11 00 11 - Delimiter 2

Reserved for future use

8.1.4.3.3 Start delimiter 3

In NRZ format; includes Manchester errors; spaces ignored

00 11 10 01 01 - Delimiter 3

Reserved for future use

8.1.4.3.4 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 4.1.

8.1.4.4 CRC

See clauses 6.4.6.2, 6.4.6.3 and Annex A.

8.1.5 Response format

The response consists of the following fields:

- Quiet
- Return Preamble
- Data fields
- CRC

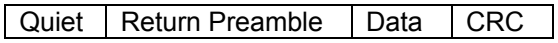


Figure 72 — General response format

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.

8.1.5.1 QUIET

The tag shall not backscatter for 2 bytes (400 μs at 40 kbps). The duration of the quiet time is determined by the communication speed of the return link.

8.1.5.2 CRC

See clauses 6.4.6.2, 6.4.6.3 and Annex A.

8.1.6 WAIT

During the WAIT field, the interrogator provides steady power to the tag for duration of at least 15ms. No on-off key data may be sent during the write operation.

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 8.2.7.8.9. 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.

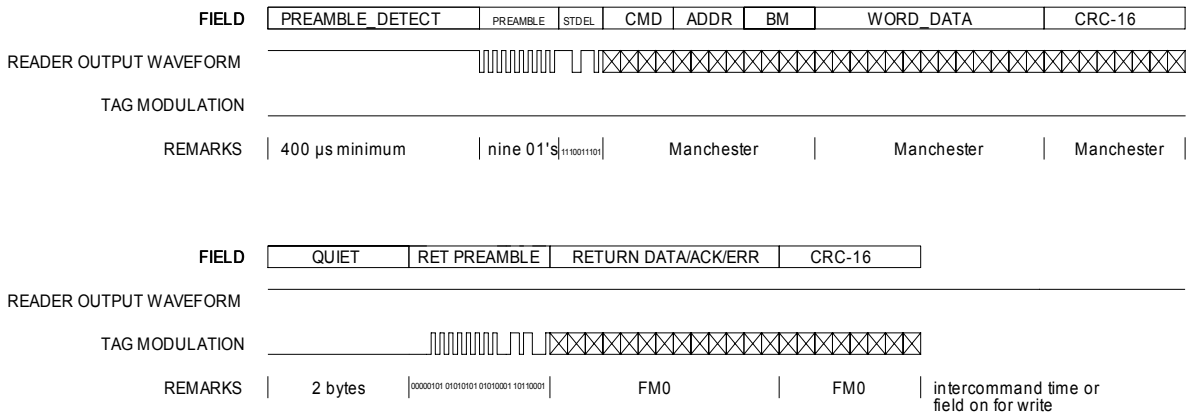


Figure 73 — Sample Command/Response Packets for (GROUP_SELECT (40 kbps on forward and return link)

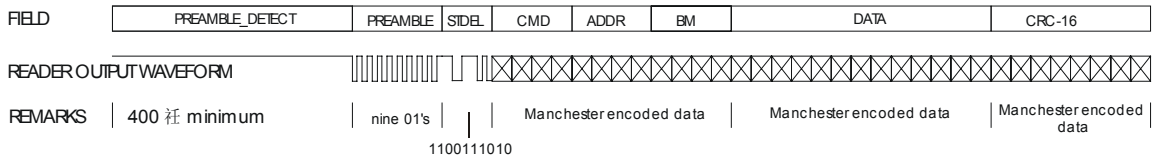


Figure 74 — Sample Command/Response Packets for WRITE (40 kbps on forward and return link)

8.1.7 Communication sequences at packet level

Figure 75, Figure 76 and Figure 77 show several examples of communication sequences at the packet level. Figure 75 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 misscalibrate as a result of the spurious signals that may be generated.

Figure 76 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.

Figure 77 depicts a packet sequence in which a frequency hop occurs between the interrogator command and the tag's response to that command. A tag resync signal is again required after the hop because spurious signals may be generated during the hop time. Although, this does not matter to any tags that are responding, any tags that don't respond may miscalibrate as a result of the spurious signals that may be generated during the hop time.

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

Figure 75 — 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 76 — Command sequence with a hop between response and next command

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

Figure 77 — Command sequence with a hop between command and response to that 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.

8.2.1.2 CRC

See Annex A.

8.2.1.3 FLAGS

The tag shall support a field of 8 flags. This field is called FLAGS.

Table 46 — 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)

8.2.1.3.1 Data Exchange Status Bit (DE_SB)

The tag shall set this bit when the tag goes into the DATA_EXCHANGE state and keep 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.

8.2.1.3.2 WRITE_OK

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

8.2.1.3.3 BATTERY_POWERED

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

8.2.1.3.4 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.

8.2.2 Tag memory organization

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.

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:

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.

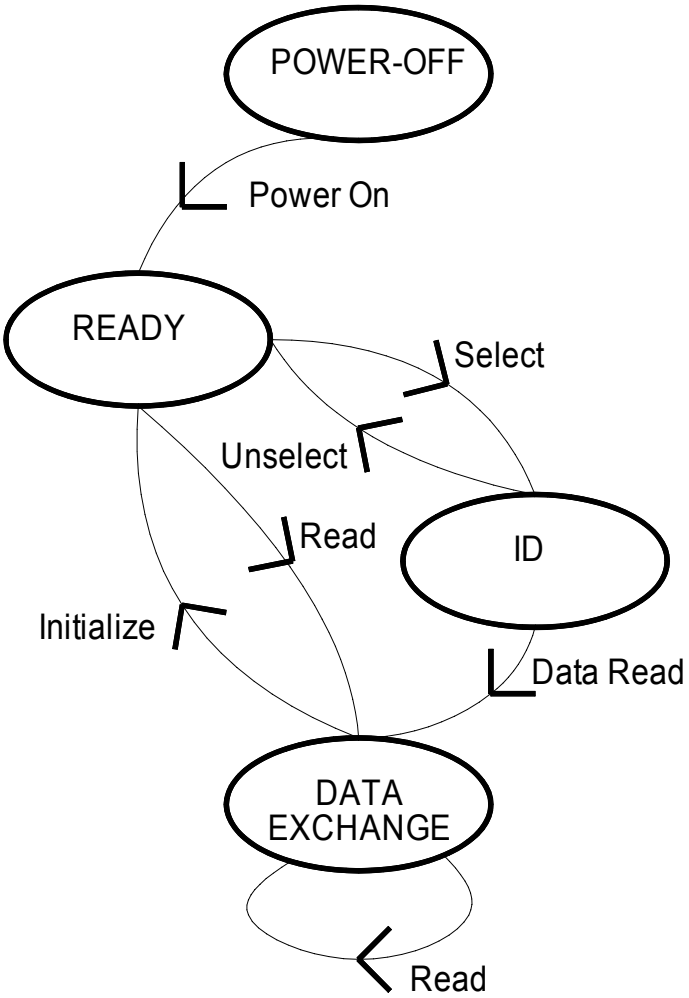


Figure 78 — State Diagram

NOTE This diagram does not show that the tag goes into POWER-OFF from all states in the case that the interrogator field is permanently turned-off.

The state diagram only shows an overview of the possible transition. Details are specified in Table 48.

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 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 48.

8.2.4.2 Detailed Command Processing

Commands shall be active in states marked with "X" and neither causes a state change, nor cause a response in the other states.

Table 47 — 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
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_SPECIAL	X	X	X
READ_UNADDRESSED	X	X	X
RESEND_O		X	

Table 48 — 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	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_SPECIAL	ID no match	READY
READY	READ_SPECIAL	ID match	DATA_EXCHANGE
READY	READ_UNADDRESSED		DATA_EXCHANGE
ID	GROUP_UNSELECT_EQ	≠	ID
ID	GROUP_UNSELECT_NE	=	ID
ID	GROUP_UNSELECT_GT	≤	ID
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 notWRITE_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	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
ID	RESEND_O		ID
ID	READ_FLAGS	ID no match	ID
ID	READ_FLAGS	ID match	DATA_EXCHANGE
ID	READ_16BYTE	ID no match	ID
ID	READ_16BYTE	ID match	DATA_EXCHANGE
ID	READ_SPECIAL	ID no match	ID
ID	READ_SPECIAL	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	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		ID
DATA_EXCHANGE	READ_FLAGS		DATA_EXCHANGE
DATA_EXCHANGE	READ_VARIABLE		DATA_EXCHANGE
DATA_EXCHANGE	READ_SPECIAL		DATA_EXCHANGE

8.2.5 Collision arbitration

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
- A random 1 or 0 generator.

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 to increment COUNT. That is, they move further away from being able to transmit.

   All tags receiving FAIL 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)

8.2.5.1 Special Collision Arbitration

In the case that parts of the user data is 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
- Multiple commands

Further, commands have one of the following types:

- Mandatory
- Optional
- Custom
- Proprietary

8.2.7 Command types

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

8.2.7.1 Mandatory

The command codes range from '00' to '0F', '11' to '13', '15' and '1D' to '3F'.

All tags shall support them.

8.2.7.2 Optional

The command codes range from '17 to '1C and from '40' to '9F'.

Tags may support them, at their option. If supported, command and response formats shall comply with the definition given in this standard.

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

NOTE The command whose code ranges from '17' to '1C' 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 49.

8.2.7.3 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 customised 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.

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

8.2.7.4 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 standard. 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.

8.2.7.5 Command codes and format

Table 49 — 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
'1C'	Recommended	WRITE4BYTE_MULTIPLE	BYTE_MASK	ADDRESS	4BYTE_DATA
'1D'-'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
'48' – '4F'	Optional	RFU			
'50'	Optional	READ_FLAGS	ID	ADDRESS	none
'51'	Optional	READ_VARIABLE	ID	ADDRESS	none
'52'	Optional	READ_SPECIAL	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			

8.2.7.5.1 Command Fields

Table 50 — 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
ACKNOWLEDGE	1 byte
ACKNOWLEDGE_OK	1 byte
ACKNOWLEDGE_NOK	1 byte
ERROR	1 byte
ERROR_OK	1 byte
ERROR_NOK	1 byte

8.2.7.5.2 Tag responses

Table 51 — 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	16 bytes
n/a	BYTE_DATA	1byte
‘02’ – ‘FD’	RFU	

8.2.7.6 Selection commands

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.

8.2.7.6.1 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, GROUP_UNSELECT_LT has 3 arguments (parameter and data):

ADDRESS

BYTE_MASK

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

LT M LOWER THAN D

The arguments of the comparison are

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 argument of the command

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.

Table 52 — 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

8.2.7.6.2 Data comparison for selection command on flags

Each select command of the commands GROUP_SELECT_EQ_FLAGS GROUP_SELECT_NE_FLAGS, GROUP_UNSELECT_EQ_FLAGS, GROUP_UNSELECT_NE_FLAGS, has 2 arguments (parameter and data):

BYTE_MASK

BYTE_DATA

and the tag shall make of 2 possible comparisons:

EQ FLAGS EQUAL D

NE FLAGS NOT EQUAL D

The arguments of the comparison are FLAGS, as defined in chapter XXX 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.

Table 53 — Data masking for Group_Select_Flags and Group_Unselect_Flags

BYTE_MASK	WORD_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

8.2.7.6.3 GROUP_SELECT_EQ

Command code = ‘00’

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

On receiving a GROUP_SELECT_EQ command, a tag which is 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 79 — GROUP_SELECT_EQ command

Preamble	ID	CRC
	64 bits	16 bits

Figure 80 — GROUP_SELECT_EQ response in the case of NO error

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

8.2.7.6.4 GROUP_SELECT_NE

Command code = '01'

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

On receiving a GROUP_SELECT_NE command, a tag which 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 81 — GROUP_SELECT_NE command

Preamble	ID	CRC
	64 bits	16 bits

Figure 82— GROUP_SELECT_NE response in the case of NO error

8.2.7.6.5 GROUP_SELECT_GT

Command code = '02'

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

On receiving a GROUP_SELECT_GT command, a tag which 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 83 — GROUP_SELECT_GT command

Preamble	ID	CRC
	64 bits	16 bits

Figure 84 — GROUP_SELECT_GT response in the case of NO error

8.2.7.6.6 GROUP_SELECT_LT

Command code = '03'

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

On receiving a GROUP_SELECT_LT command, a tag which 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 85 — GROUP_SELECT_LT command

Preamble	ID	CRC
	64 bits	16 bits

Figure 86 — GROUP_SELECT_LT response in the case of NO error

8.2.7.6.7 GROUP_UNSELECT_EQ

Command code = '04'

On receiving a GROUP_UNSELECT_EQ command, a tag which is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the DATA sent by the interrogator. In the case that the memory content is equal to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 87 — GROUP_UNSELECT_EQ command

Preamble	ID	CRC
	64 bits	16 bits

Figure 88 — GROUP_UNSELECT_EQ response in the case of NO error and comparison fails

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

8.2.7.6.8 GROUP_UNSELECT_NE

Command code = ‘05’

On receiving a GROUP_UNSELECT_NE command, a tag which is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the DATA sent by the interrogator. In the case that the memory content is not equal to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 89 — GROUP_UNSELECT_NE command

Preamble	ID	CRC
	64 bits	16 bits

Figure 90 — GROUP_UNSELECT_NE response in the case of NO error and comparison fails

8.2.7.6.9 GROUP_UNSELECT_GT

Command code = ‘06’

On receiving a GROUP_UNSELECT_GT command, a tag which is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the DATA sent by the interrogator. In the case that the memory content is greater than to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 91 — GROUP_UNSELECT_GT command

Preamble	ID	CRC
	64 bits	16 bits

Figure 92 — GROUP_UNSELECT_GT response in the case of NO error and comparison fails

8.2.7.6.10 GROUP_UNSELECT_LT

Command code = ‘07’

On receiving a GROUP_UNSELECT_LT command, a tag which is in the ID state shall read the 8-byte memory content beginning at the specified address and compare it with the DATA sent by the interrogator. In the case that the memory content is lower than to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	64 bits	16 bits

Figure 93 — GROUP_UNSELECT_LT command

Preamble	ID	CRC
	64 bits	16 bits

Figure 94 — GROUP_UNSELECT_LT response in the case of NO error and comparison fails

8.2.7.6.11 MULTIPLE_UNSELECT

Command code = '13'

On receiving a MULTIPLE_UNSELECT command, a tag which is in the ID state shall read the 1-byte memory content beginning at the specified address and compare it with the DATA sent by the interrogator. In the case that the memory content is equal to 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.

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

Preamble	Delimiter	Command	ADDRESS	DATA	CRC
		8 bits	8 bits	8 bits	16 bits

Figure 95 — MULTIPLE_UNSELECT command

Preamble	ID	CRC
	64 bits	16 bits

Figure 96 — MULTIPLE_UNSELECT response in the case that WRITE_OK is reset or DATA is not equal to memory content at ADDRESS

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.

8.2.7.6.12 GROUP_SELECT_EQ_FLAGS

Command code = '17'

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

On receiving a GROUP_SELECT_EQ_FLAGS command, a tag which 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	8 bits	16 bits

Figure 97 — GROUP_SELECT_EQ_FLAGS command

Preamble	ID	CRC
	64 bits	16 bits

Figure 98 — GROUP_SELECT_EQ_FLAGS response in the case of NO error

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

8.2.7.6.13 GROUP_SELECT_NE_FLAGS

Command code = '18'

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

On receiving a GROUP_SELECT_NE_FLAGS command, a tag which 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.

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	8 bits	16 bits

Figure 99 — GROUP_SELECT_NE_FLAGS command

Preamble	ID	CRC
	64 bits	16 bits

Figure 100 — GROUP_SELECT_NE_FLAGS response in the case of NO error

8.2.7.6.14 GROUP_UNSELECT_EQ_FLAGS

Command code = '19'

On receiving a GROUP_UNSELECT_EQ_FLAGS command, a tag which is in the ID state shall compare the FLAGS with the DATA sent by the interrogator. In the case that the FLAGS are equal to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	8 bits	16 bits

Figure 101 — GROUP_UNSELECT_EQ_FLAGS command

Preamble	ID	CRC
	64 bits	16 bits

Figure 102 — GROUP_UNSELECT_EQ_FLAGS response in the case of NO error and comparison fails

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

8.2.7.6.15 GROUP_UNSELECT_NE_FLAGS

Command code = ‘1A’

On receiving a GROUP_UNSELECT_NE_FLAGS command, a tag which is in the ID state shall compare the FLAGS with the DATA sent by the interrogator. In the case that the FLAGS are not equal to 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.

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

Preamble	Delimiter	Command	Mask	DATA	CRC
		8 bits	8 bits	8 bits	16 bits

Figure 103 —GROUP_UNSELECT_NE_FLAGS command

Preamble	ID	CRC
	64 bits	16 bits

Figure 104 — GROUP_UNSELECT_NE_FLAGS response in the case of NO error and comparison fails

8.2.7.7 Identification commands

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

8.2.7.7.1.1 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 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 resulting COUNT value is 0, then the tag shall read its UID and sent back it in the response.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 105 — FAIL command

Preamble	ID	CRC
	64 bits	16 bits

Figure 106 — FAIL response in the case that COUNT stays zero

8.2.7.7.1.2 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.
- 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 if it is in the ID state. In the case that its internal counter COUNT is not zero it 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.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 107 — SUCCESS command

Preamble	ID	CRC
	64 bits	16 bits

Figure 108 — SUCCESS response in the case that COUNT is zero

8.2.7.7.1.3 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 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.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 109 — RESEND command

Preamble	ID	CRC
	64 bits	16 bits

Figure 110 — RESEND response in the case that COUNT is zero

8.2.7.7.2 INITIALIZE

Command code = ‘0A’

On receiving a INITIALIZE command a tag shall go into the READY state and reset the Data_Exchange_Status_Bit.

It shall not send any response.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 111 —INITIALIZE command

8.2.7.7.2.1 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 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 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.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 112 — FAIL_O command

Preamble	WORD_DATA	CRC
	32 or 64 bits	16 bits

Figure 113 — FAIL_O response in the case that COUNT stays zero

8.2.7.7.2.2 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.
- 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 if it is in the ID state. In the case that its 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.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 114 — SUCCESS_O command

Preamble	WORD_DATA	CRC
	64 bits	16 bits

Figure 115 — SUCCESS_O response in the case that COUNT is zero

8.2.7.7.2.3 RESEND_0

Command code = ’46’ or ‘47’

The identification algorithm uses RESEND 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 command if it is in the ID state. If the COUNT value is 0, then the tag shall read the memory at address ‘14’ sent back either 32 bit (for command code ‘46’) or 64 bit (for command code ‘47’) in the response.

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 116 — RESEND_O command

Preamble	ID	CRC
	64 bits	16 bits

Figure 117 — RESEND_O response in the case that COUNT is zero

8.2.7.8 Data Transfer commands

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

8.2.7.8.1 READ

Command code = ‘0C’

On receiving the READ command, 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. 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 form ‘00’ to ‘FF’ (0 to 255).

Preamble	Delimiter	Command	UID	Address	CRC
		8 bits	64 bits	8 bits	16 bits

Figure 118 — Read command

Preamble	WORD_DATA	CRC
	64 bits	16 bits

Figure 119 — Read response in the case of NO error

8.2.7.8.2 DATA_READ

Command code = ‘0B’

On receiving the DATA_READ command, 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. 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 form ‘00’ to ‘FF’ (0 to 255).

Preamble	Delimiter	Command	UID	Address	CRC
		8 bits	64 bits	8 bits	16 bits

Figure 120 — DATA_READ command

Preamble	WORD_DATA	CRC
	64 bits	16 bits

Figure 121 — DATA_READ response in the case of NO error

8.2.7.8.3 DATA_READ_O

Command code = ‘44’, or ‘45’

On receiving the DATA_READ_O command, the tag shall only if it is in either the state ID or the state DATA_EXCHANGE compare the sent DATA with its memory content starting at address ‘10’. In the case that the used command code is ‘44’ 32 bits shall be compared. In the case that the used command code is ‘45’ 64 bits shall be compared. In the case that the two data streams are equal, then 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. 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 form ‘00’ to ‘FF’ (0 to 255).

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

Figure 122 — DATA_READ_O command

Preamble	WORD_DATA	CRC
	64 bits	16 bits

Figure 123 — DATA_READ_O response in the case of NO error

8.2.7.8.4 READ_FLAGS

Command code = ‘50’

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

Preamble	Delimiter	Command	CRC
		8 bits	16 bits

Figure 124 — READ_FLAGS command

Preamble	BYTE_DATA	CRC
	8 bits	16 bits

Figure 125 — READ_FLAGS in the case of NO error

8.2.7.8.5 READ_VARIABLE

Command code = '51'

On receiving the READ_VARIABLE command, 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. 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 form '00' to 'FF' (0 to 255). Length is numbered from '00' to 'FF'.

Preamble	Delimiter	Command	UID	Address	Length	CRC
		8 bits	64 bits	8 bits	8 bits	16 bits

Figure 126 — READ_VARIABLE command

Preamble	Length * BYTE_DATA	CRC
	Length * 8 bits	16 bits

Figure 127 — READ_VARIABLE response in the case of NO error

8.2.7.8.6 READ_SPECIAL

Command code = '52'

On receiving the READ_SPECIAL command, 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 FLAGS combination of FLAGS and the specified length of the memory content beginning at the specified address and send back the FLAGS and the memory content in the response. 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 form '00' to 'FF' (0 to 255). Length is numbered from '00' to 'FF'

Preamble	Delimiter	Command	UID	Address	Length	CRC
		8 bits	64 bits	8 bits	8 bits	16 bits

Figure 128 — READ_SPECIAL command

Preamble	Length * BYTE_DATA	CRC
	Length * 8 bits	16 bits

Figure 129 — READ_SPECIAL response in the case of NO error

This command should fulfil special customer requirements that need both flag and memory information in one command.

8.2.7.8.7 READ_UNADDRESSED

Command code = ‘53’

On receiving the READ_UNADDRESSED command, 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.

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

Preamble	Delimiter	Command	Address	CRC
		8 bits	8 bits	16 bits

Figure 130 — READ_UNADDRESSED command

Preamble	WORD_DATA	WORD_DATA	CRC
	64 bits	64 bits	16 bits

Figure 131 — READ_UNADDRESSED response in the case of NO error

8.2.7.8.8 READ_VERIFY

Command code = ‘12’

On receiving the READ_VERIFY command, 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. 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 form ‘00’ to ‘FF’ (0 to 255).

Preamble	Delimiter	Command	UID	Address	CRC
		8 bits	64 bits	8 bits	16 bits

Figure 132 — READ_VERIFY command

Preamble	BYTE_DATA	CRC
	8 bits	16 bits

Figure 133 — READ_VERIFY response in the case of NO error

8.2.7.8.9 WRITE

Command code = '0D'

On receiving the WRITE command, 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. In the case that the memory is unlocked, it shall send back the ACKNOWLEDGE and program the data into the specified memory address.

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 form '00' to 'FF' (0 to 255).

Preamble	Delimiter	Command	UID	Address	Data	CRC
		8 bits	64 bits	8 bits	8 bits	16 bits

Figure 134 — Write command

Preamble	ACKNOWLEDGE	CRC
	8 bits	16 bits

Figure 135 — WRITE response in the case of NO error

Preamble	ERROR	CRC
	8 bits	16 bits

Figure 136 — WRITE response in the case of locked memory

8.2.7.8.10 WRITE4BYTE

Command code = '1B'

On receiving the WRITE4BYTE command, 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 byte of the memory is locked, it shall send back the ERROR response. In the case that all bytes are unlocked, it shall send back the ACKNOWLEDGE and 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.

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

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

Figure 137 — WRITE4BYTE command

Preamble	ACKNOWLEDGE	CRC
	8 bits	16 bits

Figure 138 — WRITE4BYTE response in the case of NO error

Preamble	ERROR	CRC
	8 bits	16 bits

Figure 139 — WRITE4BYTE response in the case that of locked memory

8.2.7.8.11 LOCK

Command code = ‘0F’

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

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).

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

Figure 140 — LOCK command

Preamble	ACKNOWLEDGE	CRC
	8 bits	16 bits

Figure 141 — LOCK response in the case that locking was possible

Preamble	ERROR	CRC
	8 bits	16 bits

Figure 142 — LOCK response in the case that locking was not possible

8.2.7.8.12 QUERY_LOCK

Command code = ‘11’

On receiving a QUERY_LOCK command, a tag shall read its UID and compare it with the DATA sent by the interrogator. In the case that the UID is equal to DATA, the ADDRESS is within the valid address range, then the tag shall move into the DATA_EXCHANGE state. Further, the tag shall read the lock bit for the memory byte at ADDRESS. In the case that this memory is not locked, then it shall response ACKNOWLEDGE_OK if WRITE_OK is set and ACKNOWLEDGE_NOK if WRITE_OK is cleared. In the case that this memory is locked, then it shall response ERROR_OK if WRITE_OK is set and ERROR_NOK 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).

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

Figure 143 — QUERY_LOCK command

Preamble	ACKNOWLEDGE_OK	CRC
	8 bits	16 bits

Figure 144 — QUERY_LOCK response in memory address is not locked and WRITE_OK is set

Preamble	ACKNOWLEDGE_OK	CRC
	8 bits	16 bits

Figure 145 — QUERY_LOCK response in memory address is not locked and WRITE_OK is cleared

Preamble	ERROR_OK	CRC
	8 bits	16 bits

Figure 146 — QUERY_LOCK response in memory address is locked and WRITE_OK is set

Preamble	ERROR_OK	CRC
	8 bits	16 bits

Figure 147 — QUERY_LOCK response in memory address is locked and WRITE_OK is cleared

8.2.7.8.13 WRITE_MULTIPLE

Command code = ‘0E’

Write Multiple commands are used to write to and to verify multiple tags in parallel.

On receiving the WRITE_MULTIPLE command, a tag which 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.

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

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

Figure 148 — WRITE_MULTIPLE command

8.2.7.8.14 WRITE4BYTE_MULTIPLE

Command code = ‘1C’

Write Multiple commands are used to write to and to verify multiple tags in parallel.

On receiving the WRITE4BYTE_MULTIPLE command, a tag which 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 byte 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.

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

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

Figure 149 — WRITE4BYTE_MULTIPLE command

8.2.7.9 Response Description (Binary Tree Protocol Type)

8.2.7.9.1 ACKNOWLEDGE

ACKNOWLEDGE indicates a successful WRITE or LOCK.

8.2.7.9.2 ERROR

ERROR indicates an error in the WRITE.

8.2.7.9.3 ACKNOWLEDGE_OK

ACKNOWLEDGE_OK is the unlocked and successful write response to a QUERY_LOCK.

8.2.7.9.4 ACKNOWLEDGE_NOK

ACKNOWLEDGE_NOK is the unlocked and unsuccessful write response to a QUERY_LOCK.

8.2.7.9.5 ERROR_OK

ERROR_OK is the locked and successful write response to a QUERY_LOCK.

8.2.7.9.6 ERROR_NOK

ERROR_NOK is the locked and unsuccessful write response to a QUERY_LOCK.

8.2.7.9.7 WORD_DATA

WORD_DATA is 8 bytes returned in response to a GROUP_SELECT, GROUP_UNSELECT, FAIL, SUCCESS, RESEND, READ, or DATA_READ command.

8.2.7.9.8 BYTE_DATA

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

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. Further, if the tag receives a valid start delimiter it returns to the READY state, so that it ignores a subsequent write multiple command.

Annex A
(informative)

Cyclic redundancy check (CRC)

A.1 Interrogator to tag CRC-5

The polynomial used to calculate the CRC is x^5+x^2+1. The 5-bit register must be preloaded with '12' (LSB to MSB 10010). In Figure A. 1, C4 to C0 are the CRC bits (LSB to MSB). The 11 bits of data must be clocked through the CRC register, using the MSB first.

The 5 CRC bits are sent, MSB first. After the LSB bit is clocked through, the 5-bit CRC register should contain all zero's.

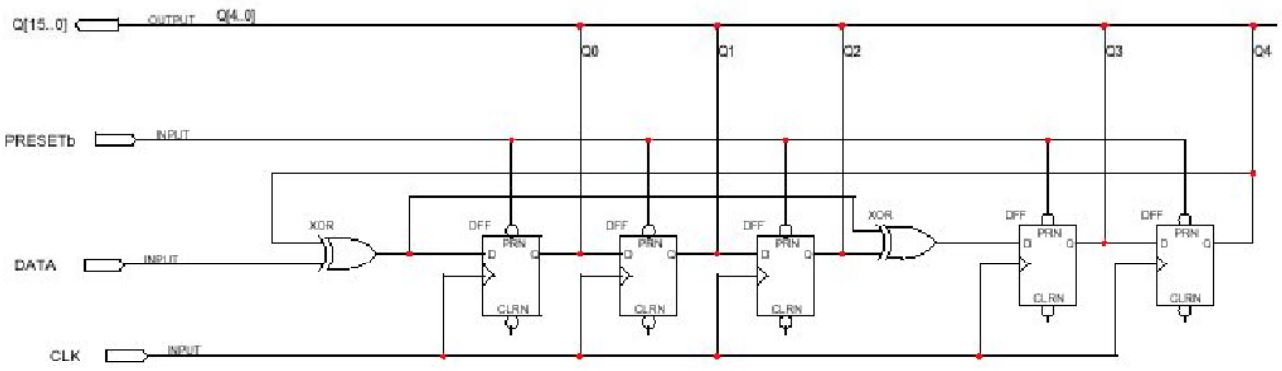


Figure A. 1 — CRC-5

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

The polynomial used to calculate the CRC-16 is $X^{16}+X^{12}+X^5+1$, which is the CRC-CCITT 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 is used from interrogator to tag and from tag to interrogator.

Table A. 1— CRC Definition

CRC Definition					
CRC type	**Length**	**Polynomial**	**Direction**	**Preset**	**Residue**
CRC-CCITT	16 bits	$X^{16} + X^{12} + X^5 + 1$	Forward	'FFFF'	'0'

The CRC algorithm is as follows:

For computing the CRC: (see example Table A.2)

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

For checking the CRC: (See example Table A.3)

- compute the CRC on the incoming packet
- invert the received CRC data bits
- accumulate the “inverted CRC bits” in the CRC registers
- 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 1D0F, as defined in CRC-CCITT. (See example Table A.4)

- compute the CRC on the incoming packet
- accumulate the 16 “received CRC” in the CRC registers
- 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);
}
```

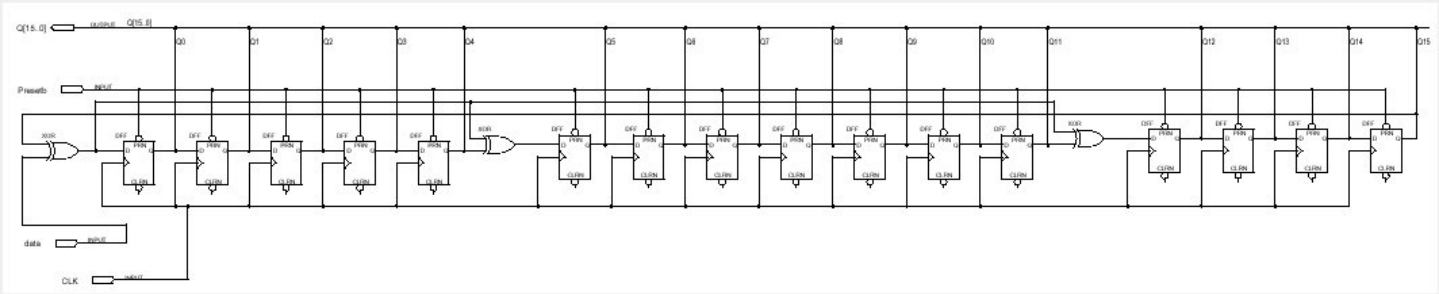


Figure A. 2— Shift Register Implementation of CRC-16 accumulator

A.2.1 CRC calculation examples

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

Command code. ‘09’

The packet 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. 2 – CRC Type B

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

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

The following example 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. 3 — Practical example of CRC calculation for a 'SUCCESS' command in the Interrogator

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

Table A.4 — Practical example of CRC checking for a ‘SUCCESS’ command in the tag

The CRC bits transmitted by the interrogator are inverted, ‘8F26’, MSB first.

The tag then inverts the received data bits and accumulates those 16 bits (i.e. ‘70D9’).

Step	Input (Sent CRC-16)	Calculated CRC in tag
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’

Table A.5 — 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 ('8F26') are used.

Step	Input (Sent CRC-16)	Calculated CRC in tag
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
(informative)

Memory mapping for Type B

B.1 Address Map

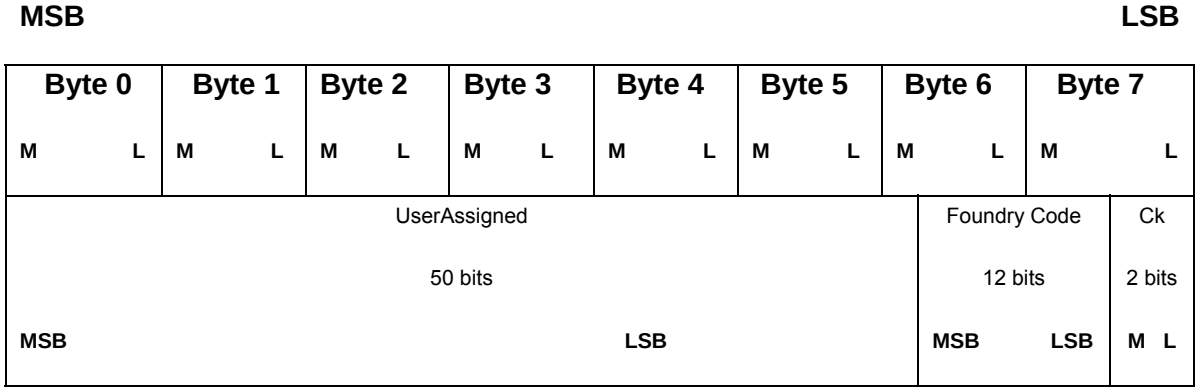
Memory storage capacity is variable and is supported via an eight bit address field used with memory-addressable commands (Read, Write, Lock, etc.). This field addresses memory according to block size. Thus, for a block size of 1 byte (1 octet) the maximum addressable memory is 256 bytes. For a block size of 4 byes, the maximum addressable memory is 1024 bytes. The tag block size is represented in the system information that is accessible through either reserved memory locations or through “special” system commands. The Binary Tree Protocol Type utilizes a reserved memory element within the tag memory map to store system information. As such, under this protocol option, memory locations 0 through 17 are reserved for system information. This is depicted below.

Table B.1 — Tag Memory Layout (Bytes 0-17)

Bytes	Field Name	Valid Range	Format
0 – 7	Unique ID (UID)	0000000000000000 - FFFFFFFFFFFFFFFF	Hexadecimal (see Figure XX)
8, 9	Tag Manufacturer		ASCII
10, 11	Hardware Tag System Information		
12	Embedded Application Code	00 – FF	Hexadecimal
13 – 17	Tag Memory Map Allocation	0000000000 – FFFFFFFFFF	Hexadecimal

B.1.1 Unique ID (bytes 0 – 7)

The tag UID contains three sub fields as defined below:



B.1.1.1 Chip Manufacturer Assigned (bits 14-63)

This is a 50-bit field that is defined and managed by the chip manufacturer. Different chip manufacturers will have different "foundry codes" (see below), thus eliminating the potential for duplicated collision arbitration data (Tag ID's). 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 250 -1.

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

B.1.1.2 Foundry Code (bits 2-13)

The foundry code consists of two sub-fields defined as follows:

Manufacturer Code (bits 6 – 13)

This 8-bit hexadecimal field is required to segregate multiple producers of chips compliant with this air interface standard. All manufacturers will have a separate and unique number allowing them to produce chips with collision arbitration numbers that do not interfere through duplication.

Registration and management of this code shall be in accordance with the specified mechanism defined by ISO/IEC JTC 1/SC 31.

Fab Code (bits 2 – 5)

This four bit hexadecimal code is available to provide further segregation within a registered Manufacturer Code to accommodate multiple chip fabs. It is the responsibility of the registered manufacturer to administer this code (if used) in conjunction with the Serial Number (bits 14 – 63) field to ensure that all tags produced by the manufacturer will have a unique and unambiguous number (used by the collision arbitration algorithm).

B.1.1.3 Check Sum (bits 0, 1)

This is a truncated 2-bit check sum. It represents the truncated sum of the bits set to 1 for the 62 bits preceding the check sum in the Tag Serial Number field. Valid values are 0, 1, 2, & 3.

B.1.2 Remaining system memory

B.1.2.1 Manufacturer ID (Bytes 8,9)

These two bytes have been reserved for encoding the Tag Manufacturer ID to provide some conformance to anticipated RFID standards. These fields will initially be encoded with the following codes depending on the manufacturer of the tags.

Table B. 2 — Manufacturer Codes

Manufacturer	ASCII Representation	Hexadecimal Code
Reserved	“AT”	4154
Reserved	“HT”	4854
Reserved	“AA”	4141
Reserved	“AS”	4153
Reserved	“AN”	414E

B.1.2.2 Tag Hardware Type (Bytes 10,11)

This is a 2-byte hexadecimal representation of the tag hardware design. This number shalll 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.1.2.3 Embedded Application Code (Byte 12)

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 as follows:

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 1)
0C – FE	To be allocated and registered by to application based needs.

NOTE 1 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.1.2.3.1 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.1.2.3.2 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.

The following table 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 with marketing. It is recommended that the manufacturer's Marketing department be contacted for obtaining a current listing and registering any additional codes.

Table B. 3 — 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
- Emergency Warning Systems

B.1.2.3.3 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.1.2.3.4 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.1.2.3.5 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.1.2.3.6 Embedded Application Codes “06 through 09”

Reserved for future use.

B.1.2.3.7 Embedded Applications Code “0A” – ISO 15962 Compliant Data Format

The Tag Memory Allocation for Embedded Applications Code “0C”, ISO 15962 Compliant Data Format, has been reserved to indicate a tag used in a fashion as defined by this standard. Under the current ISO 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.1.2.3.8 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 this application standard.

B.1.2.3.9 Embedded Application Codes “0C through 0E”

Reserved for future use.

B.1.2.3.10 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 this application standard. Under this definition byte 13 (Tag Memory Allocation Map) is allocated to retain the Application Family Identifier (AFI) information

B.1.2.3.11 Embedded Application Codes “10 through FF”

Reserved for future use.

B.1.2.3.12 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.1.2.3.13 Tag Application Memory (Bytes 18 and above)

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

Annex C
(informative)

Tag sensitivity

Table C.1 — Tag sensitivity

Sensitivity class	Lower sensitivity limit V/m	Upper sensitivity limit V/m	Field strength required for write operation
S1	10	5	15
S2	4	2.5	6
S3	1.5	N/A	2