Specification for RFID Air Interface

EPC™ Radio-Frequency Identity Protocols
Class-1 Generation-2 UHF RFID
Protocol for Communications at 860 MHz – 960 MHz
Version 1.0.9

Contents

Index of Figures

Index of Tables

Foreword

This specification defines the base class (Class-1) of four radio-frequency identification (RFID) Tag classes. The class structure is described as follows:

Class-1: Identity Tags (normative)

Passive-backscatter Tags with the following minimum features:

- An electronic product code (EPC) identifier,
- A Tag identifier (TID),
- A 'kill' function that permanently disables the Tag,
- Optional password-protected access control, and
- Optional user memory.

Class restrictions (normative)

Class-2, Class-3, Class-4, or higher class Tags shall not conflict with the operation of, nor degrade the performance of, Class-1 Tags located in the same RF environment.

Higher-class Tags (informative)

The following class descriptions provide an example of how higher-class Tag features might be delineated:

Class-2: Higher-Functionality Tags

Passive Tags with the following anticipated features above and beyond those of Class-1 Tags:

- An extended TID,
- Extended user memory,
- Authenticated access control, and
- Additional features (TBD) as will be defined in the Class-2 specification.

Class-3: Semi-Passive Tags

Semi-passive Tags with the following anticipated features above and beyond those of Class-2 Tags:

- An integral power source, and
- Integrated sensing circuitry.

Class-4: Active Tags

Active Tags with the following anticipated features above and beyond those of Class-3 Tags:

- Tag-to-Tag communications,
- Active communications, and
- Ad-hoc networking capabilities.

Introduction

This specification defines the physical and logical requirements for a passive-backscatter, Interrogator-talks-first (ITF), radio-frequency identification (RFID) system operating in the 860 MHz – 960 MHz frequency range. The system comprises Interrogators, also known as Readers, and Tags, also known as Labels.

An Interrogator transmits information to a Tag by modulating an RF signal in the 860 MHz – 960 MHz frequency range. The Tag receives both information and operating energy from this RF signal. Tags are passive, meaning that they receive all of their operating energy from the Interrogator's RF waveform.

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

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

1. Scope

This document specifies:

- Physical interactions (the signaling layer of the communication link) between Interrogators and Tags,
- Interrogator and Tag operating procedures and commands, and
- The collision arbitration scheme used to identify a specific Tag in a multiple-Tag environment.

2. Conformance

2.1 Claiming conformance

A device shall not claim conformance with this specification unless the device complies with

a) all clauses in this specification (except those marked as optional), and

b) the conformance document associated with this specification, and,

c) all local radio regulations.

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

2.2 General conformance requirements

2.2.1 Interrogators

To conform to this specification, an Interrogator shall:

- Meet the requirements of this specification,
- Implement the mandatory commands defined in this specification,
- Modulate/transmit and receive/demodulate a sufficient set of the electrical signals defined in the signaling layer of this specification to communicate with conformant Tags, and
- Conform to all local radio regulations.

To conform to this specification, an Interrogator may:

- Implement any subset of the optional commands defined in this specification, and
- Implement any proprietary and/or custom commands in conformance with this specification.

To conform to this specification, an Interrogator shall not:

- Implement any command that conflicts with this specification, or
- Require using an optional, proprietary, or custom command to meet the requirements of this specification.

2.2.2 Tags

To conform to this specification, a Tag shall:

- Meet the requirements of this specification,
- Implement the mandatory commands defined in this specification,
- Modulate a backscatter signal only after receiving the requisite command from an Interrogator, and
- Conform to all local radio regulations.

To conform to this specification, a Tag may:

- Implement any subset of the optional commands defined in this specification, and
- Implement any proprietary and/or custom commands as defined in 2.3.3 and 2.3.4, respectively.

To conform to this specification, a Tag shall not:

- Implement any command that conflicts with this specification,
- Require using an optional, proprietary, or custom command to meet the requirements of this specification, or

- Modulate a backscatter signal unless commanded to do so by an Interrogator using the signaling layer defined in this specification.

2.3 Command structure and extensibility

Subclause 6.3.2.10 defines the structure of the command codes used by Interrogators and Tags, as well as the availability of future extensions. Each command is defined and labeled as mandatory or optional.

2.3.1 Mandatory commands

Conforming Tags and Interrogators shall support all mandatory commands.

2.3.2 Optional commands

Conforming Interrogators may or may not support optional commands. Conforming Tags may or may not support optional commands. If an Interrogator or a Tag implements an optional command, it shall implement it in the manner specified.

2.3.3 Proprietary commands

Proprietary commands may be enabled in conformance with this specification, but are not specified herein. All proprietary commands shall be capable of being permanently disabled. Proprietary commands are intended for manufacturing purposes and shall not be used in field-deployed RFID systems.

2.3.4 Custom commands

Custom commands may be enabled in conformance with this specification, but are not specified herein. An Interrogator shall issue a custom command only after singulating a Tag and reading (or having prior knowledge of) the Tag manufacturer's identification in the Tag's TID memory. An Interrogator shall use a custom command only in accordance with the specifications of the Tag manufacturer identified in the TID.

3. Normative references

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

EPCglobal™: *EPC™ Tag Data Standards*

EPCglobal™ (2004): *FMCG RFID Physical Requirements Document (draft)*

EPCglobal™ (2004): *Class-1 Generation-2 UHF RFID Implementation Reference (draft)*

European Telecommunications Standards Institute (ETSI), EN 302 208: *Electromagnetic compatibility and radio spectrum matters (ERM) – Radio-frequency identification equipment operating in the band 865 MHz to 868 MHz with power levels up to 2 W, Part 1 – Technical characteristics and test methods*

European Telecommunications Standards Institute (ETSI), EN 302 208: *Electromagnetic compatibility and radio spectrum matters (ERM) – Radio-frequency identification equipment operating in the band 865 MHz to 868 MHz with power levels up to 2 W, Part 2 – Harmonized EN under article 3.2 of the R&TTE directive*

ISO/IEC Directives, Part 2: *Rules for the structure and drafting of International Standards*

ISO/IEC 3309: *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure*

ISO/IEC 15961: *Information technology, Automatic identification and data capture – Radio frequency identification (RFID) for item management – Data protocol: application interface*

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

ISO/IEC 15963: *Information technology — Radiofrequency identification for item management — Unique identification for RF tags*

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

ISO/IEC 18000-6: *Information technology automatic identification and data capture techniques — Radio frequency identification for item management air interface — Part 6: Parameters for air interface communications at 860–960 MHz*

ISO/IEC 19762: *Information technology AIDC techniques – Harmonized vocabulary – Part 3: radio-frequency identification (RFID)*

U.S. Code of Federal Regulations (CFR), Title 47, Chapter I, Part 15: *Radio-frequency devices, U.S. Federal Communications Commission*

4. Terms and definitions

The principal terms and definitions used in this specification are described in ISO/IEC 19762.

4.1 Additional terms and definitions

Terms and definitions specific to this document that supersede any normative references are as follows:

- **Air interface**

  The complete communication link between an Interrogator and a Tag including the physical layer, collision arbitration algorithm, command and response structure, and data-coding methodology.

- **Command set**

  The set of commands used to explore and modify a Tag population.

- **Continuous wave**

  Typically a sinusoid at a given frequency, but more generally any Interrogator waveform suitable for powering a passive Tag without amplitude and/or phase modulation of sufficient magnitude to be interpreted by a Tag as transmitted data.

- **Cover-coding**

  A method by which an Interrogator obscures information that it is transmitting to a Tag. To cover-code data or a password, an Interrogator first requests a random number from the Tag. The Interrogator then performs a bit-wise EXOR of the data or password with this random number, and transmits the cover-coded (also called ciphertext) string to the Tag. The Tag uncovers the data or password by performing a bit-wise EXOR of the received cover-coded string with the original random number.

- **Dense-Interrogator environment**

  An operating environment (defined below) within which the number of simultaneously active Interrogators is large relative to the number of available channels (for example, 50 active Interrogators operating in 50 available channels).

- **Extended temperature range**

  –40 °C to +65 °C (see nominal temperature range).

- **Full-duplex communications**

  A communications channel that carries data in both directions at once. See also half-duplex communications.

- **Half-duplex communications**

  A communications channel that carries data in one direction at a time rather than in both directions at once. See also full-duplex communications.

- **Inventoried flag**

  A flag that indicates whether a Tag may respond to an Interrogator. Tags maintain a separate **inventoried** flag for each of four sessions; each flag has symmetric *A* and *B* values. Within any given session, Interrogators typically inventory Tags from *A* to *B* followed by a re-inventory of Tags from *B* back to *A* (or vice versa).

- **Inventory round**

  The period between successive *Query* commands.

- **Multiple-Interrogator environment**

  An operating environment (defined below) within which the number of simultaneously active Interrogators is modest relative to the number of available channels (for example, 10 active Interrogators operating in 50 available channels).

- **Nominal temperature range**

  –25 °C to +40 °C (see extended temperature range).

- **Operating environment**

  A region within which an Interrogator's RF transmissions are attenuated by less than 90dB. In free space, the operating environment is a sphere whose radius is approximately 1000m, with the Interrogator located at the

center. In a building or other enclosure, the size and shape of the operating environment depends on factors such as the material properties and shape of the building, and may be less than 1000m in certain directions and greater than 1000m in other directions.

- **Operating procedure**
  Collectively, the set of functions and commands used by an Interrogator to identify and modify Tags. (Also known as the *Tag-identification layer*.)
- **Passive Tag (or passive Label)**
  A Tag (or Label) whose transceiver is powered by the RF field.
- **Permalock or Permalocked**
  A memory location whose lock status is unchangeable (i.e. the memory location is permanently locked or permanently unlocked) is said to be permalocked.
- **Persistent memory or persistent flag**
  A memory or flag value whose state is maintained during a brief loss of Tag power.
- **Physical layer**
  The data coding and modulation waveforms used in Interrogator-to-Tag and Tag-to-Interrogator signaling.
- **Protocol**
  Collectively, a physical layer and a Tag-identification layer specification.
- ***Q***
  A parameter that an Interrogator uses to regulate the probability of Tag response. An Interrogator commands Tags in an inventory round to load a *Q*-bit random (or pseudo-random) number into their slot counter; the Interrogator may also command Tags to decrement their slot counter. Tags reply when the value in their slot counter (i.e. their slot – see below) is zero. *Q* is an integer in the range (0,15); the corresponding Tag-response probabilities range from $2^0 = 1$ to $2^{-15} = 0.000031$.
- **Session**
  An inventory process comprising an Interrogator and an associated Tag population. An Interrogator chooses one of four sessions and inventories Tags within that session. The Interrogator and associated Tag population operate in one and only one session for the duration of an inventory round (defined above). For each session, Tags maintain a corresponding **inventoried** flag. Sessions allow Tags to keep track of their inventoried status separately for each of four possible time-interleaved inventory processes, using an independent **inventoried** flag for each process.
- **Single-Interrogator environment**
  An operating environment (defined above) within which there is a single active Interrogator at any given time.
- **Singulation**
  Identifying an individual Tag in a multiple-Tag environment.
- **Slot**
  Slot corresponds to the point in an inventory round at which a Tag may respond. Slot is the value output by a Tag's slot counter; Tags reply when their slot (i.e. the value in their slot counter) is zero. See also *Q* (above).
- **Slotted random anticollision**
  An anticollision algorithm where Tags load a random (or pseudo-random) number into a slot counter, decrement this slot counter based on Interrogator commands, and reply to the Interrogator when their slot counter reaches zero.
- **Tag-identification layer**
  Collectively, the set of functions and commands used by an Interrogator to identify and modify Tags (also known as the *operating procedure*).
- **Tari**
  Reference time interval for a data-0 in Interrogator-to-Tag signaling. The mnemonic "Tari" derives from the ISO/IEC 18000-6 (part A) specification, in which Tari is an abbreviation for Type A Reference Interval.

5. Symbols, abbreviated terms, and notation

The principal symbols and abbreviated terms used in this specification are detailed in ISO/IEC 19762, *Information technology AIDC techniques – vocabulary*. Symbols, abbreviated terms, and notation specific to this document are as follows:

5.1 Symbols

DR	Divide ratio
FT	Frequency tolerance
LF	Link frequency (LF = $1/T_{pri}$)
M_h	RF signal envelope ripple (overshoot)
M_{hh}	FHSS signal envelope ripple (overshoot)
M_l	RF signal envelope ripple (undershoot)
M_{hl}	FHSS signal envelope ripple (undershoot)
M_s	RF signal level when OFF
M_{hs}	FHSS signal level during a hop
Q	Slot-count parameter
R=>T	Interrogator-to-Tag
RTcal	Interrogator-to-Tag calibration symbol
T_1	Time from Interrogator transmission to Tag response
T_2	Time from Tag response to Interrogator transmission
T_3	Time an Interrogator waits, after T_1, before it issues another command
T_4	Minimum time between Interrogator commands
T_f or $T_{f,10\text{-}90\%}$	RF signal envelope fall time
T_{hf}	FHSS signal envelope fall time
T_{hr}	FHSS signal envelope rise time
T_{hs}	FHSS signal settling time
T_{pri}	Link pulse-repetition interval (T_{pri} = 1/LF)
T_r or $T_{r,10\text{-}90\%}$	RF signal envelope rise time
T_s	RF signal settling time
T=>R	Tag-to-Interrogator
TRcal	Tag-to-Interrogator calibration symbol
x_{fp}	floating-point value
$xxxx_2$	binary notation
$xxxx_h$	hexadecimal notation

5.2 Abbreviated terms

AFI	Application family identifier
AM	Amplitude modulation
ASK	Amplitude shift keying
CEPT	Conference of European Posts and Telecommunications
Ciphertext	Information that is cover-coded
CRC	Cyclic redundancy check
CW	Continuous wave
dBch	Decibels referenced to the integrated power in the reference channel
DSB	Double sideband
DSB-ASK	Double-sideband amplitude-shift keying
EPC	Electronic product code
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FHSS	Frequency-hopping spread spectrum

Handle	16-bit Tag-authentication number
NSI	Numbering system identifier
PIE	Pulse-interval encoding
Pivot	The average length of an R=>T data symbol: *pivot* = $(0_{length} + 1_{length}) / 2$
Plaintext	Information that is not cover-coded
ppm	Parts-per-million
PSK	Phase shift keying or phase shift keyed
PR-ASK	Phase-reversal amplitude shift keying
RF	Radio frequency
RFID	Radio-frequency identification
RFU	Reserved for future use
RN16	16-bit random or pseudo-random number
RNG	Random or pseudo-random number generator
ITF	Interrogator talks first (reader talks first)
SSB	Single sideband
SSB-ASK	Single-sideband amplitude-shift keying
TDM	Time-division multiplexing or time-division multiplexed (as appropriate)
TID	Tag-identification or Tag identifier, depending on context
Word	16 bits

5.3 Notation

This specification uses the following notational conventions:

- States and flags are denoted in bold. Example: **ready**.
- Commands are denoted in italics. Variables are also denoted in italics. Where there might be confusion between commands and variables, this specification will make an explicit statement. Example: *Query*.
- Command parameters are underlined. Example: Pointer.
- For logical negation, labels are preceded by '~'. Example: If **flag** is true, then **~flag** is false.
- The symbol, R=>T, refers to commands or signaling from an Interrogator to a Tag (Reader-to-Tag).
- The symbol, T=>R, refers to commands or signaling from a Tag to an Interrogator (Tag-to-Reader).

6. Protocol requirements

6.1 Protocol overview

6.1.1 Physical layer

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

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

6.1.2 Tag-identification layer

An Interrogator manages Tag populations using three basic operations:

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

b) **Inventory.** The operation of identifying Tags. An Interrogator begins an inventory round by transmitting a *Query* command in one of four sessions. One or more Tags may reply. The Interrogator detects a single Tag reply and requests the PC, EPC, and CRC-16 from the Tag. Inventory comprises multiple commands. An inventory round operates in one and only one session at a time.

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

6.2 Protocol parameters

6.2.1 Signaling – Physical and media access control (MAC) parameters

Table 6.1 and Table 6.2 provide an overview of parameters for R=>T and T=>R communications according to this specification; for detailed requirements refer to the referenced Subclause. For those parameters that do not apply to or are not used in this specification, the notation “N/A” shall indicate that the parameter is “Not Applicable”.

Table 6.1 – Interrogator-to-Tag (R=>T) communications

Ref.	Parameter Name	Description	Subclause
M1-Int:1	Operating Frequency Range	860 – 960 MHz, as required by local regulations	6.3.1.1
M1-Int:1a	Default Operating Frequency	Determined by local radio regulations and by the radio-frequency environment at the time of the communication	6.3.1.1
M1-Int:1b	Operating Channels (spread-spectrum systems)	50 channels for dense-Interrogator operation in FCC part 15.247 regulatory environments; other channelization allowed	6.3.1.2.10, Table 6.10
M1-Int:1c	Operating Frequency Accuracy	As specified	6.3.1.2.1
M1-Int:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	In accordance with local regulations	6.3.1.2.9
M1-Int:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	In accordance with local regulations	6.3.1.2.9
M1-Int:2	Occupied Channel Bandwidth	In accordance with local regulations	N/A
M1-Int:2a	Minimum Receiver Bandwidth	In accordance with local regulations	N/A
M1-Int:3	Interrogator Transmit Maximum EIRP	In accordance with local regulations	N/A
M1-Int:4	Interrogator Transmit Spurious Emissions	As specified; tighter emission limits may be imposed by local regulations	6.3.1.2.11
M1-Int:4a	Interrogator Transmit Spurious Emissions, In-Band (spread-spectrum systems)	As specified; tighter emission limits may be imposed by local regulations	6.3.1.2.11
M1-Int:4b	Interrogator Transmit Spurious Emissions, Out-of-Band	As specified; tighter emission limits may be imposed by local regulations	6.3.1.2.11
M1-Int:5	Interrogator Transmitter Spectrum Mask	As specified; tighter emission limits may be imposed by local regulations	Figure 6.6, Figure 6.7
M1-Int:6	Timing	As specified	6.3.1.5, Figure 6.16, Table 6.13
M1-Int:6a	Transmit-to-Receive Turn-Around Time	MAX(RTcal,$10T_{pri}$) nominal	6.3.1.5, Figure 6.16, Table 6.13
M1-Int:6b	Receive-to-Transmit Turn-Around Time	$3T_{pri}$ minimum; $20T_{pri}$ maximum when Tag is in **reply** & **acknowledged** states; no limit otherwise	6.3.1.5, Figure 6.16, Table 6.13
M1-Int:6c	Dwell Time or Interrogator Transmit Power-On Ramp	1500 µs, maximum settling time	6.3.1.2.6, Figure 6.3, Table 6.7
M1-Int:6d	Decay Time or Interrogator Transmit Power-Down Ramp	500 µs, maximum	6.3.1.2.7 Figure 6.3, Table 6.8
M1-Int:7	Modulation	DSB-ASK, SSB-ASK, or PR-ASK	6.3.1.2.2
M1-Int:7a	Spreading Sequence (direct-sequence [DSSS] systems)	N/A	N/A
M1-Int:7b	Chip Rate (spread-spectrum systems)	N/A	N/A
M1-Int:7c	Chip Rate Accuracy (spread-spectrum systems)	N/A	N/A
M1-Int:7d	Modulation Depth	90% nominal	6.3.1.2.5, Figure 6.2, Table 6.6
M1-Int:7e	Duty Cycle	48% – 82.3% (time the waveform is high)	6.3.1.2.3, Figure 6.1, Table 6.6
M1-Int:7f	FM Deviation	N/A	N/A
M1-Int:8	Data Coding	PIE	6.3.1.2.3, Figure 6.1
M1-Int:9	Bit Rate	26.7 kbps to 128 kbps (assuming equiprobable data)	6.3.1.2.4
M1-Int:9a	Bit Rate Accuracy	+/– 1%, minimum	6.3.1.2.4, Table 6.5

Ref.	Parameter Name	Description	Subclause
M1-Int:10	Interrogator Transmit Modulation Accuracy	As specified	6.3.1.2.4
M1-Int:11	Preamble	Required	6.3.1.2.8
M1-Int:11a	Preamble Length	As specified	6.3.1.2.8
M1-Int:11b	Preamble Waveform(s)	As specified	Figure 6.4
M1-Int:11c	Bit Sync Sequence	None	N/A
M1-Int:11d	Frame Sync Sequence	Required	6.3.1.2.8, Figure 6.4
M1-Int:12	Scrambling (spread-spectrum systems)	N/A	N/A
M1-Int:13	Bit Transmission Order	MSB is transmitted first	6.3.1.4
M1-Int:14	Wake-up process	As specified	6.3.1.3.4
M1-Int:15	Polarization	Interrogator dependent; not specified by this document	N/A

Table 6.2 – Tag-to-Interrogator (T=>R) communications

Ref.	Parameter Name	Description	Subclause
M1-Tag:1	Operating Frequency Range	860 – 960 MHz, inclusive	6.3.1.1
M1-Tag:1a	Default Operating Frequency	Tags respond to Interrogator signals that satisfy M1-Int:1a	6.3.1.1
M1-Tag:1b	Operating Channels (spread-spectrum systems)	Tags respond to Interrogator signals that satisfy M1-Int:1b	6.3.1.2.10, Table 6.10
M1-Tag:1c	Operating Frequency Accuracy	As specified	6.3.1.3.3 Table 6.11
M1-Tag:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	Tags respond to Interrogator signals that satisfy M1-Int:1d	6.3.1.2.9
M1-Tag:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	Tags respond to Interrogator signals that satisfy M1-Int:1e	6.3.1.2.9
M1-Tag:2	Occupied Channel Bandwidth	In accordance with local regulations	N/A
M1-Tag:3	Transmit Maximum EIRP	In accordance with local regulations	N/A
M1-Tag:4	Transmit Spurious Emissions	In accordance with local regulations	N/A
M1-Tag:4a	Transmit Spurious Emissions, In-Band (spread spectrum systems)	In accordance with local regulations	N/A
M1-Tag:4b	Transmit Spurious Emissions, Out-of-Band	In accordance with local regulations	N/A
M1-Tag:5	Transmit Spectrum Mask	In accordance with local regulations	N/A
M1-Tag:6a	Transmit-to-Receive Turn-Around Time	$3T_{pri}$ minimum, $20T_{pri}$ maximum in **reply** & **acknowledged** states; no limit otherwise	6.3.1.5 Figure 6.16, Table 6.13
M1-Tag:6b	Receive-to-Transmit Turn-Around Time	MAX(RTcal,$10T_{pri}$) nominal	6.3.1.5 Figure 6.16, Table 6.13
M1-Tag:6c	Dwell Time or Transmit Power-On Ramp	Receive commands 1.5 ms after power-up	6.3.1.3.4
M1-Tag:6d	Decay Time or Transmit Power-Down Ramp	N/A	N/A

Ref.	Parameter Name	Description	Subclause
M1-Tag:7	Modulation	ASK and/or PSK modulation (selected by Tag)	6.3.1.3.1
M1-Tag:7a	Spreading Sequence (direct sequence [DSSS] systems)	N/A	N/A
M1-Tag:7b	Chip Rate (spread spectrum systems)	N/A	N/A
M1-Tag:7c	Chip Rate Accuracy (spread spectrum systems)	N/A	N/A
M1-Tag:7d	On-Off Ratio	Tag dependent; not specified by this document	N/A
M1-Tag:7e	Subcarrier Frequency	40 kHz to 640 kHz	6.3.1.3.3 Table 6.11
M1-Tag:7f	Subcarrier Frequency Accuracy	As specified	Table 6.11
M1-Tag:7g	Subcarrier Modulation	Miller, at the data rate	6.3.1.3.2.3, Figure 6.13
M1-Tag:7h	Duty Cycle	FM0: 50%, nominal Subcarrier: 50%, nominal	6.3.1.3.2.1, 6.3.1.3.2.3
M1-Tag:7I	FM Deviation	N/A	N/A
M1-Tag:8	Data Coding	Baseband FM0 or Miller-modulated subcarrier (selected by the Interrogator)	6.3.1.3.2
M1-Tag:9	Bit Rate	FM0: 40 kbps to 640 kbps Subcarrier modulated: 5 kbps to 320 kbps	6.3.1.3.3 Table 6.11
M1-Tag:9a	Bit Rate Accuracy	Same as Subcarrier Frequency Accuracy; see M1-Tag:7f	6.3.1.3.3, Table 6.11, Table 6.12
M1-Tag:10	Tag Transmit Modulation Accuracy (frequency-hopping [FHSS] systems	N/A	N/A
M1-Tag:11	Preamble	Required	6.3.1.3.2.2, 6.3.1.3.2.4
M1-Tag:11a	Preamble Length	As specified	Figure 6.11, Figure 6.15
M1-Tag:11b	Preamble Waveform	As specified	Figure 6.11, Figure 6.15
M1-Tag:11c	Bit-Sync Sequence	None	N/A
M1-Tag:11d	Frame-Sync Sequence	None	N/A
M1-Tag:12	Scrambling (spread-spectrum systems)	N/A	N/A
M1-Tag:13	Bit Transmission Order	MSB is transmitted first	6.3.1.4
M1-Tag:14	Reserved	Deliberately left blank	N/A
M1-Tag:15	Polarization	Tag dependent; not specified by this document	N/A
M1-Tag:16	Minimum Tag Receiver Bandwidth	Tag dependent; not specified by this document.	N/A

6.2.2 Logical – Operating procedure parameters

Table 6.3 and Table 6.4 identify and describe parameters used by an Interrogator during the selection, inventory, and access of Tags according to this specification. For those parameters that do not apply to or are not used in this specification, the notation "N/A" shall indicate that the parameter is "Not Applicable".

Table 6.3 – Tag inventory and access parameters

Ref.	Parameter Name	Description	Subclause
M1-P:1	Who talks first	Interrogator	6.3
M1-P:2	Tag addressing capability	As specified	6.3.2.1
M1-P:3	Tag EPC	Contained in Tag memory	6.3.2.1
M1-P:3a	EPC Length	As specified in EPC™ Tag Data Standards	N/A
M1-P:3b	EPC Format	As specified in EPC™ Tag Data Standards	N/A
M1-P:4	Read size	Multiples of 16 bits	6.3.2.10.3.2 Table 6.31
M1-P:5	Write Size	Multiples of 16 bits	6.3.2.10.3.3, Table 6.33, 6.3.2.10.3.7, Table 6.43
M1-P:6	Read Transaction Time	Varied with R=>T and T=>R link rate and number of bits being read	6.3.2.10.3.2
M1-P:7	Write Transaction Time	20 ms (maximum) after end of *Write* command	6.3.2.10.3.3, 6.3.2.10.3.7, Figure 6.22
M1-P:8	Error detection	Interrogator-to-Tag: *Select* command: CRC-16 *Query* command: CRC-5 Other Inventory commands: Command length Access commands: CRC-16 Tag-to-Interrogator: PC, EPC: CRC-16 RN16: None or CRC-16 (varies with command) handle: CRC-16 All other: CRC-16	6.3.2.10 and its subsections
M1-P:9	Error correction	None	N/A
M1-P:10	Memory size	Tag dependent, extensible (size is neither limited nor specified by this document)	N/A
M1-P:11	Command structure and extensibility	As specified	Table 6.16

Table 6.4 – Collision management parameters

Ref.	Parameter Name	Description	Subclause
M1-A:1	Type (Probabilistic or Deterministic)	Probabilistic	6.3.2.6
M1-A:2	Linearity	Linear up to 2^{15} Tags in the Interrogator's RF field; above that number, NlogN for Tags with unique EPCs	6.3.2.8
M1-A:3	Tag inventory capacity	Unlimited for Tags with unique EPCs	6.3.2.8

6.3 Description of operating procedure

The operating procedure defines the physical and logical requirements for an Interrogator-talks-first (ITF), slotted random anticollision, RFID system operating in the 860 MHz – 960 MHz frequency range.

6.3.1 Signaling

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

6.3.1.1 Operational frequencies

Tags shall be capable of receiving power from and communicating with Interrogators within the frequency range from 860 MHz to 960 MHz, inclusive. An Interrogator's choice of operational frequency will be determined by local radio regulations and by the local radio-frequency environment. Interrogators certified for operation in dense-Interrogator environments shall be capable of communications as described in Annex G.

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

An Interrogator communicates with one or more Tags by modulating an RF carrier using DSB-ASK, SSB-ASK, or PR-ASK with PIE encoding. Interrogators shall use a fixed modulation format and data rate for the duration of an inventory round, where "inventory round" is defined in 6.3.2.8. The Interrogator sets the data rate by means of the preamble that initiates the round.

The high values in Figure 6.1, Figure 6.2, Figure 6.3, Figure 6.4, and Figure 6.5, correspond to emitted CW (i.e. an Interrogator delivering power to the Tag or Tags) whereas the low values correspond to attenuated CW.

6.3.1.2.1 Interrogator frequency accuracy

Interrogators certified for operation in single- or multiple-Interrogator environments shall have a frequency accuracy that meets local regulations. Interrogators certified for operation in dense-Interrogator environments shall have a frequency accuracy of +/– 10 ppm over the nominal temperature range (–25 °C to +40 °C) and +/– 20 ppm over the extended temperature range (–40°C to +65 °C), unless local regulations specify tighter accuracy, in which case the Interrogator frequency accuracy shall meet the local regulations.

6.3.1.2.2 Modulation

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

6.3.1.2.3 Data encoding

The R=>T link shall use PIE, shown in Figure 6.1. Tari is the reference time interval for Interrogator-to-Tag signaling, and is the duration of a data-0. High values represent transmitted CW; low values represent attenuated CW. The tolerance on all parameters shall be +/–1%. Pulse modulation depth, rise time, fall time, and PW shall be as specified in Table 6.6, and shall be the same for a data-0 and a data-1. Interrogators shall use a fixed modulation depth, rise time, fall time, PW, and Tari for the duration of an inventory round. The RF envelope shall be as specified in Figure 6.2.

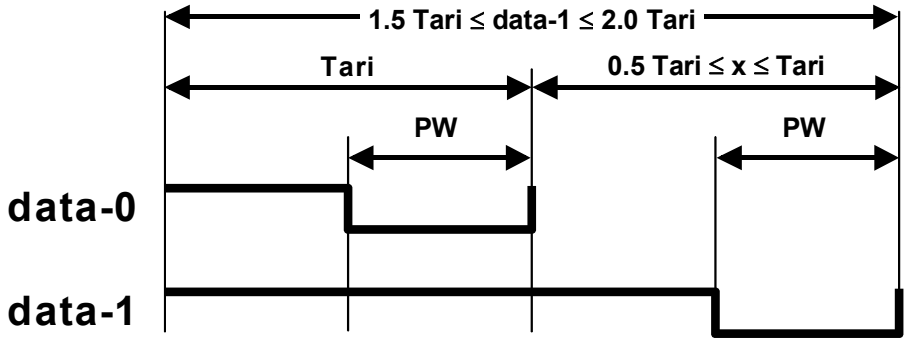


Figure 6.1 – PIE symbols

6.3.1.2.4 Data rates

Interrogators shall communicate using Tari values between 6.25µs and 25µs, inclusive. Interrogator compliance shall be evaluated using the preferred Tari values specified in Table 6.5 and the encoding shown in Figure 6.1 with *x* = 0.5 Tari and *x* = 1.0 Tari. An Interrogator shall use fixed data-0 and data-1 symbol lengths for the duration of an inventory round, where "inventory round" is defined in 6.3.2.8. The choice of Tari value shall be in accordance with local radio regulations.

Table 6.5 – Preferred Interrogator-to-Tag Tari values

Tari Value	Tari-Value Tolerance	Spectrum
6.25 µs	+/– 1%	DSB-ASK, SSB-ASK, or PR-ASK
12.5 µs	+/– 1%	
25 µs	+/– 1%	

6.3.1.2.5 R=>T RF envelope

The R=>T RF envelope shall comply with Figure 6.2 and Table 6.6. The electric field strength A is the maximum amplitude of the RF envelope. Tari is defined in Figure 6.1. The pulsewidth is measured at the 50% point on the pulse. An Interrogator shall not change the R=>T modulation type (i.e. shall not switch between DSB-ASK, SSB-ASK, or PR-ASK) without first powering down its RF waveform (see 6.3.1.2.7).

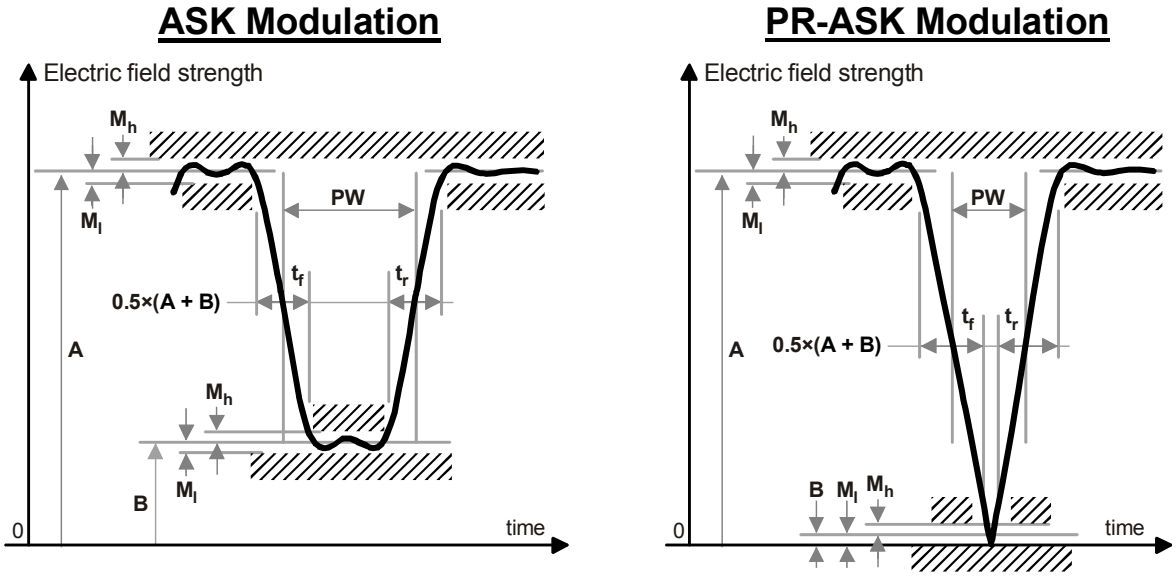


Figure 6.2 – Interrogator-to-Tag RF envelope

Table 6.6 – RF envelope parameters

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

6.3.1.2.6 Interrogator power-up waveform

The Interrogator power-up RF envelope shall comply with Figure 6.3 and Table 6.7. Once the carrier level has risen above the 10% level, the power-up envelope shall rise monotonically until at least the ripple limit M_l. The RF envelope shall not fall below the 90% point in Figure 6.3 during interval T_s. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table 6.7 (i.e. before T_s).

6.3.1.2.7 Interrogator power-down waveform

The Interrogator power-down RF envelope shall comply with Figure 6.3 and Table 6.8. Once the carrier level has fallen below the 90% level, the power-down envelope shall fall monotonically until the power-off limit M_s. Once powered off, an Interrogator shall remain powered off for a least 1ms before powering up again.

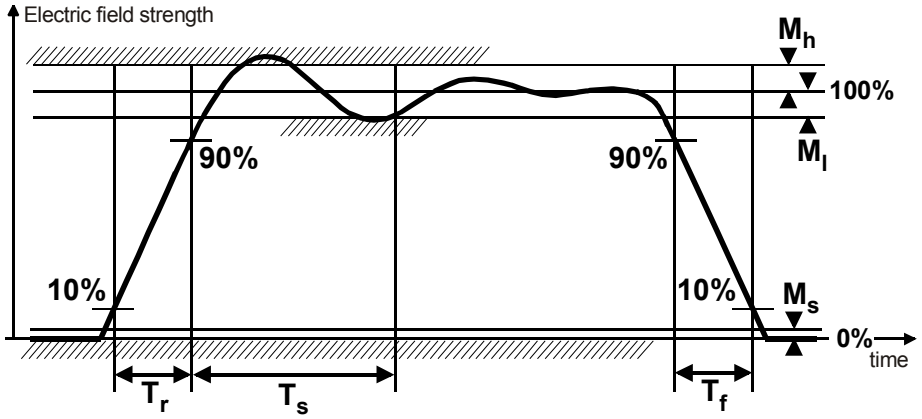


Figure 6.3 – Interrogator power-up and power-down RF envelope

Table 6.7 – Interrogator power-up waveform parameters

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

Table 6.8 – Interrogator power-down waveform parameters

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

6.3.1.2.8 R=>T preamble and frame-sync

An Interrogator shall begin all R=>T signaling with either a preamble or a frame-sync, both of which are shown in Figure 6.4. A preamble shall precede a *Query* command (see 6.3.2.10.2.1) and denotes the start of an inventory round. All other signaling shall begin with a frame-sync. The tolerance on all parameters specified in units of Tari shall be +/–1%. PW shall be as specified in Table 6.6. The RF envelope shall be as specified in Figure 6.2. A Tag may compare the length of the data-0 with the length of RTcal to validate the preamble.

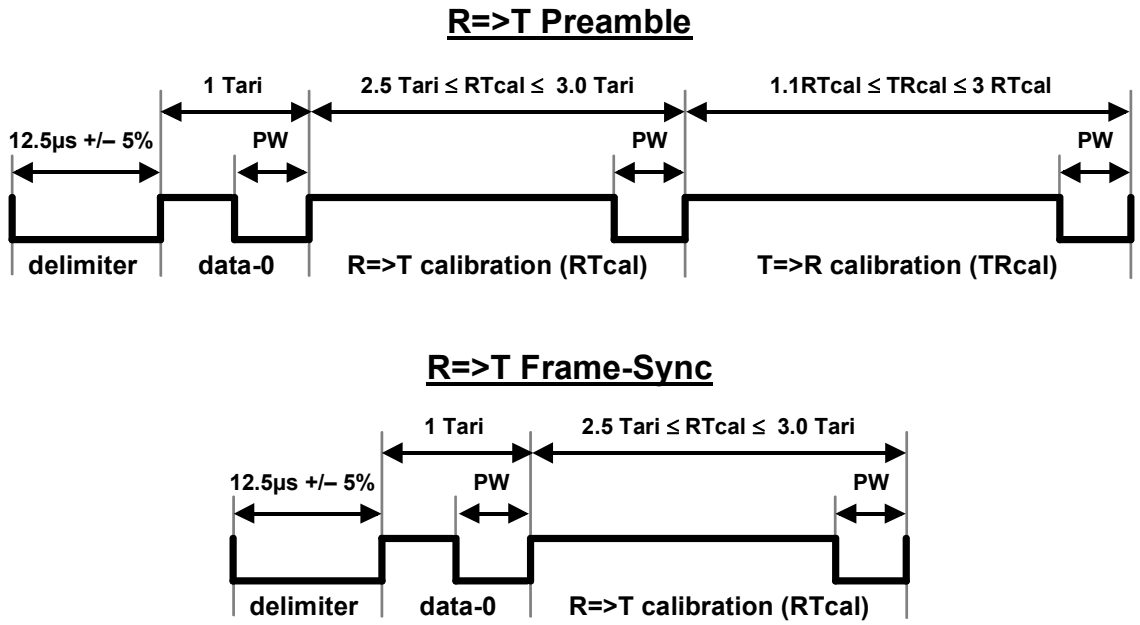


Figure 6.4 – R=>T preamble and frame-sync

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

- **RTcal:** An Interrogator shall set RTcal equal to the length of a data-0 symbol plus the length of a data-1 symbol (RTcal = 0_{length} + 1_{length}). A Tag shall measure the length of RTcal and compute *pivot* = RTcal / 2. The Tag shall interpret subsequent Interrogator symbols shorter than *pivot* to be data-0s, and subsequent Interrogator symbols longer than *pivot* to be data-1s. The Tag shall interpret symbols longer than 4 RTcal to be bad data. Prior to changing RTcal, an Interrogator shall transmit CW for a minimum of 8 RTcal.
- **TRcal:** An Interrogator shall specify a Tag's backscatter link frequency (its FM0 datarate or the frequency of its Miller subcarrier) using the TRcal and divide ratio (DR) in the preamble and payload, respectively, of a *Query* command that initiates an inventory round. Equation (1) specifies the relationship between the backscatter link frequency (LF), TRcal, and DR. A Tag shall measure the length of TRcal, compute LF, and adjust its T=>R link rate to be equal to LF (Table 6.11 shows LF values and tolerances). The TRcal and RTcal that an Interrogator uses in any inventory round shall meet the constraints in Equation (2):

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

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

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

6.3.1.2.9 Frequency-hopping spread-spectrum waveform

When an Interrogator uses frequency-hopping spread spectrum (FHSS) signaling, the Interrogator's RF envelope shall comply with Figure 6.5 and Table 6.9. The RF envelope shall not fall below the 90% point in Figure 6.5 during interval T_{hs}. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table 6.9 (i.e. before T_{hs}). The maximum time between frequency hops and the minimum RF-off time during a hop shall meet local regulatory requirements.

6.3.1.2.10 Frequency-hopping spread-spectrum channelization

Interrogators certified for operation in single-Interrogator environments shall meet local regulations for spread-spectrum channelization. Interrogators certified for operation in multiple- or dense-Interrogator environments, when operating under FCC Title 47, Part 15 regulations, shall be additionally capable of centering their R=>T signaling in channels whose width and center frequencies are shown in Table 6.10 (Annex G describes dense-Interrogator channelized signaling).

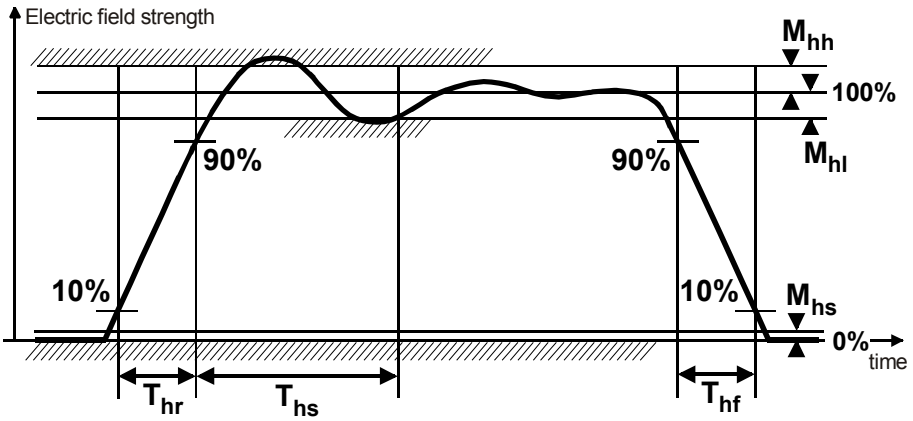


Figure 6.5 – FHSS Interrogator RF envelope

Table 6.9 – FHSS waveform parameters

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

Table 6.10 – Frequency-hopping spread-spectrum channelization

Commanded Tag backscatter format	Channel width	Channel center frequencies f_c	Guardbands
Subcarrier	500 kHz	Channel 1: 902.75 MHz Channel 2: 903.25 MHz • • Channel 50: 927.25 MHz	Lower bandedge: 902 MHz – 902.5 MHz Upper bandedge: 927.5 MHz – 928 MHz
FM0	In accordance with local regulations		

6.3.1.2.11 Transmit mask

Interrogators certified for operation according to this protocol shall meet local regulations for out-of-channel and out-of-band spurious radio-frequency emissions.

Interrogators certified for operation in multiple-Interrogator environments, in addition to meeting local regulations, shall also meet the specified Multiple-Interrogator Transmit Mask:

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

- $|R - S| = 1$: $10\log_{10}(P(S) / P(R)) < -20$ dB
- $|R - S| = 2$: $10\log_{10}(P(S) / P(R)) < -50$ dB
- $|R - S| = 3$: $10\log_{10}(P(S) / P(R)) < -60$ dB
- $|R - S| > 3$: $10\log_{10}(P(S) / P(R)) < -65$ dB

Where $P()$ denotes the total integrated power in the specified channel. This mask is shown graphically in Figure 6.6, with dBch defined as dB referenced to the integrated power in the reference channel. For any transmit channel R, two exceptions to the mask are permitted, provided that

- neither exception exceeds –50 dBch, and
- neither exception exceeds local regulatory requirements.

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

Interrogators certified for operation in dense-Interrogator environments shall meet both local regulations and the Transmit Mask shown in Figure 6.6. In addition, they shall be capable of meeting the following Dense-Interrogator Transmit Mask when using dense-Interrogator channelized signaling as outlined in Annex G. Finally, unlike Interrogators certified for operation in multiple-Interrogator environments, those certified for operation in dense-Interrogator environments shall not be permitted any exceptions to a transmit mask.

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

- $|n| = 1$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -30$ dB
- $|n| = 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -60$ dB
- $|n| > 2$: $10\log_{10}(P(S_{BW}) / P(R_{BW})) < -65$ dB

Where $P()$ denotes the total integrated power in the 2.5/Tari reference bandwidth. This mask is shown graphically in Figure 6.7, with dBch defined as dB referenced to the integrated power in the reference channel.

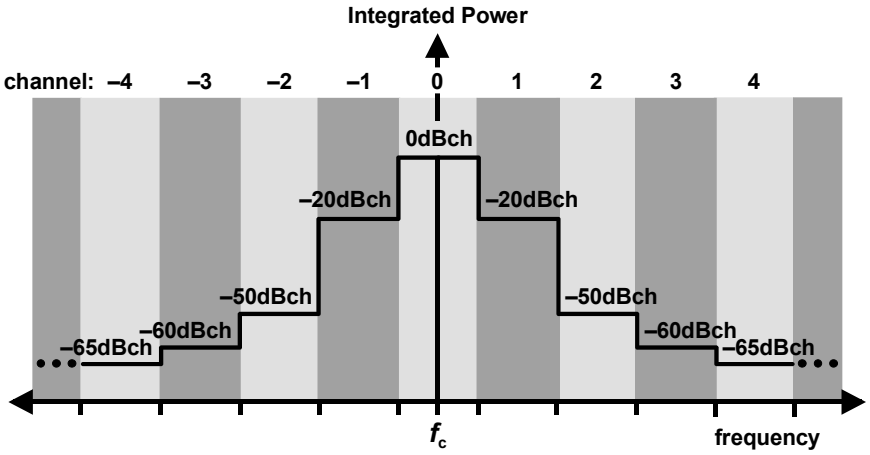


Figure 6.6 – Transmit mask for multiple-Interrogator environments

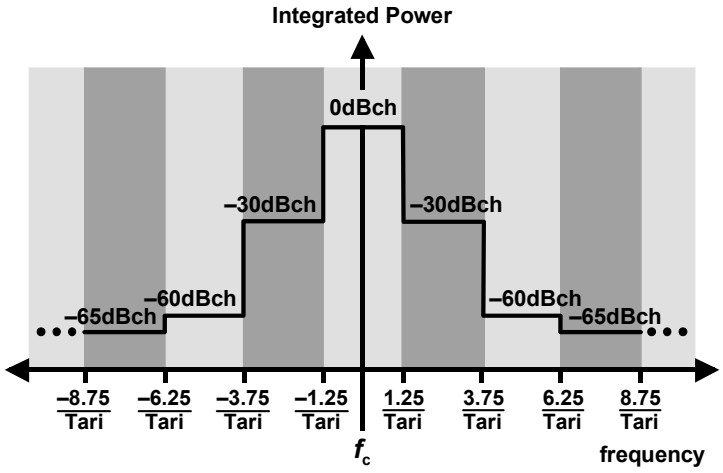


Figure 6.7 – Transmit mask for dense-Interrogator environments

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

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

A Tag shall backscatter using a fixed modulation format, data encoding, and data rate for the duration of an inventory round, where "inventory round" is defined in 6.3.2.8. The Tag selects the modulation format; the Interrogator selects the encoding and data rate by means of the *Query* command that initiates the round. The low values in Figure 6.9, Figure 6.10, Figure 6.11, Figure 6.13, Figure 6.14, and Figure 6.15 correspond to the antenna-reflectivity state the Tag exhibits during the CW period prior to a T=>R preamble (i.e. Tag absorbing power), whereas the high values correspond to the antenna-reflectivity state the Tag exhibits during the first high pulse of a T=>R preamble (i.e. Tag reflecting power).

6.3.1.3.1 Modulation

Tag backscatter shall use ASK and/or PSK modulation. The Tag vendor selects the modulation format. Interrogators shall be capable of demodulating either modulation type.

6.3.1.3.2 Data encoding

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

6.3.1.3.2.1 FM0 baseband

Figure 6.8 shows basis functions and a state diagram for generating FM0 (bi-phase space) encoding. FM0 inverts the baseband phase at every symbol boundary; a data-0 has an additional mid-symbol phase inversion. The state diagram in Figure 6.8 maps a logical data sequence to the FM0 basis functions that are transmitted. The state

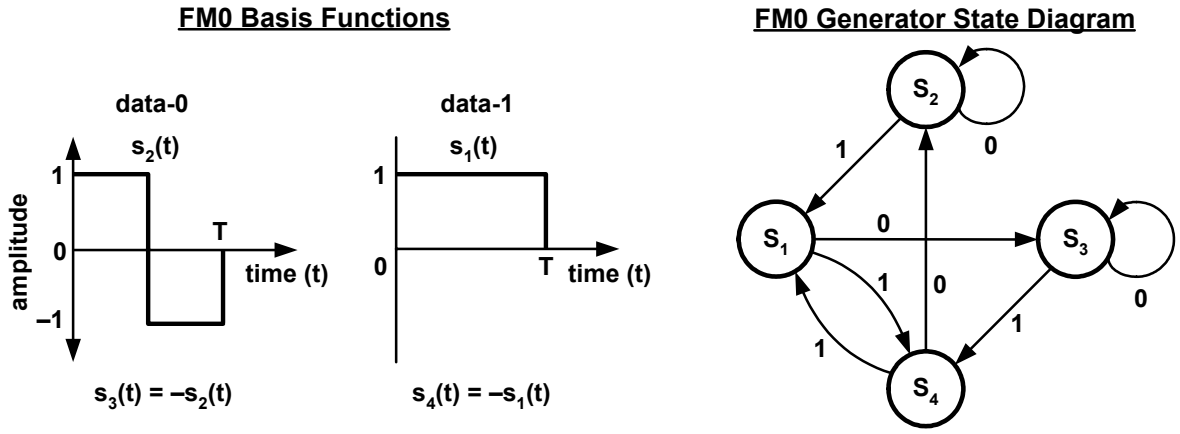


Figure 6.8 – FM0 basis functions and generator state diagram

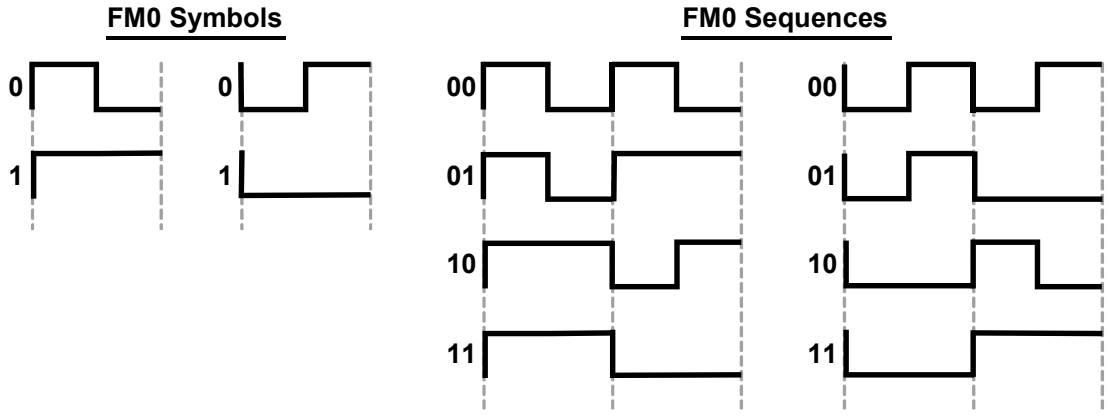


Figure 6.9 – FM0 symbols and sequences

labels, S_1–S_4, indicate four possible FM0-encoded symbols, represented by the two phases of each of the FM0 basis functions. The state labels also represent the FM0 waveform that is transmitted upon entering the state. The labels on the state transitions indicate the logical values of the data sequence to be encoded. For example, a transition from state S_2 to S_3 is disallowed because the resulting transmission would not have a phase inversion on a symbol boundary. The state diagram in Figure 6.8 does not imply any specific implementation.

Figure 6.9 shows generated baseband FM0 symbols and sequences. The duty cycle of a 00 or 11 sequence, measured at the modulator output, shall be a minimum of 45% and a maximum of 55%, with a nominal value of 50%. FM0 encoding has memory; consequently, the choice of FM0 sequences in Figure 6.9 depends on prior transmissions. FM0 signaling shall always end with a "dummy" data-1 bit at the end of a transmission, as shown in Figure 6.10.

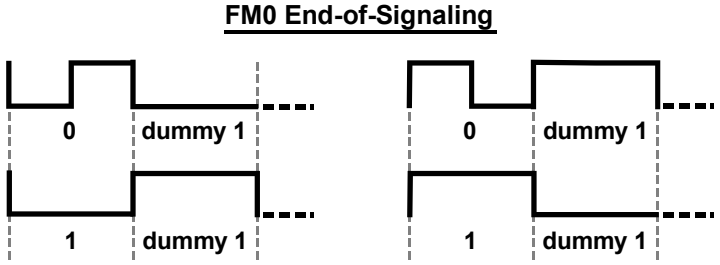


Figure 6.10 – Terminating FM0 transmissions

6.3.1.3.2.2 FM0 preamble

T=>R FM0 signaling shall begin with one of the two preambles shown in Figure 6.11. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round. The "v" shown in Figure 6.11 indicates an FM0 violation (i.e. a phase inversion should have occurred but did not).

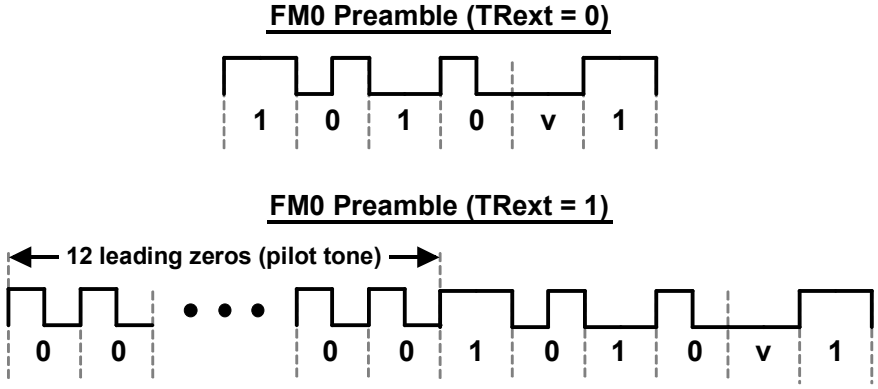


Figure 6.11 – FM0 T=>R preamble

6.3.1.3.2.3 Miller-modulated subcarrier

Figure 6.12 shows basis functions and a state diagram for generating Miller encoding. Baseband Miller inverts its phase between two data-0s in sequence. Baseband Miller also places a phase inversion in the middle of a data-1 symbol. The state diagram in Figure 6.12 maps a logical data sequence to baseband Miller basis functions. The state labels, S_1–S_4, indicate four possible Miller-encoded symbols, represented by the two phases of each of the Miller basis functions. The state labels also represent the baseband Miller waveform that is generated upon entering the state. The transmitted waveform is the baseband waveform multiplied by a square-wave at M times the symbol rate. The labels on the state transitions indicate the logical values of the data sequence to be encoded. For example, a transition from state S_1 to S_3 is disallowed because the resulting transmission would have a phase inversion on a symbol boundary between a data-0 and a data-1. The state diagram in Figure 6.12 does not imply any specific implementation.

Figure 6.13 shows Miller-modulated subcarrier sequences; the Miller sequence shall contain exactly two, four, or eight subcarrier cycles per bit, depending on the M value specified in the *Query* command that initiated the inven-

tory round (see Table 6.12). The duty cycle of a 0 or 1 symbol, measured at the modulator output, shall be a minimum of 45% and a maximum of 55%, with a nominal value of 50%. Miller encoding has memory; consequently, the choice of Miller sequences in Figure 6.13 depends on prior transmissions. Miller signaling shall always end with a “dummy” data-1 bit at the end of a transmission, as shown in Figure 6.14.

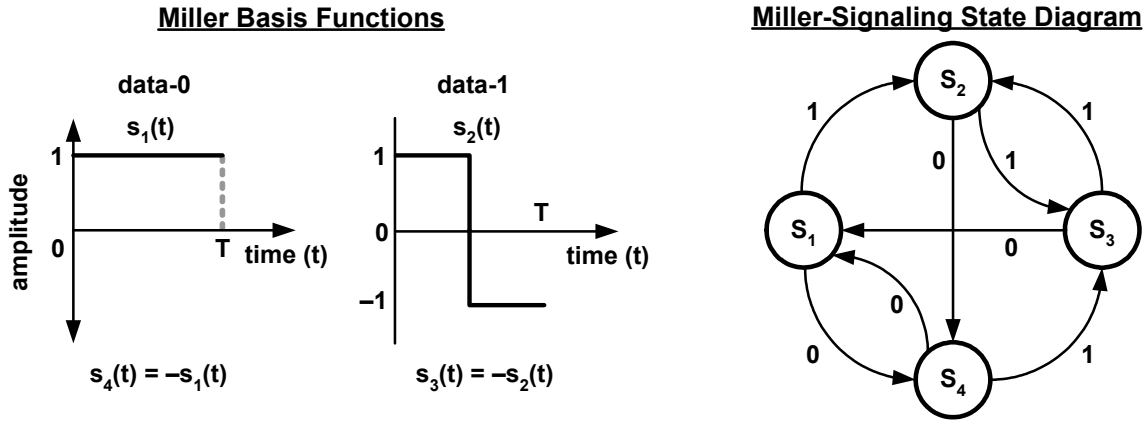


Figure 6.12 – Miller basis functions and generator state diagram

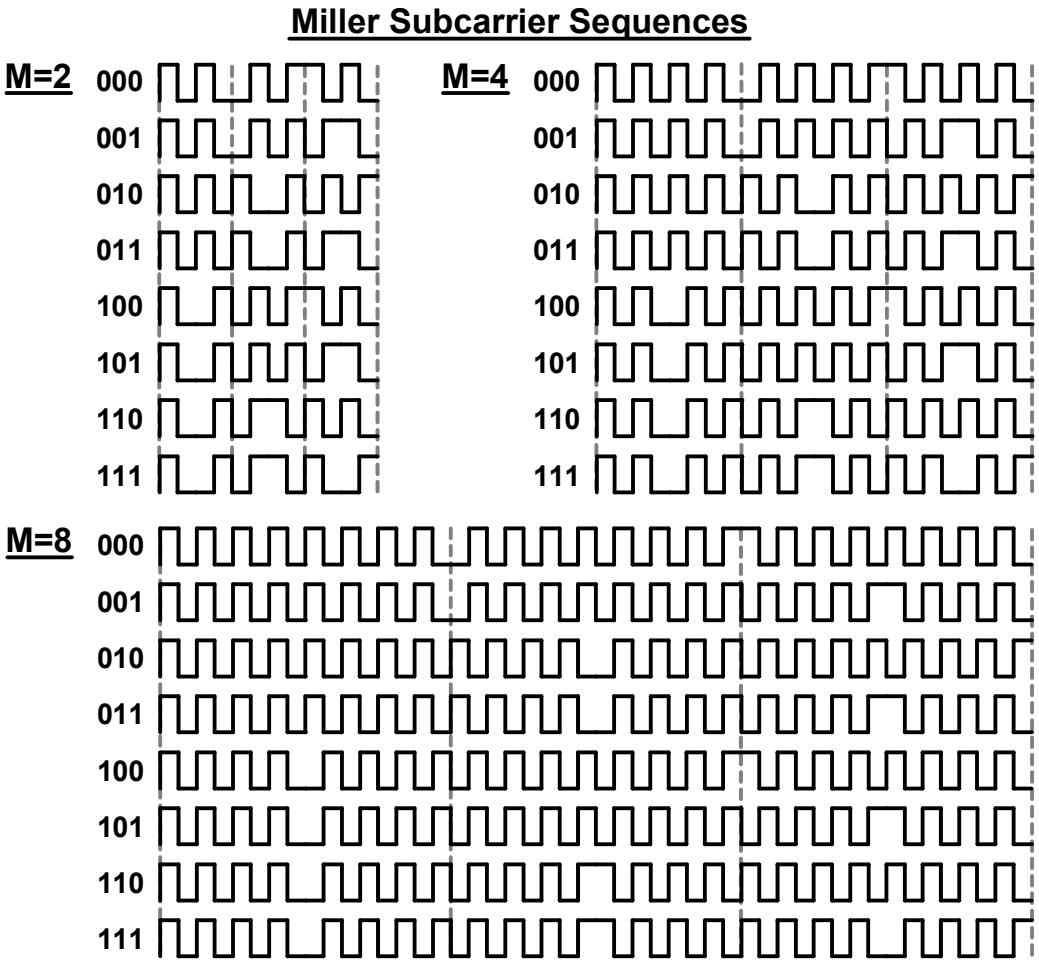


Figure 6.13 – Subcarrier sequences

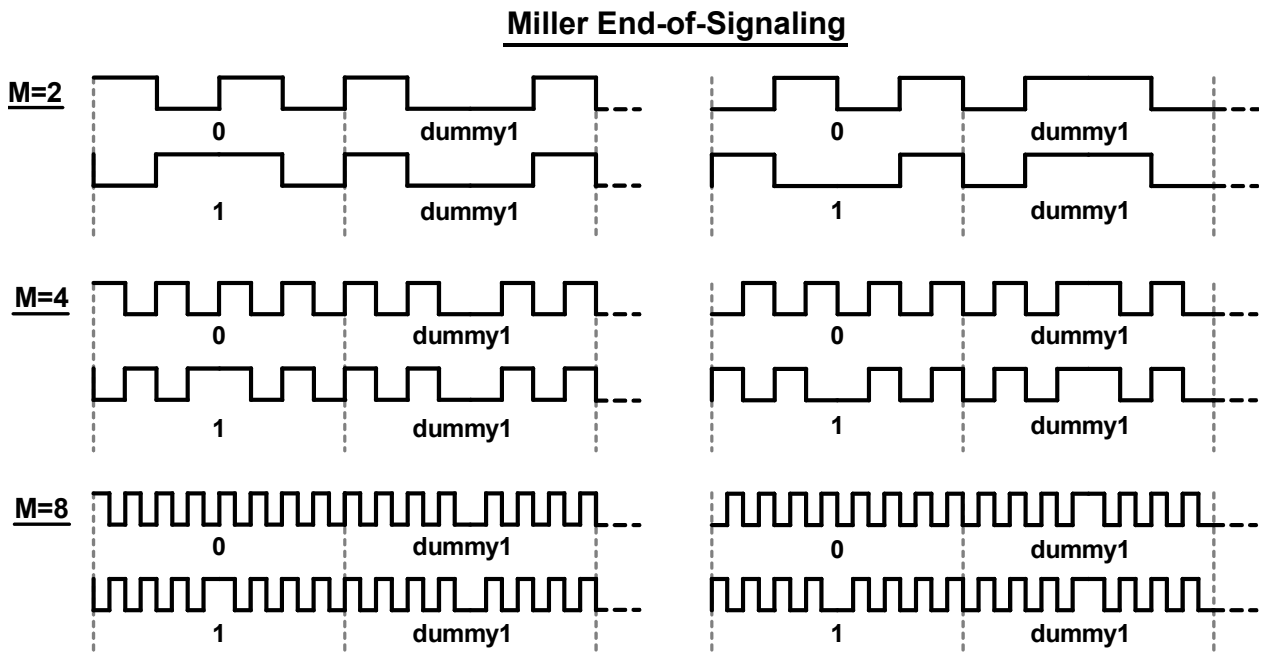


Figure 6.14 – Terminating subcarrier transmissions

6.3.1.3.2.4 Miller subcarrier preamble

T=>R subcarrier signaling shall begin with one of the two preambles shown in Figure 6.15. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round.

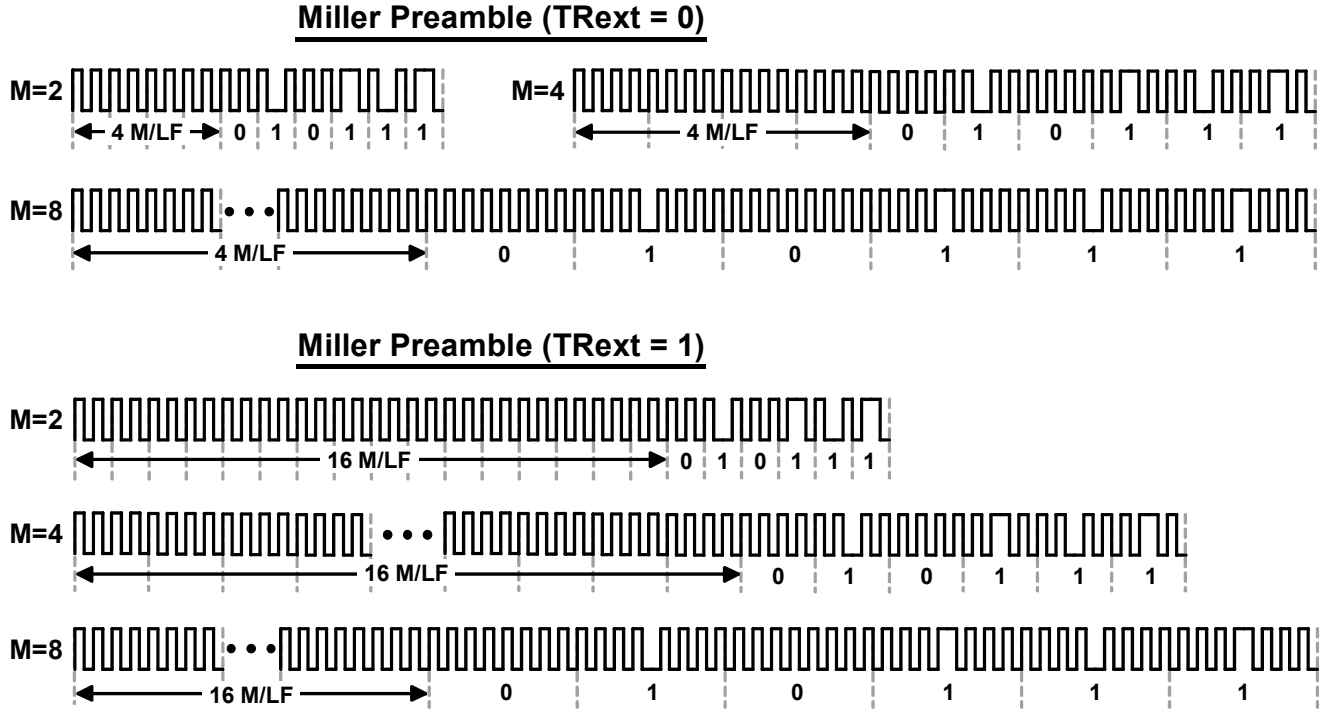


Figure 6.15 – Subcarrier T=>R preamble

6.3.1.3.3 Data rates

Tags shall support the R=>T Tari values specified in 6.3.1.2.4, the T=>R link frequencies and tolerances specified in Table 6.11, and the T=>R data rates specified in Table 6.12. The frequency-variation requirement in Table 6.11 includes both frequency drift and short-term frequency variation during Tag response to an Interrogator command. The *Query* command that initiates an inventory round specifies DR in Table 6.11 and M in Table 6.12; the preamble that precedes the *Query* specifies TRcal. LF is computed using Eq. (1). These four parameters together define the backscatter frequency, modulation type (FM0 or Miller), and T=>R data rate for the round (see also 6.3.1.2.8).

Table 6.11 – Tag-to-Interrogator link frequencies

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

Note 1. Allowing two different TRcal values (with two different DR values) to specify the same LF offers flexibility in specifying Tari and RTcal.

Table 6.12 – Tag-to-Interrogator data rates

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

6.3.1.3.4 Tag power-up timing

Tags energized by an Interrogator shall be capable of receiving and acting on Interrogator commands within a period not exceeding the maximum settling-time interval specified in Table 6.7 or Table 6.9, as appropriate (i.e. within T_s or T_{hs}, respectively).

6.3.1.3.5 Minimum operating field strength and backscatter strength

For Tags certified to this protocol, operating under manufacturer's specified conditions in accordance with local regulations, and mounted on one or more manufacturer-selected materials, the Tag manufacturer shall specify:

1. Free-space, interference-free sensitivity, and
2. Minimum relative backscattered modulated power (ASK modulation) or change in radar cross-section or equivalent (phase modulation).

6.3.1.4 Transmission order

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

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

6.3.1.5 Link timing

Figure 6.16 illustrates R=>T and T=>R link timing. The figure (not drawn to scale) defines Interrogator interactions with a Tag population. Table 6.13 shows the timing requirements for Figure 6.16, while 6.3.2.10 describes the commands. Tags and Interrogators shall meet all timing requirements shown in Table 6.13. RTcal is defined in 6.3.1.2.8; T_{pri} is the T=>R link period (T_{pri} = 1 / LF). As described in 6.3.1.2.8, an Interrogator shall use a fixed R=>T link rate for the duration of an inventory round; prior to changing the R=>T link rate, an Interrogator shall transmit CW for a minimum of 8 RTcal.

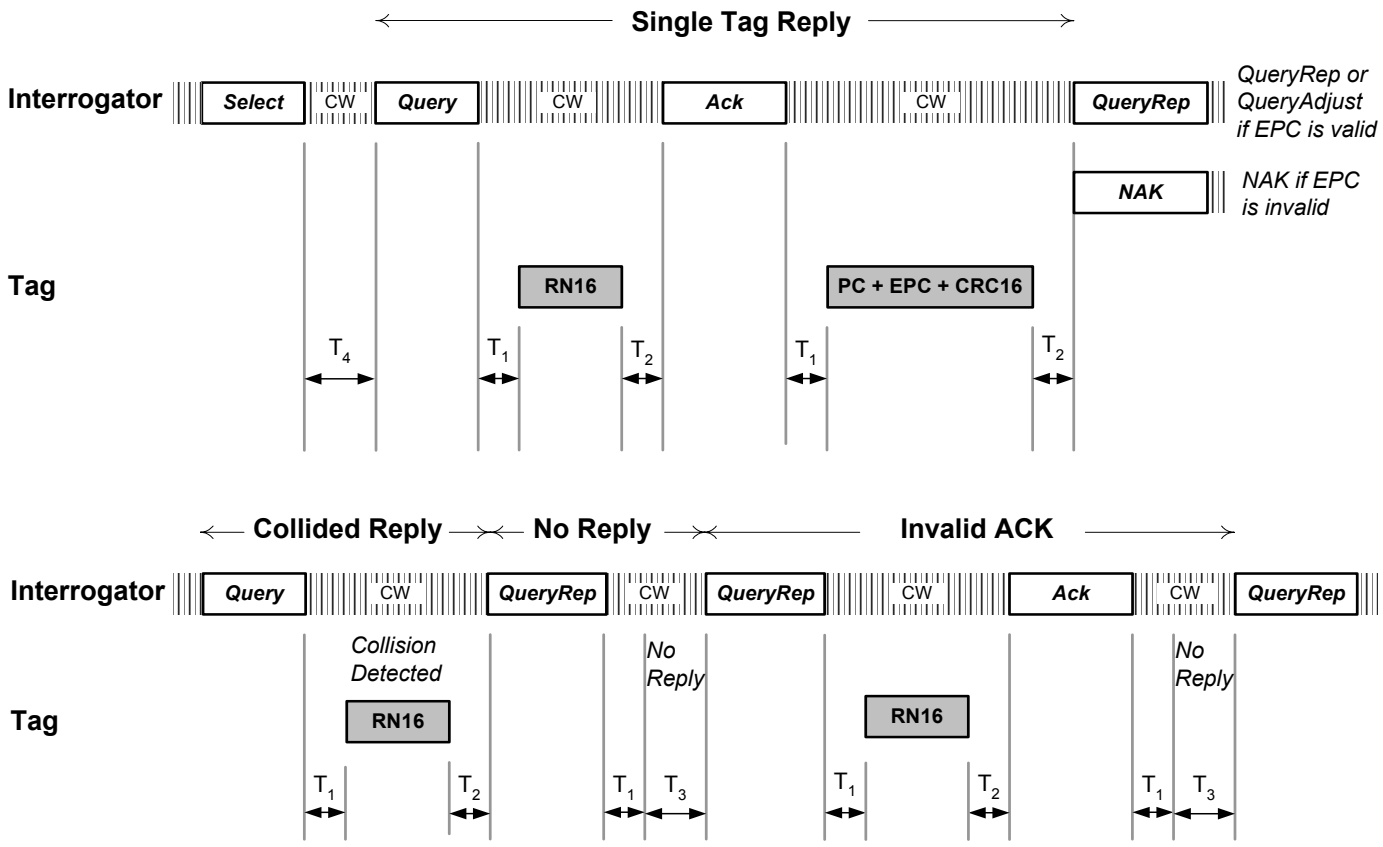


Figure 6.16 – Link timing

Table 6.13 – Link timing parameters

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

Notes

1. T_{pri} denotes either the period of an FM0 symbol or the period of a single subcarrier cycle, as appropriate.
2. The maximum value for T_2 shall apply only to Tags in the **reply** or **acknowledged** states (see 6.3.2.4.3 and 6.3.2.4.4). For a Tag in the **reply** or **acknowledged** states, if T_2 expires (i.e. reaches its maximum value):
   - Without the Tag receiving a valid command, the Tag shall transition to the **arbitrate** state (see 6.3.2.4.2),
   - During the reception of a valid command, the Tag shall execute the command,
   - During the reception of an invalid command, the Tag shall transition to **arbitrate** upon determining that the command is invalid.

   In all other states the maximum value for T_2 shall be unrestricted. "Invalid command" is defined in 6.3.2.10.
3. An Interrogator may transmit a new command prior to interval T_2 (i.e. during a Tag response). In this case the responding Tag is not required to demodulate or otherwise act on the new command, and may undergo a power-on reset.
4. FT is the frequency tolerance specified in Table 6.11
5. T_1+T_3 shall not be less than T_4.

6.3.2 Tag selection, inventory, and access

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

6.3.2.1 Tag memory

Tag memory shall be logically separated into four distinct banks, each of which may comprise zero or more memory words. A logical memory map is shown in Figure 6.17. The memory banks are:

a) **Reserved memory** shall contain the kill and access passwords. The kill password shall be stored at memory addresses 00_h to $1F_h$; the access password shall be stored at memory addresses 20_h to $3F_h$. If a Tag does not implement the kill and/or access password(s), the Tag shall act as though it had zero-valued password(s) that are permanently read/write locked (see 6.3.2.10.3.5), and the corresponding memory locations in Reserved memory need not exist.

b) **EPC memory** shall contain a CRC-16 at memory addresses 00_h to $0F_h$, Protocol-Control (PC) bits at memory addresses 10_h to $1F_h$, and a code (such as an EPC, and hereafter referred to as an EPC) that identifies the object to which the tag is or will be attached beginning at address 20_h. As detailed in 6.3.2.1.4, the PC is subdivided into an EPC length field in memory locations 10_h to 14_h, RFU bits in memory locations 15_h and 16_h, and a Numbering System Identifier (NSI) in memory locations 17_h to $1F_h$. The CRC-16, PC, and EPC shall be stored MSB first (the EPC's MSB is stored in location 20_h).

c) **TID memory** shall contain an 8-bit ISO/IEC 15963 allocation class identifier (11100010_2 for EPCglobal) at memory locations 00_h to 07_h. TID memory shall contain sufficient identifying information above 07_h for an Interrogator to uniquely identify the custom commands and/or optional features that a Tag supports. For Tags whose ISO/IEC 15963 allocation class identifier is 11100010_2, this identifying information shall comprise a 12-bit Tag mask-designer identifier (free to members of EPCglobal) at memory locations 08_h to 13_h and a 12-bit Tag model number at memory locations 14_h to $1F_h$. Tags may contain Tag- and vendor-specific data (for example, a Tag serial number) in TID memory above $1F_h$.

d) **User memory** allows user-specific data storage. The memory organization is user-defined.

The logical addressing of all memory banks shall begin at zero (00_h). The physical memory map is vendor-specific. Commands that access memory have a MemBank parameter that selects the bank, and an address parameter, specified using the EBV format described in Annex A, to select a particular memory location within that bank.

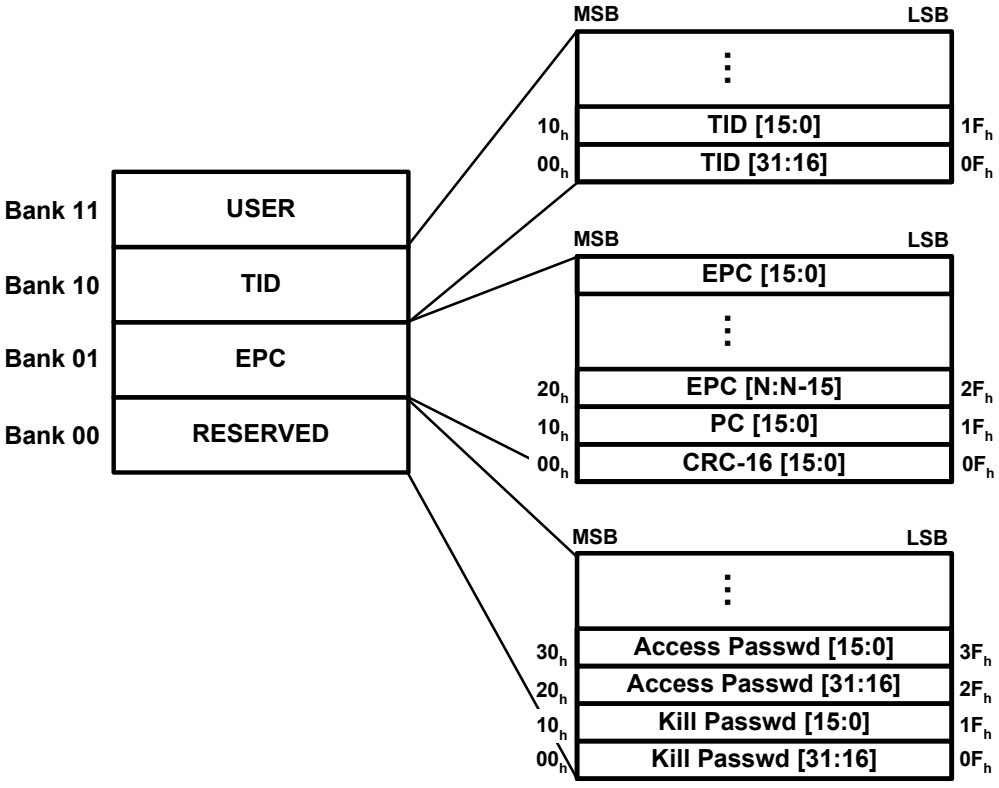


Figure 6.17 – Logical memory map

When Tags backscatter memory contents, this backscatter shall fall on word boundaries (except in the case of a truncated reply – see 6.3.2.10.1.1).

MemBank is defined as follows:

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

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

Memory writes, detailed in 6.3.2.9, involve the transfer of 16-bit words from Interrogator to Tag. A *Write* command writes 16 bits (i.e. one word) at a time, using link cover-coding to obscure the data during R=>T transmission. The optional *BlockWrite* command writes one or more 16-bit words at a time, without link cover-coding. The optional *BlockErase* command erases one or more 16-bit words at a time. A *Write*, *BlockWrite*, or *BlockErase* shall not alter a Tag's killed status regardless of the memory address (whether valid or invalid) specified in the command.

Interrogators may lock, permanently lock, unlock, or permanently unlock memory, thereby preventing or allowing subsequent changes (as appropriate). See 6.3.2.9 and 6.3.2.10.3.5 for a detailed description of memory locking and unlocking. The kill and access passwords are individually lockable, as are EPC, TID, and User memory. If the kill and/or access passwords are locked they are rendered both unwriteable and unreadable (but still usable by the *Kill* and *Access* commands, respectively), unlike other memory banks, which are always readable regardless of their lock status.

6.3.2.1.1 Kill password

The kill password is a 32-bit value stored in Reserved memory 00_h to $1F_h$, MSB first. The default (unprogrammed) value shall be zero. An Interrogator shall use a Tag's kill password once, to kill the Tag and render it silent thereafter. A Tag shall not execute a kill operation if its kill password is zero. A Tag that does not implement a kill password acts as though it had a zero-valued kill password that is permanently read/write locked.

6.3.2.1.2 Access password

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

6.3.2.1.3 CRC-16

A CRC-16 is a cyclic-redundancy check that an Interrogator uses when protecting certain R=>T commands, and a Tag uses when protecting certain backscattered T=>R sequences. To generate a CRC-16 an Interrogator or Tag shall first generate the CRC-16 precursor shown in Table 6.14, then take the ones-complement of the generated precursor to form the CRC-16 (see also Annex F).

One sequence protected by a CRC-16 is the PC bits and EPC that a Tag backscatters during an inventory operation. Because Interrogators may issue a *Select* command that includes all or part of this CRC-16 in the mask, and may issue a *Read* command to cause the Tag to backscatter this CRC-16, this CRC-16 is logically mapped into EPC memory. At power-up a Tag shall compute this CRC-16 over EPC memory location 10_h to the end of the EPC (not necessarily to the end of EPC memory, but to the end of the EPC specified by the length field in the PC — see 6.3.2.1.4) and map the computed CRC-16 into EPC memory 00_h to $0F_h$, MSB first. Because the {PC+EPC} is stored in EPC memory on word boundaries, this CRC-16 shall be computed on word boundaries. Tags shall finish this CRC-16 computation and memory mapping by the end of interval T_s or T_{hs} (as appropriate) in Figure 6.3 or Figure 6.5, respectively. Tags shall not recalculate this CRC-16 for a truncated reply (see 6.3.2.10.1.1).

To facilitate message verification, an Interrogator or a Tag may add the CRC-16 to the transmitted message and recalculate the CRC-16. If the message is uncorrupted then the residue will be $1D0F_h$ (see Annex F).

Table 6.14 – CRC-16 precursor

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

6.3.2.1.4 Protocol-control (PC) bits

The PC bits contain physical-layer information that a Tag backscatters with its EPC during an inventory operation. There are 16 PC bits, stored in EPC memory at addresses 10_h to $1F_h$, with bit values defined as follows:

- Bits 10_h – 14_h: The length of the (PC + EPC) that a Tag backscatters, in words:
  - 00000_2: One word (addresses 10_h to $1F_h$ in EPC memory).
  - 00001_2: Two words (addresses 10_h to $2F_h$ in EPC memory).
  - 00010_2: Three words (addresses 10_h to $3F_h$ in EPC memory).
  - •
  - •
  - •
  - 11111_2: 32 words (addresses 10_h to $20F_h$ in EPC memory).
- Bits 15_h – 16_h: RFU (shall be set to 00_2 for Class-1 Tags).
- Bits 17_h – $1F_h$: A numbering system identifier (NSI) whose default value is 000000000_2. The MSB of the NSI is stored in memory location 17_h. If bit 17_h contains a logical 0, then PC bits 18_h – $1F_h$ shall contain an EPCglobal™ Header as defined in the EPC™ Tag Data Standards. If bit 17_h contains a logical 1, then PC bits 18_h – $1F_h$ shall contain the entire AFI defined in ISO/IEC 15961.

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

During truncated replies a Tag substitutes 00000_2 for the PC bits — see 6.3.2.10.1.1.

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

At power-up a Tag shall compute its CRC-16 over the number of (PC + EPC) words designated by the first 5 bits of the PC rather than over the length of the entire EPC memory (see 6.3.2.1.3).

6.3.2.1.5 EPC

An EPC is an electronic product code that identifies the object to which a Tag is affixed. The EPC is stored in EPC memory beginning at address 20_h, MSB first. Interrogators may issue a *Select* command that includes all or part of the EPC in the mask. Interrogators may issue an *ACK* command to cause a Tag to backscatter its PC, EPC, and CRC-16 (under certain circumstances the Tag may truncate its reply — see 6.3.2.10.1.1). Finally, an Interrogator may issue a *Read* command to read all or part of the EPC.

6.3.2.2 Sessions and inventoried flags

Interrogators shall support and Tags shall provide 4 sessions (denoted S0, S1, S2, and S3). Tags shall participate in one and only one session during an inventory round. Two or more Interrogators can use sessions to independently inventory a common Tag population. The sessions concept is illustrated in Figure 6.18.

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

Sessions allow Tags to associate a separate and independent **inventoried** flag to each of several readers.

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

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

- Interrogator #1 powers-on, then
  - It initiates an inventory round during which it singulates *A* Tags in session S2 to *B,*
  - It powers off.
- Interrogator #2 powers-on, then
  - It initiates an inventory round during which it singulates *B* Tags in session S3 to *A,*
  - It powers off.

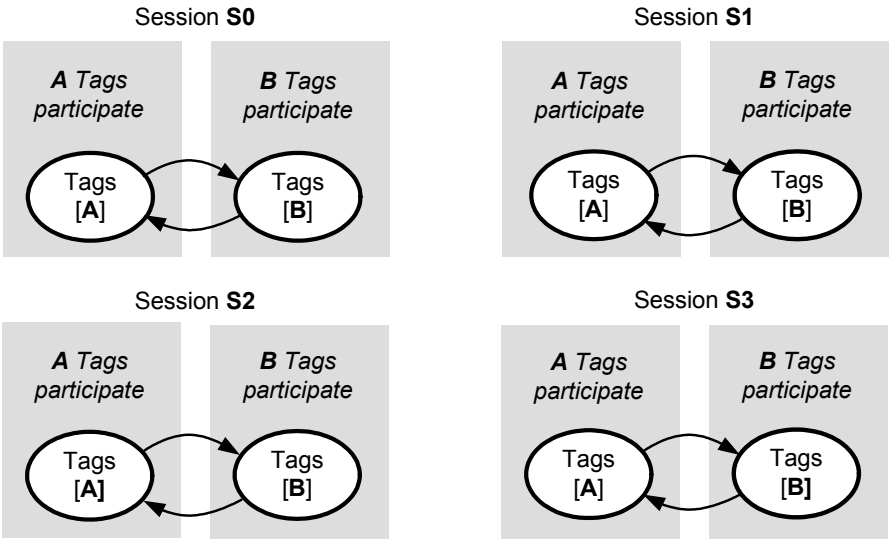


Figure 6.18 – Session diagram

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

A Tag's **inventoried** flags shall have the persistence times shown in Table 6.15. A Tag shall power-up with its **inventoried** flags set as follows:

- The S0 **inventoried** flag shall be set to *A*.
- The S1 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the flag was set longer in the past than its persistence time, in which case the Tag shall power-up with its S1 **inventoried** flag set to *A*. Because the S1 **inventoried** flag is not automatically refreshed, it may revert from *B* to *A* even when the Tag is powered.
- The S2 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the Tag has lost power for a time greater than its persistence time, in which case the Tag shall power-up with the S2 **inventoried** flag set to *A*.
- The S3 **inventoried** flag shall be set to either *A* or *B*, depending on its stored value, unless the Tag has lost power for a time greater than its persistence time, in which case the Tag shall power-up with its S3 **inventoried** flag set to *A*.

A Tag shall be capable of setting any of its inventoried flags to either *A* or *B* in 2 ms or less, regardless of the initial flag value. A Tag shall refresh its S2 and S3 flags while powered, meaning that every time a Tag loses power its S2 and S3 **inventoried** flags shall have the persistence times shown in Table 6.15. A Tag shall not let its S1 **inventoried** flag lose persistence while the Tag is participating in an inventory round. Instead, the Tag shall retain the flag value until the next *Query* command, at which point the flag may lose its persistence (unless the flag was refreshed during the round, in which case the flag shall assume its new value and new persistence).

6.3.2.3 Selected flag

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

A Tag's **SL** flag shall have the persistence times shown in Table 6.15. A Tag shall power-up with its **SL** flag either asserted or deasserted, depending on the stored value, unless the Tag has lost power for a time greater than the **SL** persistence time, in which case the Tag shall power-up with its **SL** flag deasserted (set to ~**SL**). A Tag shall be capable of asserting or deasserting its **SL** flag in 2 ms or less, regardless of the initial flag value. A Tag shall refresh its **SL** flag when powered, meaning that every time a Tag loses power its **SL** flag shall have the persistence times shown in Table 6.15.

Table 6.15 – Tag flags and persistence values

Flag	Required persistence
S0 **inventoried** flag	Tag energized: Indefinite Tag not energized: None
S1 **inventoried** flag[1]	Tag energized: Nominal temperature range: 500ms < persistence < 5s Extended temperature range: Not specified Tag not energized: Nominal temperature range: 500ms < persistence < 5s Extended temperature range: Not specified
S2 **inventoried** flag[1]	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified
S3 **inventoried** flag[1]	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified
Selected (**SL**) flag[1]	Tag energized: Indefinite Tag not energized: Nominal temperature range: 2s < persistence Extended temperature range: Not specified

Note 1: For a randomly chosen and sufficiently large Tag population, 95% of the Tag persistence times shall meet the persistence requirement, with a 90% confidence interval.

6.3.2.4 Tag states and slot counter

Tags shall implement the states and the slot counter shown in Figure 6.19. Annex B shows the associated state-transition tables; Annex C shows the associated command-response tables.

6.3.2.4.1 Ready state

Tags shall implement a **ready** state. **Ready** can be viewed as a "holding state" for energized Tags that are neither killed nor currently participating in an inventory round. Upon entering an energizing RF field a Tag that is not killed shall enter **ready**. The Tag shall remain in **ready** until it receives a *Query* command (see 6.3.2.10.2.1) whose inventoried parameter (for the session specified in the *Query*) and sel parameter match its current flag values. Matching Tags shall draw a *Q*-bit number from their RNG (see 6.3.2.5), load this number into their slot counter, and transition to the **arbitrate** state if the number is nonzero, or to the **reply** state if the number is zero. If a Tag in any state except **killed** loses power it shall return to **ready** upon regaining power.

6.3.2.4.2 Arbitrate state

Tags shall implement an **arbitrate** state. **Arbitrate** can be viewed as a "holding state" for Tags that are participating in the current inventory round but whose slot counters (see 6.3.2.4.8) hold nonzero values. A Tag in **arbitrate** shall decrement its slot counter every time it receives a *QueryRep* command (see 6.3.2.10.2.3) whose session parameter matches the session for the inventory round currently in progress, and it shall transition to the **reply** state when its slot counter reaches 0000_h. Tags that return to **arbitrate** (for example, from the **reply** state) with a slot value of 0000_h shall decrement their slot counter from 0000_h to $7FFF_h$ at the next *QueryRep* (with matching session) and, because their slot value is now nonzero, shall remain in **arbitrate**.

6.3.2.4.3 Reply state

Tags shall implement a **reply** state. Upon entering **reply** a Tag shall backscatter an RN16. If the Tag receives a valid acknowledgement (*ACK*) it shall transition to the **acknowledged** state, backscattering its PC, EPC and CRC-16. If the Tag fails to receive an *ACK*, or receives an invalid *ACK*, it shall return to **arbitrate**. Tag and Interrogator shall meet all timing requirements specified in Table 6.13.

6.3.2.4.4 Acknowledged state

Tags shall implement an **acknowledged** state. A Tag in **acknowledged** may transition to any state except **killed**,

depending on the received command (see Figure 6.19). Tag and Interrogator shall meet all timing requirements specified in Table 6.13.

6.3.2.4.5 Open state

Tags shall implement an **open** state. A Tag in the **acknowledged** state whose access password is nonzero shall transition to **open** upon receiving a *Req_RN* command, backscattering a new RN16 (denoted handle) that the Interrogator shall use in subsequent commands and the Tag shall use in subsequent replies. Tags in the **open** state can execute all access commands except *Lock*. A Tag in **open** may transition to any state except **acknowledged**, depending on the received command (see Figure 6.19). Tag and Interrogator shall meet all timing requirements specified in Table 6.13 except $T_{2(max)}$; in the **open** state the maximum delay between Tag response and Interrogator transmission is unrestricted.

6.3.2.4.6 Secured state

Tags shall implement a **secured** state. A Tag in the **acknowledged** state whose access password is zero shall transition to **secured** upon receiving a *Req_RN* command, backscattering a new RN16 (denoted handle) that the Interrogator shall use in subsequent commands and the Tag shall use in subsequent replies. A Tag in the **open** state whose access password is nonzero shall transition to **secured** upon receiving a valid *Access* command, maintaining the same handle that it previously backscattered when it transitioned from the **acknowledged** to the **open** state. Tags in the **secured** state can execute all access commands. A Tag in **secured** may transition to any state except **open** or **acknowledged**, depending on the received command (see Figure 6.19). Tag and Interrogator shall meet all timing requirements specified in Table 6.13 except $T_{2(max)}$; in the **secured** state the maximum delay between Tag response and Interrogator transmission is unrestricted.

6.3.2.4.7 Killed state

Tags shall implement a **killed** state. A Tag in either the **open** or **secured** states shall enter the **killed** state upon receiving a *Kill* command (see 6.3.2.10.3.4) with a valid nonzero kill password and valid handle. *Kill* permanently disables a Tag. Upon entering the **killed** state a Tag shall notify the Interrogator that the kill operation was successful, and shall not respond to an Interrogator thereafter. Killed Tags shall remain in the **killed** state under all circumstances, and shall immediately enter killed upon subsequent power-ups. A kill operation is not reversible.

6.3.2.4.8 Slot counter

Tags shall implement a 15-bit slot counter. Upon receiving a *Query* or *QueryAdjust* command a Tag shall preload a value between 0 and 2^Q-1, drawn from the Tag's RNG (see 6.3.2.5), into its slot counter. *Q* is an integer in the range (0,15). A *Query* specifies *Q*; a *QueryAdjust* may modify *Q* from the prior *Query*. Upon receiving a *QueryRep* command a Tag shall decrement its slot counter. The slot counter shall be capable of continuous counting, meaning that, after the slot counter decrements to 0000_h it shall roll over and begin counting down from $7FFF_h$. See also Annex J.

6.3.2.5 Tag random or pseudo-random number generator

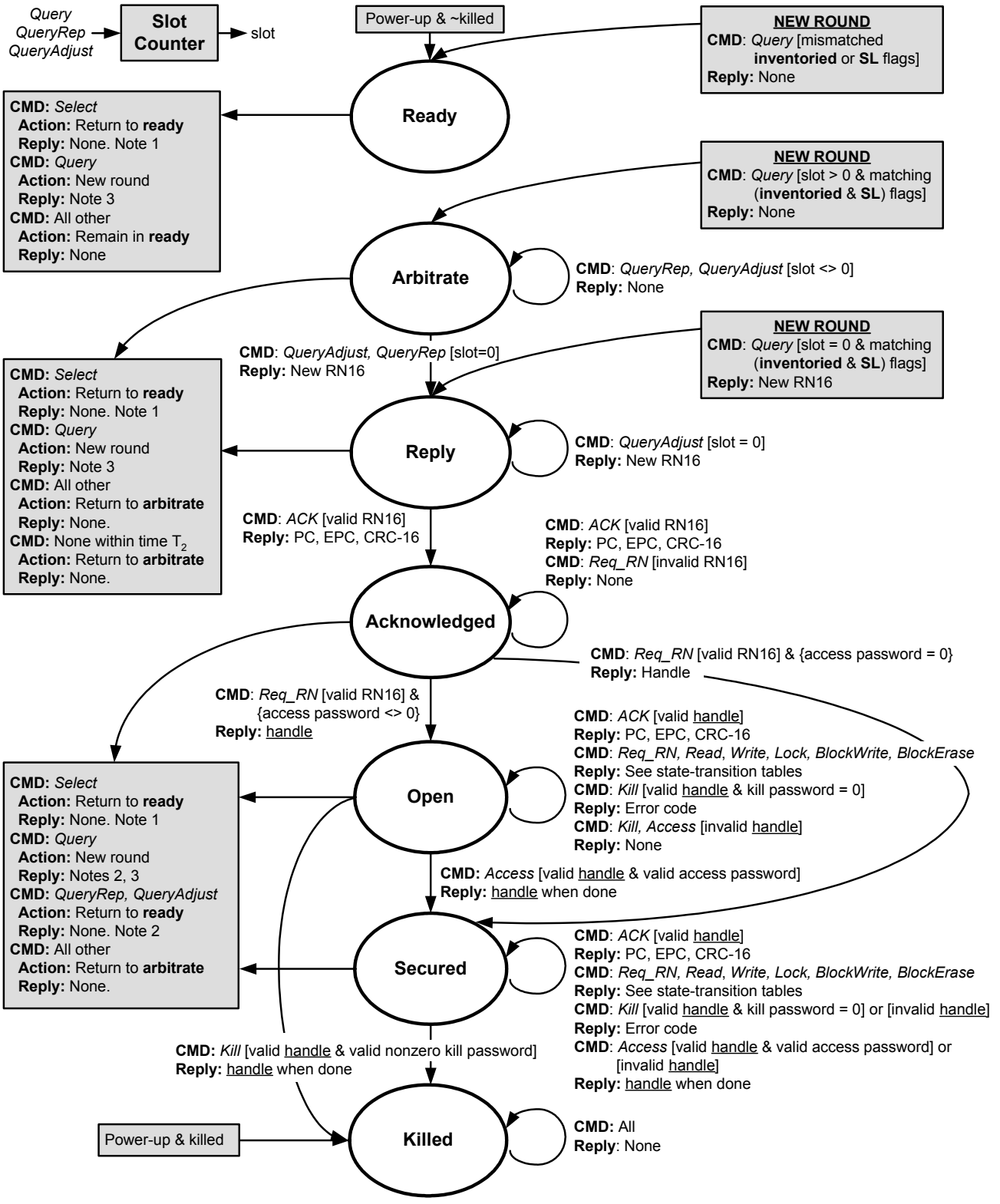
Tags shall implement a random or pseudo-random number generator (RNG). The RNG shall meet the following randomness criteria independent of the strength of the energizing field, the R=>T link rate, and the data stored in the Tag (including the PC, EPC, and CRC-16). Tags shall generate 16-bit random or pseudo-random numbers (RN16) using the RNG, and shall have the ability to extract *Q*-bit subsets from an RN16 to preload the Tag's slot counter (see 6.3.2.4.8). Tags shall have the ability to temporarily store at least two RN16s while powered, to use, for example, as a handle and a 16-bit cover-code during password transactions (see Figure 6.23 or Figure 6.25).

Probability of a single RN16: The probability that any RN16 drawn from the RNG has value RN16 = *j*, for any *j*, shall be bounded by $0.8/2^{16} < P(RN16 = j) < 1.25/2^{16}$.

Probability of simultaneously identical sequences: For a Tag population of up to 10,000 Tags, the probability that any two or more Tags simultaneously generate the same sequence of RN16s shall be less than 0.1%, regardless of when the Tags are energized.

Probability of predicting an RN16: An RN16 drawn from a Tag's RNG 10ms after the end of T_r in Figure 6.3 shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from the RNG, performed under identical conditions, are known.

This protocol recommends that Interrogators wait 10ms after T_r in Figure 6.3 or T_{hr} in Figure 6.5 before issuing passwords to Tags.



NOTES

1. *Select*: Assert/deassert **SL** or set **inventoried** to *A* or *B*.
2. *Query*: *A*→ *B* or *B*→ *A* if the new session matches the prior session; otherwise no change to the **inventoried** flag.
   QueryRep/QueryAdjust: *A*→ *B* or *B*→ *A* if the session matches the prior *Query*; otherwise, the command is invalid and ignored by the Tag.
3. *Query* starts a new round and may change the session. Tags may go to **ready**, **arbitrate**, or **reply**.

Figure 6.19 – Tag state diagram

6.3.2.6 Managing Tag populations

Interrogators manage Tag populations using the three basic operations shown in Figure 6.20. Each of these operations comprises one or more commands. The operations are defined as follows:

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

b) **Inventory:** The process by which an Interrogator identifies Tags. An Interrogator begins an inventory round by transmitting a *Query* command in one of four sessions. One or more Tags may reply. The Interrogator detects a single Tag reply and requests the PC, EPC, and CRC-16 from the Tag. An inventory round operates in one and only one session at a time. Annex E shows an example of an Interrogator inventorying and accessing a single Tag.

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

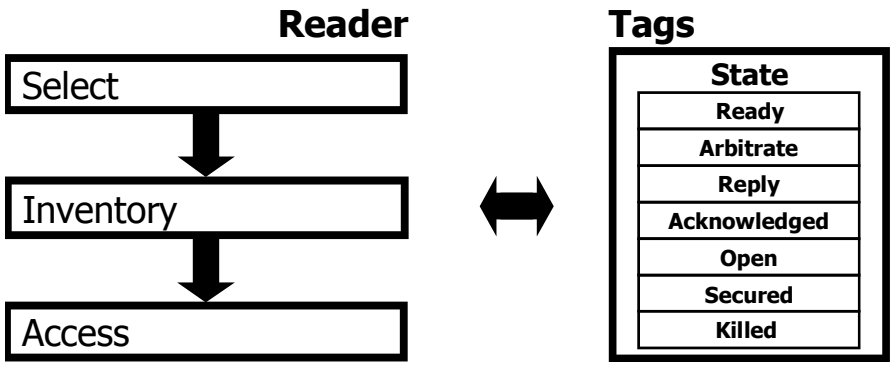


Figure 6.20 – Interrogator/Tag operations and Tag state

6.3.2.7 Selecting Tag populations

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

- Target and Action indicate whether and how a *Select* modifies a Tag's **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. A *Select* that modifies **SL** shall not modify **inventoried**, and vice versa.
- MemBank specifies if the mask applies to EPC, TID, or User memory. *Select* commands apply to a single memory bank. Successive *Selects* may apply to different memory banks.
- Pointer, Length, and Mask: Pointer and Length describe a memory range. Mask, which must be Length bits long, contains a bit string that a Tag compares against the specified memory range.
- Truncate specifies whether a Tag backscatters its entire EPC, or only that portion of the EPC immediately following Mask. Truncated replies are always followed by the CRC-16 in EPC memory 00_h to $0F_h$; a Tag does not recompute this CRC for a truncated reply.

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

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

6.3.2.8 Inventorying Tag populations

The inventory command set includes *Query*, *QueryAdjust*, *QueryRep*, *ACK*, and *NAK*. *Query* initiates an inventory round and decides which Tags participate in the round (where "inventory round" is defined as the period between successive *Query* commands).

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

a) The Tag backscatters an RN16 as it enters **reply**,
b) The Interrogator acknowledges the Tag with an *ACK* containing this same RN16,
c) The acknowledged Tag transitions to the **acknowledged** state, backscattering its PC, EPC, and CRC-16,
d) The Interrogator issues a *QueryAdjust* or *QueryRep,* causing the identified Tag to invert its **inventoried** flag (i.e. *A*→*B* or *B*→*A*) and transition to **ready**, and potentially causing another Tag to initiate a query-response dialog with the Interrogator, starting in step (a), above.

If the Tag fails to receive the *ACK* in step (b) within time T_2 (see Figure 6.16), or receives the *ACK* with an erroneous RN16, it shall return to **arbitrate**.

If multiple Tags reply in step (a) but the Interrogator, by detecting and resolving collisions at the waveform level, can resolve an RN16 from one of the Tags, the Interrogator can *ACK* the resolved Tag. Unresolved Tags receive erroneous RN16s and return to **arbitrate** without backscattering their PC, EPC, and CRC-16.

If the Interrogator sends a valid *ACK* (i.e. an *ACK* containing the correct RN16) to the Tag in the **acknowledged** state the Tag shall re-backscatter its PC, EPC, and CRC-16.

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

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

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

Tags in the **arbitrate** state decrement their slot counter every time they receive a *QueryRep*, transitioning to the **reply** state and backscattering an RN16 when their slot counter reaches 0000_h. Tags whose slot counter reached 0000_h, who replied, and who were not acknowledged (including Tags that responded to the original *Query* and were not acknowledged) shall return to **arbitrate** with a slot value of 0000_h and shall decrement this slot value from 0000_h to $7FFF_h$ at the next *QueryRep*, thereby effectively preventing subsequent replies until the Tag loads a new random value into its slot counter. Tags shall reply at least once in 2^Q-1 *QueryRep* commands.

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

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

The scenario outlined above assumed a single Interrogator operating in a single session. However, as described in 6.3.2.2, an Interrogator can inventory a Tag population in one of four sessions. Furthermore, as described in 6.3.2.10.2, the *Query*, *QueryAdjust*, and *QueryRep* commands each contain a session parameter. How a Tag responds to these commands varies with the command, session parameter, and Tag state, as follows:

- *Query*: A *Query* command starts an inventory round and chooses the session for the round. Tags in any state except **killed** shall execute a *Query*, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure 6.19).
    - If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter matches the prior session it shall invert its **inventoried** flag (i.e. *A*→*B* or *B*→*A*) for the session before it evaluates whether to transition to **ready**, **arbitrate**, or **reply**.
    - If a Tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter does not match the prior session it shall leave its **inventoried** flag for the prior session unchanged as it evaluates whether to transition to **ready**, **arbitrate**, or **reply**.

- *QueryAdjust, QueryRep*: Tags in any state except **ready** or **killed** shall execute a *QueryAdjust* or *QueryRep* command if, and only if, the session parameter in the command matches the session parameter in the *Query* that started the round. Tags shall ignore a *QueryAdjust* or *QueryRep* with mismatched session.
    - If a Tag in the **acknowledged**, **open**, or **secured** states receives a *QueryAdjust* or *QueryRep* whose session parameter matches the session parameter in the prior *Query*, it shall invert its **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$) for the current session then transition to **ready**.

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

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

b) **One Tag replies** (see Figure 6.21): The Tag transitions to the **reply** state and backscatters an RN16. The Interrogator acknowledges the Tag by sending an *ACK*. If the Tag receives the *ACK* with a correct RN16 it backscatters its PC, EPC, and CRC-16 and transitions to the **acknowledged** state. If the Tag receives the *ACK* with an incorrect RN16 it transitions to **arbitrate**. Assuming a successful *ACK*, the Interrogator may either access the acknowledged Tag or issue a *QueryAdjust* or *QueryRep* to invert the Tag's **inventoried** flag from $A \rightarrow B$ and send the Tag to **ready** (a *Query* with matching prior-round session parameter will also invert the **inventoried** flag from $A \rightarrow B$).

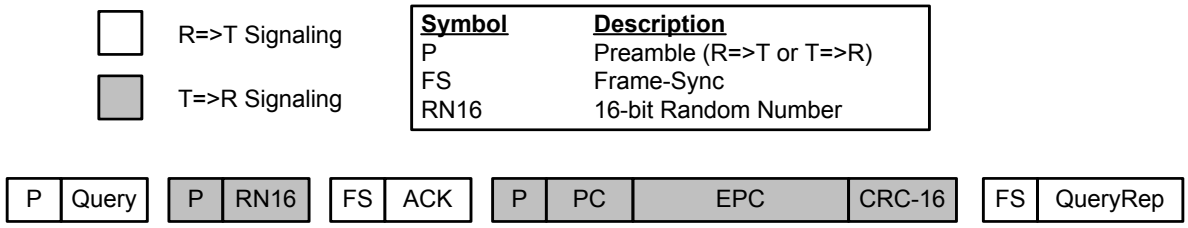


Figure 6.21 – One Tag reply

c) **Multiple Tags reply:** The Interrogator observes a backscattered waveform comprising multiple RN16s. It may try to resolve the collision and issue an *ACK*; not resolve the collision and issue a *QueryAdjust, QueryRep*, or *NAK*; or quickly identify the collision and issue a *QueryAdjust* or *QueryRep* before the collided Tags have finished backscattering. In the latter case the collided Tags, not observing a valid reply within T_2 (see Figure 6.16), shall return to **arbitrate** and await the next *Query* or *QueryAdjust* command.

6.3.2.9 Accessing individual Tags

After acknowledging a Tag, an Interrogator may choose to access it. The access command set comprises *Req_RN, Read, Write*, *Kill*, *Lock*, *Access*, *BlockWrite*, and *BlockErase*. Tags execute *Req_RN* from the **acknowledged**, **open**, or **secured** states. Tags execute *Read*, *Write*, *Kill*, *Access, BlockWrite*, and *BlockErase* from the **open** or **secured** states. Tags execute *Lock* only from the **secured** state.

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

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

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

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

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

An Interrogator may issue an *ACK* to a Tag in the **open** or **secured** states, causing the Tag to backscatter its PC, EPC, and CRC-16.

Interrogator and Tag can communicate indefinitely in the **open** or **secured** states. The Interrogator may terminate the communications at any time by issuing a *Query*, *QueryAdjust*, *QueryRep*, or a *NAK*. The Tag's response to a *Query*, *QueryAdjust*, or *QueryRep* is described in 6.3.2.8. A *NAK* causes all Tags in the inventory round to return to **arbitrate** without changing their **inventoried** flag.

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

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

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

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

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

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

The *BlockWrite* command (see 6.3.2.10.3.7) communicates multiple 16-bit words from Interrogator to Tag. Unlike *Write*, *BlockWrite* does not use link cover-coding.

A Tag responds to a command that writes or erases memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite*, and *BlockErase*) by backscattering its handle, indicating that the operation was successful, or by backscattering an error code (see Annex I), indicating that the operation was unsuccessful.

Killing a Tag is a multi-step procedure, described in 6.3.2.10.3.4 and outlined in Figure 6.23.

Issuing an access password to a Tag is a multi-step procedure, described in 6.3.2.10.3.6 and outlined in Figure 6.25.

Tag memory may be unlocked or locked. The lock status may be changeable or permalocked (i.e. permanently unlocked or permanently locked). An Interrogator may write to unlocked memory from either the **open** or **secured** states. An Interrogator may write to locked memory that is not permalocked from the **secured** state only. See Table 6.40 for a detailed description of memory lock, permalock, and the Tag state required to modify memory.

Interrogator and Tag shall transmit all strings MSB first.

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

6.3.2.10 Interrogator commands and Tag replies

Interrogator-to-Tag commands shall have the format shown in Table 6.16.

- *QueryRep* and *ACK* have 2-bit command codes beginning with 0_2.
- *Query*, *QueryAdjust*, and *Select* have 4-bit command codes beginning with 10_2.
- All other base commands have 8-bit command codes beginning with 110_2.
- All extended commands have 16-bit command codes beginning with 1110_2.
- *QueryRep*, *ACK*, *Query*, *QueryAdjust,* and *NAK* have the unique command lengths shown in Table 6.16. No other commands shall have these lengths. If a Tag receives one of these commands with an incorrect length it shall ignore the command.
- *Query, QueryAdjust*, and *QueryRep* contain a session parameter.
- *Query* is protected by a CRC-5, shown in Table 6.17 and detailed in Annex F.

- *Select*, *Req_RN*, *Read*, *Write*, *Kill*, *Lock*, *Access*, *BlockWrite* and *BlockErase* are protected by a CRC-16, defined in 6.3.2.1.3 and detailed in Annex F.
- R=>T commands begin with either a preamble or a frame-sync, as described in 6.3.1.2.8. The command-code lengths specified in Table 6.16 do not include the preamble or frame-sync.
- Tags shall ignore invalid commands. In general, "invalid" means a command that (1) is incorrect given the current Tag state, (2) is unsupported by the Tag, (3) has incorrect parameters, (4) has a CRC error, (5) specifies an incorrect session, or (6) is in any other way not recognized or not executable by the Tag. The actual definition of "invalid" is state-specific and defined, for each Tag state, in Annex B and Annex C.

Table 6.16 – Commands

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

Table 6.17 – CRC-5 definition. See also Annex F

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

6.3.2.10.1 Select commands

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

6.3.2.10.1.1 *Select* (mandatory)

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

Interrogators and Tags shall implement the *Select* command shown in Table 6.18. Target shall indicate whether the *Select* modifies a Tag's **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. Action shall elicit the Tag response shown in Table 6.19. The criteria for determining whether a Tag is matching or non-matching are specified in the MemBank, Pointer, Length and Mask fields. Truncate indicates whether a Tag's backscattered reply shall be truncated to include only those EPC and CRC-16 bits following Mask. *Select* passes the following parameters from Interrogator to Tags:

- Target indicates whether the *Select* command modifies a Tag's **SL** flag or its **inventoried** flag, and in the case of **inventoried** it further specifies one of four sessions. A *Select* command that modifies **SL** does not modify **inventoried**, and vice versa. Class-1 Tags shall ignore *Select* commands whose Target is 101_2, 110_2, or 111_2.
- Action indicates whether matching Tags assert or deassert **SL**, or set their **inventoried** flag to *A* or to *B*. Tags conforming to the contents of the MemBank, Pointer, Length, and Mask fields are considered matching. Tags not conforming to the contents of these fields are considered non-matching.
- MemBank specifies whether Mask applies to EPC, TID, or User memory. *Select* commands shall apply to a single memory bank. Successive *Selects* may apply to different banks. MemBank shall not specify Reserved memory; if a Tag receives a *Select* specifying MemBank = 00_2 it shall ignore the *Select*. MemBank parameter value 00_2 is reserved for future use (RFU).
- Pointer, Length, and Mask: Pointer and Length describe a memory range. Pointer references a memory bit address (Pointer is not restricted to word boundaries) and uses EBV formatting (see Annex A). Length is 8 bits, allowing Masks from 0 to 255 bits in length. Mask, which is Length bits long, contains a bit string that a Tag compares against the memory location that begins at Pointer and ends Length bits later. If Pointer and Length reference a memory location that does not exist on the Tag then the Tag shall consider the *Select* to be non-matching. If Length is zero then all Tags shall be considered matching, unless Pointer references a memory location that does not exist on the Tag, in which case the Tag shall consider the *Select* to be non-matching.
- Truncate: If an Interrogator asserts Truncate, and if a subsequent *Query* specifies Sel=10 or Sel=11, then matching Tags shall truncate their reply to an *ACK* to that portion of the EPC immediately following Mask, followed by the CRC-16 stored in EPC memory 00_h to $0F_h$. Interrogators shall assert Truncate:
  - in the last (and only in the last) *Select* that the Interrogator issues prior to sending a *Query*,
  - if and only if the *Select* has Target = 100_2, and
  - if and only if Mask ends in the EPC.

  These constraints *do not* preclude an Interrogator from issuing multiple *Select* commands that target the **SL** and/or **inventoried** flags. They *do* require that an Interrogator assert Truncate only in the last *Select*, and only if this last *Select* targets the **SL** flag. Tags shall power-up with Truncate deasserted.

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

  Mask may end at the last bit of the EPC, in which case a selected Tag shall backscatter its CRC-16.

  Truncated replies never include PC bits, because Mask must end in the EPC.

  A Tag shall preface its truncated reply with five leading zeros (00000_2) inserted between the preamble and the truncated reply. A Tag does not recalculate the CRC-16 for a truncated reply.

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

Interrogators shall prepend a *Select* with a frame-sync (see 6.3.1.2.8). The CRC-16 is calculated over the first command-code bit to the Truncate bit.

Tags shall not reply to a *Select*.

Table 6.18 – *Select* command

	Command	Target	Action	MemBank	Pointer	Length	Mask	Truncate	CRC-16
# of bits	4	3	3	2	EBV	8	Variable	1	16
description	1010	000: **Inventoried** (S0) 001: **Inventoried** (S1) 010: **Inventoried** (S2) 011: **Inventoried** (S3) 100: **SL** 101: RFU 110: RFU 111: RFU	See Table 6.19	00: RFU 01: EPC 10: TID 11: User	Starting Mask address	Mask length (bits)	Mask value	0: Disable truncation 1: Enable truncation	

Table 6.19 – Tag response to Action parameter

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

6.3.2.10.2 Inventory commands

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

6.3.2.10.2.1 *Query* (mandatory)

Interrogators and Tags shall implement the *Query* command shown in Table 6.20. *Query* initiates and specifies an inventory round. *Query* includes the following fields:

- DR (TRcal divide ratio) sets the T=>R link frequency as described in 6.3.1.2.8 and Table 6.11.
- M (cycles per symbol) sets the T=>R data rate and modulation format as shown in Table 6.12.
- TRext chooses whether the T=>R preamble is prepended with a pilot tone as described in 6.3.1.3.2.2 and 6.3.1.3.2.4.
- Sel chooses which Tags respond to the *Query* (see 6.3.2.10.1.1 and 6.3.2.8).
- Session chooses a session for the inventory round (see 6.3.2.8).
- Target selects whether Tags whose **inventoried** flag is *A* or *B* participate in the inventory round. Tags may change their inventoried flag from *A* to *B* (or vice versa) as a result of being singulated.
- Q sets the number of slots in the round (see 6.3.2.8).

Interrogators shall prepend a *Query* with a preamble (see 6.3.1.2.8).

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

Upon receiving a *Query*, Tags with matching Sel and Target pick a random value in the range (0, $2^{Q}-1$), inclusive, and load this value into their slot counter. If a Tag, in response to the *Query*, loads its slot counter with zero, then its reply to a *Query* shall be as shown in Table 6.21; otherwise the Tag shall remain silent.

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

Tags shall support all DR and M values specified in Table 6.11 and Table 6.12, respectively.

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

Table 6.20 – *Query* command

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

Table 6.21 – Tag reply to a *Query* command

	Response
# of bits	16
description	RN16

6.3.2.10.2.2 *QueryAdjust* (mandatory)

Interrogators and Tags shall implement the *QueryAdjust* command shown in Table 6.22. *QueryAdjust* adjusts *Q* (i.e. the number of slots in an inventory round – see 6.3.2.8) without changing any other round parameters.

QueryAdjust includes the following fields:

- Session corroborates the session number for the inventory round (see 6.3.2.8 and 6.3.2.10.2.1). If a Tag receives a *QueryAdjust* whose session number is different from the session number in the *Query* that initiated the round it shall ignore the command.
- UpDn determines whether and how the Tag adjusts *Q*, as follows:

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

  000: No change to *Q*.

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

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

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

A *QueryAdjust* shall be prepended with a frame-sync (see 6.3.1.2.8).

Upon receiving a *QueryAdjust* Tags first update *Q*, then pick a random value in the range $(0, 2^Q - 1)$, inclusive, and load this value into their slot counter. If a Tag, in response to the *QueryAdjust*, loads its slot counter with zero, then its reply to a *QueryAdjust* shall be shown in Table 6.23; otherwise, the Tag shall remain silent. Tags shall respond to a *QueryAdjust* only if they received a prior *Query*.

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryAdjust* invert their **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$, as appropriate) for the current session and transition to **ready**.

Table 6.22 – *QueryAdjust* command

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

Table 6.23 – Tag reply to a *QueryAdjust* command

	Response
# of bits	16
description	RN16

6.3.2.10.2.3 *QueryRep* (mandatory)

Interrogators and Tags shall implement the *QueryRep* command shown in Table 6.24. *QueryRep* instructs Tags to decrement their slot counters and, if slot = 0 after decrementing, to backscatter an RN16 to the Interrogator.

QueryRep includes the following field:

- Session corroborates the session number for the inventory round (see 6.3.2.8 and 6.3.2.10.2.1). If a Tag receives a *QueryRep* whose session number is different from the session number in the *Query* that initiated the round it shall ignore the command.

A *QueryRep* shall be prepended with a frame-sync (see 6.3.1.2.8).

If a Tag, in response to the *QueryRep*, decrements its slot counter and the decremented slot value is zero, then its reply to a *QueryRep* shall be as shown in Table 6.25; otherwise the Tag shall remain silent. Tags shall respond to a *QueryRep* only if they received a prior *Query*.

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryRep* invert their **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$, as appropriate) for the current session and transition to **ready**.

Table 6.24 – *QueryRep* command

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

Table 6.25 – Tag reply to a *QueryRep* command

	Response
# of bits	16
description	RN16

6.3.2.10.2.4 *ACK* (mandatory)

Interrogators and Tags shall implement the *ACK* command shown in Table 6.26. An Interrogator sends an *ACK* to acknowledge a single Tag. *ACK* echoes the Tag's backscattered RN16.

If an Interrogator issues an *ACK* to a Tag in the **reply** or **acknowledged** states, then the echoed RN16 shall be the RN16 that the Tag previously backscattered as it transitioned from the **arbitrate** state to the **reply** state. If an Interrogator issues an *ACK* to a Tag in the **open** or **secured** states, then the echoed RN16 shall be the Tag's handle (see 6.3.2.10.3.1).

An *ACK* shall be prepended with a frame-sync (see 6.3.1.2.8).

The Tag reply to a successful *ACK* shall be as shown in Table 6.27. As described in 6.3.2.10.1.1, the reply may be truncated. A Tag that receives an *ACK* with an incorrect RN16 or an incorrect handle (as appropriate) shall return to **arbitrate** without responding, unless the Tag is in **ready** or **killed**, in which case it shall ignore the *ACK* and remain in its present state.

Table 6.26 – *ACK* command

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

Table 6.27 – Tag reply to a successful *ACK* command

	Response
# of bits	21 to 528
description	{PC, EPC, CRC-16} OR {00000_2, truncated EPC, CRC-16}

6.3.2.10.2.5 *NAK* (mandatory)

Interrogators and Tags shall implement the *NAK* command shown in Table 6.28. *NAK* shall return all Tags to the **arbitrate** state unless they are in **ready** or **killed**, in which case they shall ignore the *NAK* and remain in their current state..

A *NAK* shall be prepended with a frame-sync (see 6.3.1.2.8).

Tags shall not reply to a *NAK*.

Table 6.28 – *NAK* command

	Command
# of bits	8
description	11000000

6.3.2.10.3 Access commands

The set of access commands comprises *Req_RN, Read, Write*, *Kill*, *Lock*, *Access*, *BlockWrite*, and *BlockErase*. As described in 6.3.2.9, Tags execute *Req_RN* from the **acknowledged**, **open**, or **secured** states. Tags execute *Read*, *Write*, *BlockWrite*, and *BlockErase* from the **secured** state; if allowed by the lock status of the memory location being accessed, they may also execute these commands from the **open** state. Tags execute *Access* and *Kill* from the **open** or **secured** states. Tags execute *Lock* only from the **secured** state.

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

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

A Tag's reply to all access commands that write memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite*, and *BlockErase*) shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. the Tag shall reply as if TRext=1 regardless of the TRext value specified in the *Query* command that initiated the inventory round).

After an Interrogator writes part or all of a Tag's PC or EPC, the CRC-16 stored in Tag EPC memory 00_h to $0F_h$ will not be valid until the Interrogator first powers-down and then re-powers-up its energizing RF field.

Req_RN, Read, Write, *Kill*, and *Lock* are required commands; *Access*, *BlockWrite*, and *BlockErase* are optional. Tags that receive an optional access command that they do not support shall ignore the command.

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

6.3.2.10.3.1 *Req_RN* (mandatory)

Interrogators and Tags shall implement the *Req_RN* command shown in Table 6.29. *Req_RN* instructs a Tag to backscatter a new RN16. Both the Interrogator's command, and the Tag's response, depend on the Tag's state:

- **Acknowledged** state: When issuing a *Req_RN* command to a Tag in the **acknowledged** state, an Interrogator shall include the Tag's last backscattered RN16 as a parameter in the *Req_RN*. The *Req_RN* command is protected by a CRC-16 calculated over the command bits and the RN16. If the Tag receives the *Req_RN* with a valid CRC-16 and a valid RN16 it shall generate and store a new RN16 (denoted handle), backscatter this handle, and transition to the **open** or **secured** state. The choice of ending state depends on the Tag's access password, as follows:
  - Access password <> 0: Tag transitions to **open** state.
  - Access password = 0: Tag transitions to **secured** state.

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

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

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

A *Req_RN* shall be prepended with a frame-sync (see 6.3.1.2.8).

The Tag reply to a *Req_RN* shall be as shown in Table 6.30. The RN16 and handle are protected by a CRC-16.

Table 6.29 – *Req_RN* command

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

Table 6.30 – Tag reply to a *Req_RN* command

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

6.3.2.10.3.2 *Read* (mandatory)

Interrogators and Tags shall implement the *Read* command shown in Table 6.31. *Read* allows an Interrogator to read part or all of a Tag's Reserved, EPC, TID, or User memory. *Read* has the following fields:

- MemBank specifies whether the *Read* accesses Reserved, EPC, TID, or User memory. *Read* commands shall apply to a single memory bank. Successive *Reads* may apply to different banks.
- WordPtr specifies the starting word address for the memory read, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).
- WordCount specifies the number of 16-bit words to be read. If WordCount = 00_h the Tag shall backscatter the contents of the chosen memory bank starting at WordPtr and ending at the end of the bank, unless MemBank = 01, in which case the Tag shall backscatter the EPC memory contents starting at WordPtr and ending at the length of the EPC specified by the first 5 bits of the PC if WordPtr lies within the EPC, and shall backscatter the EPC memory contents starting at WordPtr and ending at the end of EPC memory if WordPtr lies outside the EPC.

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

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

A *Read* shall be prepended with a frame-sync (see 6.3.1.2.8).

If all of the memory words specified in a *Read* exist and none are read-locked, the Tag reply to the *Read* shall be as shown in Table 6.32. The Tag responds by backscattering a header (a 0-bit), the requested memory words, and its handle. The reply includes a CRC-16 calculated over the 0-bit, memory words, and handle.

If a one or more of the memory words specified in the *Read* command either do not exist or are read-locked, the Tag shall backscatter an error code, within time T_1 in Table 6.13, rather than the reply shown in Table 6.32 (see Annex I for error-code definitions and for the reply format).

Table 6.31 – *Read* command

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

Table 6.32 – Tag reply to a successful *Read* command

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

6.3.2.10.3.3 *Write* (mandatory)

Interrogators and Tags shall implement the *Write* command shown in Table 6.33. *Write* allows an Interrogator to write a word in a Tag's Reserved, EPC, TID, or User memory. *Write* has the following fields:

- MemBank specifies whether the *Write* occurs in Reserved, EPC, TID, or User memory.
- WordPtr specifies the word address for the memory write, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).
- Data contains a 16-bit word to be written. Before each and every *Write* the Interrogator shall first issue a *Req_RN* command; the Tag responds by backscattering a new RN16. The Interrogator shall cover-code the data by EXORing it with this new RN16 prior to transmission.

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

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

A *Write* shall be prepended with a frame-sync (see 6.3.1.2.8).

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

- **The *Write* succeeds:** After completing the *Write* a Tag shall backscatter the reply shown in Table 6.34 and Figure 6.22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *Write* completed successfully.
- **The Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 6.34 (see Annex I for error-code definitions and for the reply format).
- **The *Write* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *Write* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *Write* command.

Upon receiving a valid *Write* command a Tag shall write the commanded Data into memory. The Tag's reply to a successful *Write* shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 6.33 – *Write* command

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

Table 6.34 – Tag reply to a successful *Write* command

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

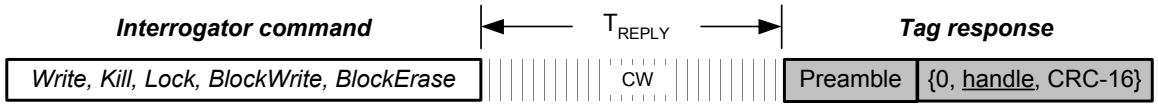


Figure 6.22 – Successful *Write* sequence

6.3.2.10.3.4 *Kill* (mandatory)

Interrogators and Tags shall implement the *Kill* command shown in Table 6.35. *Kill* allows an Interrogator to permanently disable a Tag.

Kill contains 3 RFU bits. When communicating with Class-1 Tags, Interrogators shall set these bits to 000_2. Class-1 Tags shall ignore these bits. Higher-functionality Tags may use these bits to expand the functionality of the *Kill* command (for example, to kill a Tag to a recycled state rather than killing it dead).

To kill a Tag, an Interrogator shall follow the multi-step kill procedure outlined in Figure 6.23. Briefly, an Interrogator issues two *Kill* commands, the first containing the 16 MSBs of the Tag's kill password EXORed with an RN16, and the second containing the 16 LSBs of the Tag's kill password EXORed with a different RN16. Each EXOR operation shall be performed MSB first (i.e. the MSB of each half-password shall be EXORed with the MSB of its respective RN16). Just prior to issuing each *Kill* the Interrogator first issues a *Req_RN* to obtain a new RN16.

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

Tag's whose kill password is zero do not execute a kill operation; if such a Tag receives a *Kill* it ignores the command and backscatters an error code (see Figure 6.23).

The Tag reply to the first *Kill* shall be as shown in Table 6.36. The reply shall use the TRext value specified in the *Query* command that initiated the round.

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

- **The *Kill* succeeds:** After completing the *Kill* the Tag shall backscatter the reply shown in Table 6.37 and Figure 6.22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. Immediately after this reply the Tag shall render itself silent and shall not respond to an Interrogator thereafter. If the Interrogator observes this reply within 20 ms then the *Kill* completed successfully.
- **The Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 6.37 (see Annex I for error-code definitions and for the reply format).
- **The *Kill* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *Kill* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reinitiate the multi-step kill procedure outlined in Figure 6.23.

A *Kill* shall be prepended with a frame-sync (see 6.3.1.2.8).

Upon receiving a valid *Kill* command sequence a Tag shall render itself killed. The Tag's reply to the second *Kill* command shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 6.35 – *Kill* command

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

Table 6.36 – Tag reply to the first *Kill* command

	RN	CRC-16
# of bits	16	16
description	handle	

Table 6.37 – Tag reply to a successful *Kill* procedure

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

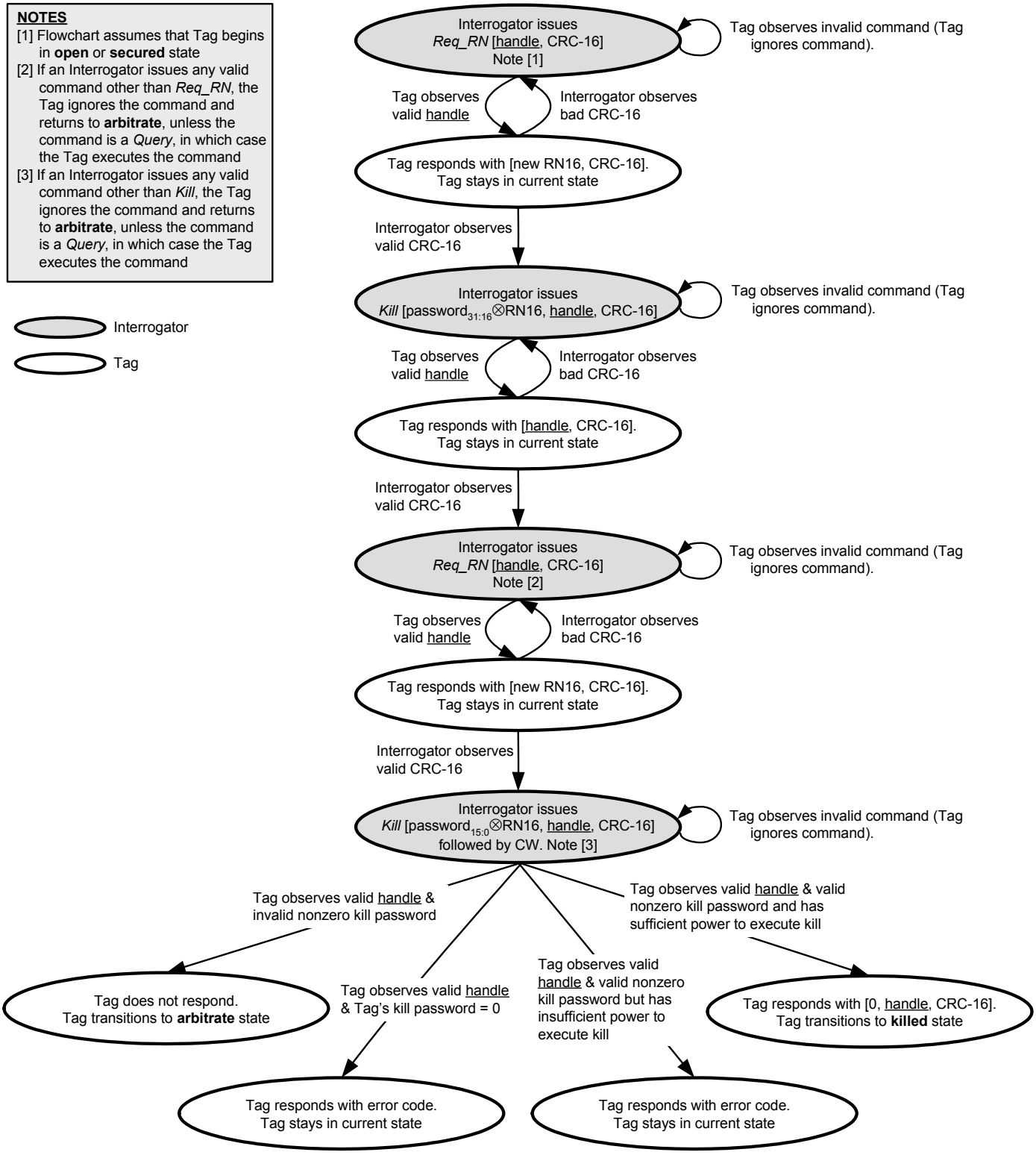


Figure 6.23 – *Kill* procedure

6.3.2.10.3.5 *Lock* (mandatory)

Interrogators and Tags shall implement the *Lock* command shown in Table 6.38 and Figure 6.24. Only Tags in the **secured** state shall execute a *Lock* command. *Lock* allows an Interrogator to:

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

Lock contains a 20-bit payload defined as follows:

- The first 10 payload bits are Mask bits. A Tag shall interpret these bit values as follows:
  - Mask = 0: Ignore the associated Action field and retain the current lock setting.
  - Mask = 1: Implement the associated Action field and overwrite the current lock setting.
- The last 10 payload bits are Action bits. A Tag shall interpret these bit values as follows:
  - Action = 0: Deassert lock for the associated memory location.
  - Action = 1: Assert lock or permalock for the associated memory location.

The functionality of the various Action fields is described in Table 6.40.

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

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

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

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

All Tags shall implement memory locking, and all Tags shall implement the *Lock* command. However, Tags need not support all the Action fields shown in Figure 6.24, depending on whether the password location or memory bank associated with an Action field exists and is lockable and/or unlockable. Specifically, if a Tag receives a *Lock* it cannot execute because one or more of the passwords or memory banks do not exist, or one or more of the Action fields attempt to change a previously permalocked value, or one or more of the passwords or memory banks are either not lockable or not unlockable, the Tag shall ignore the entire *Lock* and instead backscatter an error code (see Annex I). The only exception to this general rule relates to Tags whose only lock functionality is to permanently lock **all** memory (i.e. all memory banks and all passwords) at once; these Tags shall execute a *Lock* whose payload is $FFFFF_h$, and shall backscatter an error code for any payload other than $FFFFF_h$.

A *Lock* shall be prepended with a frame-sync (see 6.3.1.2.8).

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

- **The *Lock* succeeds:** After completing the *Lock* the Tag shall backscatter the reply shown in Table 6.39 and Figure 6.22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *Lock* completed successfully.
- **The Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 6.39 (see Annex I for error-code definitions and for the reply format).
- **The *Lock* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *Lock* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *Lock*.

Upon receiving a valid *Lock* command a Tag shall perform the commanded lock operation. The Tag's reply to a *Lock* shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

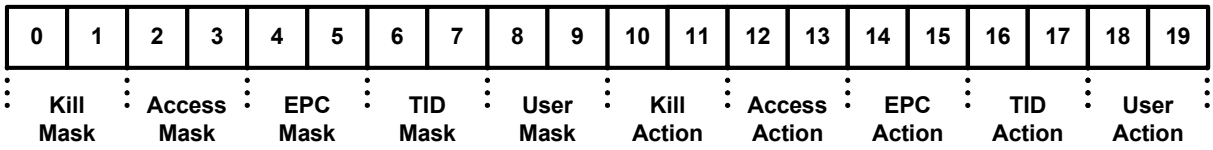
Table 6.38 – *Lock* command

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

Table 6.39 – Tag reply to a *Lock* command

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

Lock-Command Payload



Masks and Associated Action Fields

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

Figure 6.24 – *Lock* payload and usage

Table 6.40 – *Lock* Action-field functionality

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

6.3.2.10.3.6 *Access* (optional)

Interrogators and Tags may implement an *Access* command; if they do, the command shall be as shown in Table 6.41. *Access* causes a Tag with a nonzero-valued access password to transition from the **open** to the **secured** state (a Tag with a zero-valued access password is never in the **open** state — see Figure 6.19) or, if the Tag is already in the **secured** state, to remain in **secured**.

To access a Tag, an Interrogator shall follow the multi-step procedure outlined in Figure 6.25. Briefly, an Interrogator issues two *Access* commands, the first containing the 16 MSBs of the Tag's access password EXORed with an RN16, and the second containing the 16 LSBs of the Tag's access password EXORed with a different RN16. Each EXOR operation shall be performed MSB first (i.e. the MSB of each half-password shall be EXORed with the MSB of its respective RN16). Just prior to issuing each *Access* command the Interrogator first issues a *Req_RN* to obtain a new RN16.

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

An *Access* shall be prepended with a frame-sync (see 6.3.1.2.8).

The Tag reply to an *Access* command shall be as shown in Table 6.42. If the *Access* is the first in the sequence, then the Tag backscatters its handle to acknowledge that it received the command. If the *Access* is the second in the sequence and the entire received 32-bit access password is correct, then the Tag backscatters its handle to acknowledge that it has executed the command successfully and has transitioned to the **secured** state; otherwise the Tag does not reply. The reply includes a CRC-16 calculated over the handle.

Table 6.41 – *Access* command

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

Table 6.42 – Tag reply to an *Access* command

	RN	CRC-16
# of bits	16	16
description	handle	

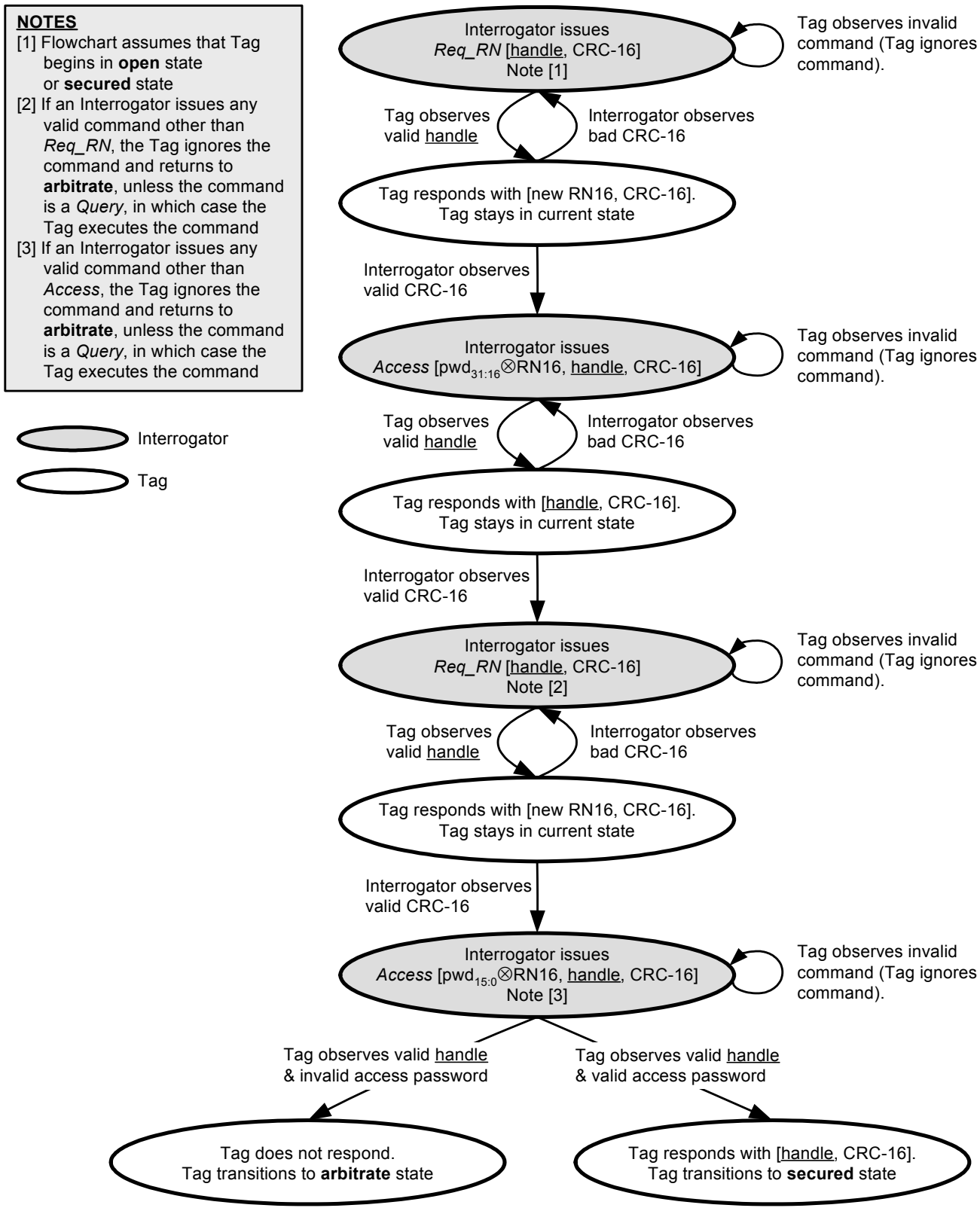


Figure 6.25 – *Access* procedure

6.3.2.10.3.7 *BlockWrite* (optional)

Interrogators and Tags may implement a *BlockWrite* command; if they do, they shall implement it as shown in Table 6.43. *BlockWrite* allows an Interrogator to write multiple words in a Tag's Reserved, EPC, TID, or User memory using a single command. *BlockWrite* has the following fields:

- MemBank specifies whether the *BlockWrite* occurs in Reserved, EPC, TID, or User memory. *BlockWrite* commands shall apply to a single memory bank. Successive *BlockWrites* may apply to different banks.
- WordPtr specifies the starting word address for the memory write, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).
- WordCount specifies the number of 16-bit words to be written. If WordCount = 00_h the Tag shall ignore the *BlockWrite*. If WordCount = 01_h the Tag shall write a single data word.
- Data contains the 16-bit words to be written, and shall be 16×WordCount bits in length. Unlike a *Write*, the data in a *BlockWrite* are not cover-coded, and an Interrogator need not issue a *Req_RN* before issuing a *BlockWrite*.

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

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

A *BlockWrite* shall be prepended with a frame-sync (see 6.3.1.2.8).

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

- **The *BlockWrite* succeeds:** After completing the *BlockWrite* a Tag shall backscatter the reply shown in Table 6.44 and Figure 6.22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockWrite* completed successfully.
- **The Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 6.44 (see Annex I for error-code definitions and for the reply format).
- **The *BlockWrite* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *BlockWrite* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *BlockWrite*.

Upon receiving a valid *BlockWrite* command a Tag shall write the commanded Data into memory. The Tag's reply to a *BlockWrite* shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 6.43 – *BlockWrite* command

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

Table 6.44 – Tag reply to a successful *BlockWrite* command

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

6.3.2.10.3.8 *BlockErase* (optional)

Interrogators and Tags may implement a *BlockErase* command; if they do, they shall implement it as shown in Table 6.45. *BlockErase* allows an Interrogator to erase multiple words in a Tag's Reserved, EPC, TID, or User memory using a single command. *BlockErase* has the following fields:

- MemBank specifies whether the *BlockErase* occurs in Reserved, EPC, TID, or User memory. *BlockErase* commands shall apply to a single memory bank. Successive *BlockErases* may apply to different banks.
- WordPtr specifies the starting word address for the memory erase, where words are 16 bits in length. For example, WordPtr = 00_h specifies the first 16-bit memory word, WordPtr = 01_h specifies the second 16-bit memory word, etc. WordPtr uses EBV formatting (see Annex A).
- WordCount specifies the number of 16-bit words to be erased. If WordCount = 00_h the Tag shall ignore the *BlockErase*. If WordCount = 01_h the Tag shall erase a single data word.

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

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

A *BlockErase* shall be prepended with a frame-sync (see 6.3.1.2.8).

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

- **The *BlockErase* succeeds:** After completing the *BlockErase* a Tag shall backscatter the reply shown in Table 6.46 and Figure 6.22 comprising a header (a 0-bit), the Tag's handle, and a CRC-16 calculated over the 0-bit and handle. If the Interrogator observes this reply within 20 ms then the *BlockErase* completed successfully.
- **The Tag encounters an error:** The Tag shall backscatter an error code during the CW period rather than the reply shown in Table 6.46 (see Annex I for error-code definitions and for the reply format).
- **The *BlockErase* does not succeed:** If the Interrogator does not observe a reply within 20ms then the *BlockErase* did not complete successfully. The Interrogator may issue a *Req_RN* command (containing the Tag's handle) to verify that the Tag is still in the Interrogator's field, and may reissue the *BlockErase*.

Upon receiving a valid *BlockErase* command a Tag shall erase the commanded memory words. The Tag's reply to a *BlockErase* shall use the extended preamble shown in Figure 6.11 or Figure 6.15, as appropriate (i.e. a Tag shall reply as if TRext=1 regardless of the TRext value in the *Query* that initiated the round).

Table 6.45 – *BlockErase* command

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

Table 6.46 – Tag reply to a successful *BlockErase* command

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

7. Intellectual property rights intrinsic to this specification

Attention is drawn to the possibility that some of the elements of this specification may be the subject of patent and/or other intellectual-property rights. EPCglobal shall not be held responsible for identifying any or all such patent or intellectual-property rights.

Annex A
(normative)
Extensible bit vectors (EBV)

An *extensible bit vector* (EBV) is a data structure with an extensible data range.

An EBV is an array of *blocks*. Each block contains a single extension bit followed by a specific number of data bits. If B represents the total number of bits in one block, then a block contains B – 1 data bits. Although a general EBV may contain blocks of varying lengths, Tags and Interrogators manufactured according to this specification shall use blocks of length 8 bits (EBV-8).

The data value represented by an EBV is simply the bit string formed by the data bits as read from left-to-right, ignoring the extension bits.

Tags and Interrogators shall use the EBV-8 word format specified in Table A.1.

Table A.1 – EBV-8 word format

	0	0	0000000				
	1	0	0000001				
$2^7 - 1$	127	0	1111111				
2^7	128	1	0000001	0	0000000		
$2^{14} - 1$	16383	1	1111111	0	1111111		
2^{14}	16384	1	0000001	1	0000000	0	0000000

Because each block has 7 data bits available, the EBV-8 can represent numeric values between 0 and 127 with a single block. To represent the value 128, the extension bit is set to 1 in the first block, and a second block is appended to the EBV-8. In this manner, an EBV-8 can represent arbitrarily large values.

This specification uses EBV-8 values to represent memory addresses and mask lengths.

Annex B
(normative)
State-transition tables

State-transition tables B.1 to B.7 shall define a Tag's response to Interrogator commands. The term "handle" used in the state-transition tables is defined in 6.3.2.4.5; error codes are defined in Table I.2; "slot" is the slot-counter output shown in Figure 6.19 and detailed in Annex J; "–" in the "Action" column means that a Tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

B.1 Present state: Ready

Table B.1 – Ready state-transition table

Command	Condition	Action	Next State
Query [1]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
QueryRep	all	–	**ready**
QueryAdjust	all	–	**ready**
ACK	all	–	**ready**
NAK	all	–	**ready**
Req_RN	all	–	**ready**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**ready**
Write	all	–	**ready**
Kill	all	–	**ready**
Lock	all	–	**ready**
Access	all	–	**ready**
BlockWrite	all	–	**ready**
BlockErase	all	–	**ready**
Invalid [2]	all	–	**ready**

1: *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

2: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

B.2 Present state: Arbitrate

Table B.2 – Arbitrate state-transition table

Command	Condition	Action	Next State
Query [1,2]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
QueryRep	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
	slot<>0 after decrementing slot counter	–	**arbitrate**
QueryAdjust [2]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	all	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
Erase	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
Invalid [3]	all	–	**arbitrate**

1: *Query* starts a new round and may change the session.

2: *Query* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.

3: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.3 Present state: Reply

Table B.3 – Reply state-transition table

Command	Condition	Action	Next State
Query [1,2]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
QueryRep	all	–	**arbitrate**
QueryAdjust [2]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	valid RN16	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
T_2 timeout	$T_2 > 20.0T_{pri}$ (see Figure 6.16)	–	**arbitrate**
Invalid [3]	all	–	**reply**

1: *Query* starts a new round and may change the session.

2: *Query* and *QueryAdjust* instruct a Tag to load a new random value into its slot counter.

3: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.4 Present state: Acknowledged

Table B.4 – Acknowledged state-transition table

Command	Condition	Action	Next State
Query [1]	slot=0; matching **inventoried**[2] & **SL** flags	backscatter new RN16; transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[2] & **SL** flags	transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid RN16	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	all	–	**arbitrate**
Write	all	–	**arbitrate**
Kill	all	–	**arbitrate**
Lock	all	–	**arbitrate**
Access	all	–	**arbitrate**
BlockWrite	all	–	**arbitrate**
BlockErase	all	–	**arbitrate**
T_2 timeout	$T_2 > 20.0T_{pri}$ (see Figure 6.16)	–	**arbitrate**
Invalid [3]	all	–	**acknowledged**

1: *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

2: As described in 6.3.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

3: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.5 Present state: Open

Table B.5 – Open state-transition table

Command	Condition	Action	Next State
Query [1]	slot=0; matching **inventoried**[2] & **SL** flags	backscatter new RN16; transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[2] & **SL** flags	transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid handle	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**open**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	valid handle & valid memory access	backscatter data and handle	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Write	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Kill (see also Figure 6.23)	valid handle & valid nonzero kill password	backscatter handle when done	**killed**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**open**
	invalid handle	–	**open**
Lock	all	–	**open**
Access (see also Figure 6.25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**open**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
BlockErase	valid handle & valid memory access	backscatter handle when done	**open**
	valid handle & invalid memory access	backscatter error code	**open**
	invalid handle	–	**open**
Invalid [3]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**open**
	commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**arbitrate**

1: *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

2: As described in 6.3.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

3: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.6 Present state: Secured

Table B.6 – Secured state-transition table

Command	Condition	Action	Next State
Query [1]	slot=0; matching **inventoried**[2] & **SL** flags	backscatter new RN16; transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[2] & **SL** flags	transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
QueryRep	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
QueryAdjust	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
ACK	valid handle	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**secured**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
Select	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
Read	valid handle & valid memory access	backscatter data and handle	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Write	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Kill (see also Figure 6.23)	valid handle & valid nonzero kill password	backscatter handle when done	**killed**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & kill password=0	backscatter error code	**secured**
	invalid handle	–	**secured**
Lock	valid handle & valid lock payload	backscatter handle when done	**secured**
	valid handle & invalid lock payload	backscatter error code	**secured**
	invalid handle	–	**secured**
Access (see also Figure 6.25)	valid handle & valid access password	backscatter handle	**secured**
	valid handle & invalid access password	–	**arbitrate**
	invalid handle	–	**secured**
BlockWrite	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
BlockErase	valid handle & valid memory access	backscatter handle when done	**secured**
	valid handle & invalid memory access	backscatter error code	**secured**
	invalid handle	–	**secured**
Invalid [3]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**secured**
	commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**arbitrate**

1: *Query* starts a new round and may change the session. *Query* also instructs a Tag to load a new random value into its slot counter.

2: As described in 6.3.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

3: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

B.7 Present state: Killed

Table B.7 – Killed state-transition table

Command	Condition	Action	Next State
Query	all	–	**killed**
QueryRep	all	–	**killed**
QueryAdjust	all	–	**killed**
ACK	all	–	**killed**
NAK	all	–	**killed**
Req_RN	all	–	**killed**
Select	all	–	**killed**
Read	all	–	**killed**
Write	all	–	**killed**
Kill	all	–	**killed**
Lock	all	–	**killed**
Access	all	–	**killed**
BlockWrite	all	–	**killed**
BlockErase	all	–	**killed**
Invalid [1]	all	–	**killed**

1: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

Annex C
(normative)
Command-Response Tables

Command-response tables C.1 to C.17 shall define a Tag's response to Interrogator commands. The term "handle" used in the state-transition tables is defined in 6.3.2.4.5; error codes are defined in Table I.2; "slot" is the slot-counter output shown in Figure 6.19 and detailed in Annex J; "–" in the "Response" column means that a Tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

C.1 Command response: Power-up

Table C.1 – Power-up command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	power-up	–	**ready**
killed	all	–	**killed**

C.2 Command response: *Query*

Table C.2 – *Query*[1] command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
acknowledged, open, secured	slot=0; matching **inventoried**[2] & **SL** flags	backscatter new RN16; transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[2] & **SL** flags	transition **inventoried**[2] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**arbitrate**
	otherwise	transition **inventoried** from *A*→*B* or *B*→*A* if and only if new session matches prior session	**ready**
killed	all	–	**killed**

1: *Query* (in any state other than **killed**) starts a new round and may change the session; *Query* also instructs a Tag to load a new random value into its slot counter.

2: As described in 6.3.2.8, a Tag transitions its **inventoried** flag prior to evaluating the condition.

C.3 Command response: *QueryRep*

Table C.3 – *QueryRep* command-response table[1]

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	slot<>0 after decrementing slot counter	–	**arbitrate**
	slot=0 after decrementing slot counter	backscatter new RN16	**reply**
reply	all	–	**arbitrate**
acknowledged, open, secured	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
killed	all	–	**killed**

1: See Table C.17 for the Tag response to a *QueryRep* whose session parameter does not match that of the current inventory round.

C.4 Command response: *QueryAdjust*

Table C.4 – *QueryAdjust*[1] command-response table[2]

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply	slot<>0	–	**arbitrate**
	slot=0	backscatter new RN16	**reply**
acknowledged, open, secured	all	transition **inventoried** from *A*→*B* or *B*→*A*	**ready**
killed	all	–	**killed**

1: *QueryAdjust*, in the **arbitrate** or **reply** states, instructs a Tag to load a new random value into its slot counter.

2: See Table C.17 for the Tag response to a *QueryAdjust* whose session parameter does not match that of the current inventory round.

C.5 Command response: *ACK*

Table C.5 – *ACK* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	all	–	**arbitrate**
reply	valid RN16	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**acknowledged**
	invalid RN16	–	**arbitrate**
acknowledged	valid RN16	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**acknowledged**
	invalid RN16	–	**arbitrate**
open	valid handle	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**open**
	invalid handle	–	**arbitrate**
secured	valid handle	backscatter {PC, EPC, CRC-16} or {00000_2, truncated EPC, CRC-16}	**secured**
	invalid handle	–	**arbitrate**
killed	all	–	**killed**

C.6 Command response: *NAK*

Table C.6 – *NAK* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged, open, secured	all	–	**arbitrate**
killed	all	–	**killed**

C.7 Command response: *Req_RN*

Table C.7 – *Req_RN* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply	all	–	**arbitrate**
acknowledged	valid RN16 & access password<>0	backscatter handle	**open**
	valid RN16 & access password=0	backscatter handle	**secured**
	invalid RN16	–	**acknowledged**
open	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
secured	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.8 Command response: *Select*

Table C.8 – *Select* command-response table

Starting State	Condition	Response	Next State
ready, arbitrate, reply, acknowledged, open, secured	all	assert or deassert **SL**, or set **inventoried** to *A* or *B*	**ready**
killed	all	–	**killed**

C.9 Command response: *Read*

Table C.9 – *Read* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter data and handle	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter data and handle	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.10 Command response: *Write*

Table C.10 – *Write* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.11 Command response: *Kill*

Table C.11 – *Kill*[1] command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & kill password=0	backscatter error code	**open**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password	backscatter handle when done	**killed**
	invalid handle	–	**open**
secured	valid handle & kill password=0	backscatter error code	**secured**
	valid handle & invalid nonzero kill password	–	**arbitrate**
	valid handle & valid nonzero kill password	backscatter handle when done	**killed**
	invalid handle	–	**secured**
killed	all	–	**killed**

1: See also Figure 6.23.

C.12 Command response: *Lock*

Table C.12 – *Lock* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	all	–	**open**
secured	valid handle & invalid lock payload	backscatter error code	**secured**
	valid handle & valid lock payload	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.13 Command response: *Access*

Table C.13 – *Access*[1] command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid access password	–	**arbitrate**
	valid handle & valid access password	backscatter handle	**secured**
	invalid handle	–	**open**
secured	valid handle & invalid access password	–	**arbitrate**
	valid handle & valid access password	backscatter handle	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

1: See also Figure 6.25.

C.14 Command response: *BlockWrite*

Table C.14 – *BlockWrite* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.15 Command response: *BlockErase*

Table C.15 – *BlockErase* command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate, reply, acknowledged	all	–	**arbitrate**
open	valid handle & invalid memory access	backscatter error code	**open**
	valid handle & valid memory access	backscatter handle when done	**open**
	invalid handle	–	**open**
secured	valid handle & invalid memory access	backscatter error code	**secured**
	valid handle & valid memory access	backscatter handle when done	**secured**
	invalid handle	–	**secured**
killed	all	–	**killed**

C.16 Command response: T_2 timeout

Table C.16 – T_2 timeout command-response table

Starting State	Condition	Response	Next State
ready	all	–	**ready**
arbitrate	all	–	**arbitrate**
reply, acknowledged	$T_2 > 20.0T_{pri}$ (see Figure 6.16)	–	**arbitrate**
open	all	–	**open**
secured	all	–	**secured**
killed	all	–	**killed**

C.17 Command response: Invalid command

Table C.17 – Invalid command-response table

Starting State	Condition	Response	Next State
ready[1]	all	–	**ready**
arbitrate[2]	all	–	**arbitrate**
reply[2]	all	–	**reply**
acknowledged[2]	all	–	**acknowledged**
open[2]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**open**
	commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**arbitrate**
secured[2]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**secured**
	commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure 6.23 and Figure 6.25).	–	**arbitrate**
killed[1]	all	–	**killed**

1: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, or any other command either not recognized or not executable by the Tag.

2: "Invalid" shall mean an erroneous command, an unsupported command, a command with invalid parameters, a command with a CRC error, a command (other than a *Query*) with a session parameter not matching that of the inventory round currently in progress, or any other command either not recognized or not executable by the Tag.

Annex D
(informative)
Example slot-count (*Q*) selection algorithm

D.1 Example algorithm an Interrogator might use to choose *Q*

Figure D.1 shows an algorithm an Interrogator might use for setting the slot-count parameter *Q* in a *Query* command. Q_{fp} is a floating-point representation of *Q*; an Interrogator rounds Q_{fp} to an integer value and substitutes this integer value for *Q* in the *Query*. Typical values for *C* are $0.1 < C < 0.5$. An Interrogator typically uses small values of *C* when *Q* is large, and larger values of *C* when *Q* is small.

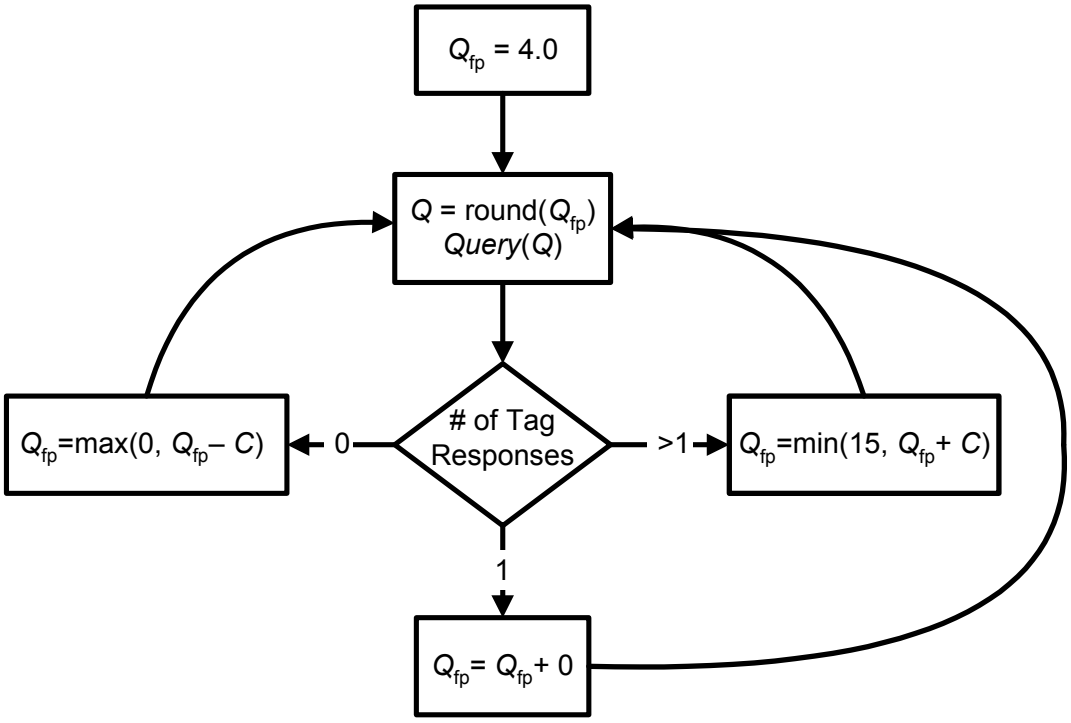


Figure D.1 – Example algorithm for choosing the slot-count parameter *Q*

Annex E
(informative)
Example of Tag inventory and access

E.1 Example inventory and access of a single Tag

Figure E.1 shows the steps by which an Interrogator inventories and accesses a single Tag.

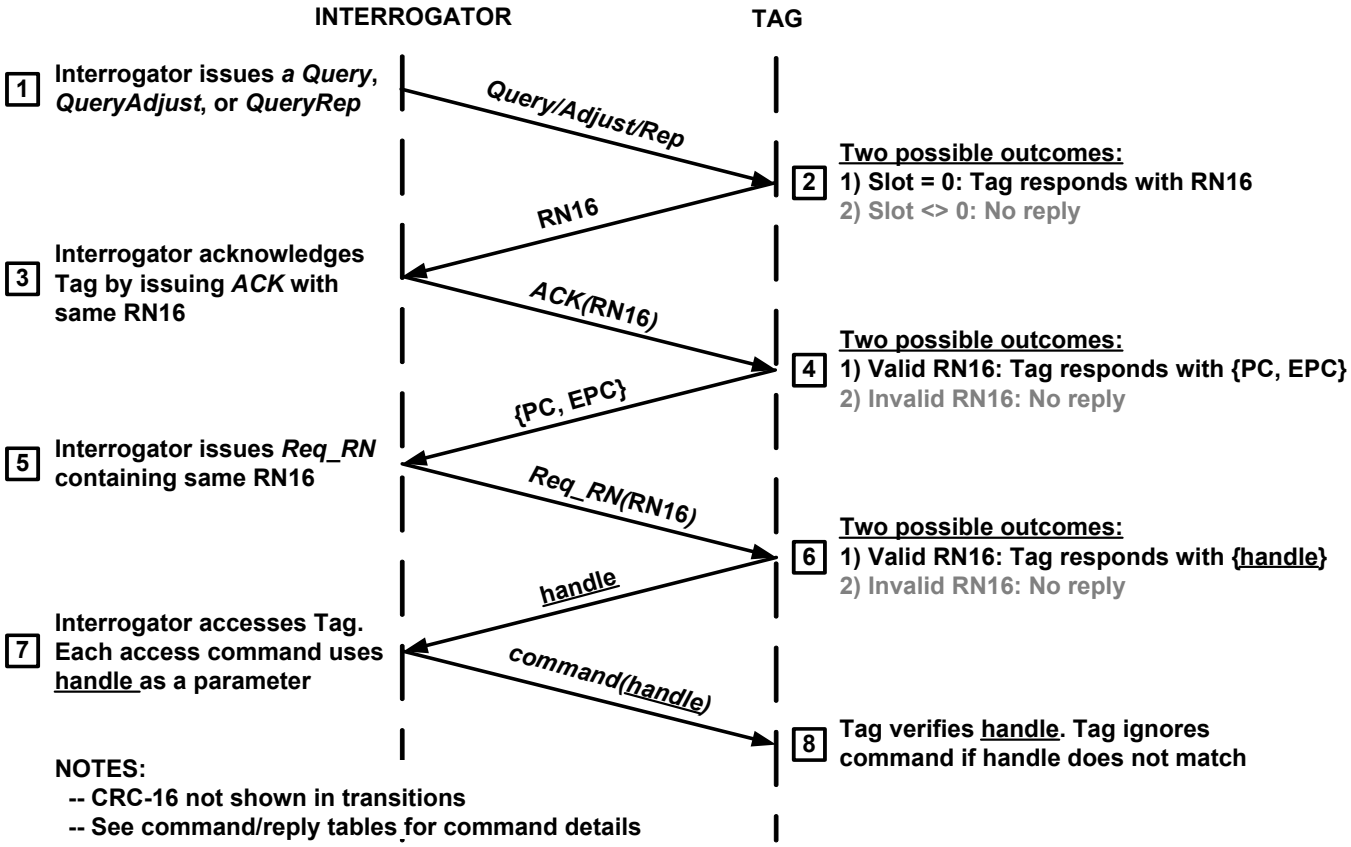


Figure E.1 – Example of Tag inventory and access

Annex F
(informative)
Calculation of 5-bit and 16-bit cyclic redundancy checks

F.1 Example CRC-5 encoder/decoder

An exemplary schematic diagram for a CRC-5 encoder/decoder is shown in Figure F.1, using the polynomial and preset defined in Table 6.17.

To encode a CRC-5, first preload the entire CRC register (i.e. C[4:0]) with 01001_2, then clock the data bits to be encoded into the input labeled DATA, MSB first. After clocking in all the data bits, C[4:0] holds the CRC-5 value.

To decode a CRC-5, first preload the entire CRC register (C[4:0]) with 01001_2, then clock the received data and CRC-5 {data, CRC-5} bits into the input labeled DATA, MSB first. The CRC-5 check passes if C[4:0] = 00000_2.

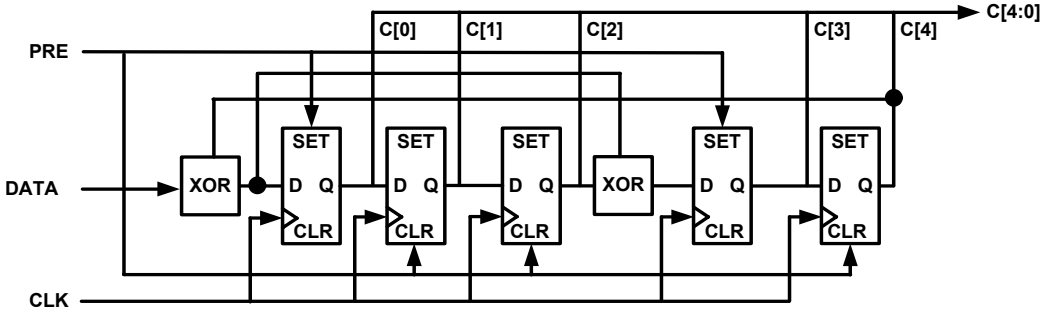


Figure F.1 – Example CRC-5 circuit

F.2 Example CRC-16 encoder/decoder

An exemplary schematic diagram for a CRC-16 encoder/decoder is shown in Figure F.2, using the polynomial and preset defined in Table 6.14.

To encode a CRC-16, first preload the entire CRC register (i.e. C[15:0]) with $FFFF_h$, then clock the data bits to be encoded into the input labeled DATA, MSB first. After clocking in all the data bits, C[15:0] holds the ones-complement of the CRC-16 value.

To decode a CRC-16, first preload the entire CRC register (C[15:0]) with $FFFF_h$, then clock the received data and CRC-16 {data, CRC-16} bits into the input labeled DATA, MSB first. The CRC-16 check passes if C[15:0] =$1D0F_h$.

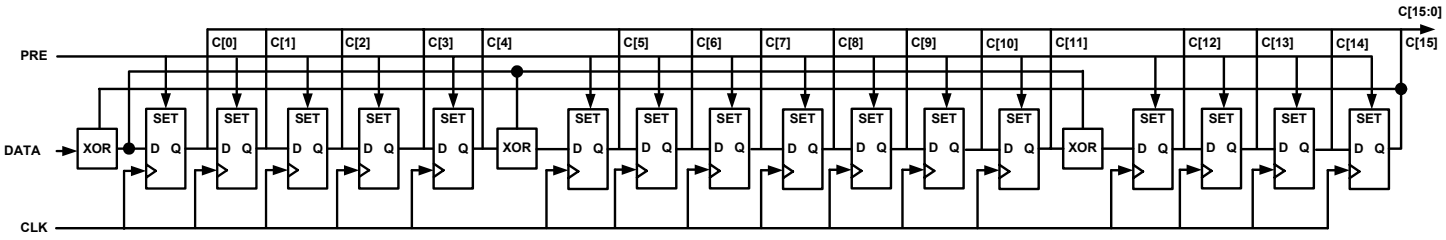


Figure F.2 – Example CRC-16 circuit

Annex G
(Normative)
Dense-Interrogator channelized signaling

G.1 Interrogator and Tag signaling in dense-Interrogator environments

In environments containing many Interrogators, the range and rate at which Interrogators singulate Tags can be improved by preventing interfering Interrogator transmissions from colliding with desired Tags responses, either spectrally or temporally. This Annex describes frequency plans and time-division multiplexing (TDM) methods, each tailored to specific regulatory environments, which minimize or eliminate Interrogator-on-Tag collisions.

Interrogators certified for operation in dense-Interrogator environments shall be capable of supporting one or more of the frequency plans or TDM approaches described below. The choice will depend on local regulations at the time and place of operation, in CEPT- or FCC-regulated environments, as follows:

CEPT Environments: There are two allowed approaches, depending on the regulatory environment.

Single-channel regulatory environment: Interrogator transmissions and Tag responses shall be separated temporally, with synchronized Interrogators first commanding Tags, then all Interrogators transmitting CW and listening for Tag responses. Interrogator signaling (both modulated and CW) shall be centered in the channel with a frequency accuracy as specified in 6.3.1.2.1. If an Interrogator uses SSB modulation, the transmit spectrum shall be centered in the channel during R=>T signaling, and the CW shall be centered in the channel during Tag backscatter. The R=>T and T=>R modulation, rate, and encoding are chosen by the Interrogator and not specified in this Annex. As an informative example, part A of Figure G.2 shows Interrogator signaling using DSB-ASK modulation with Tari=25μs, and 20 kbps Tag backscatter on an 80 kHz subcarrier (M=4).

Multi-channel regulatory environment: Interrogator transmissions and Tag responses shall be separated spectrally, with Interrogator transmissions located in even-numbered channels and Tag backscatter located in odd-numbered channels (see Figure G.1). Interrogator signaling (both modulated and CW) shall be centered in a channel with a frequency accuracy as specified in 6.3.1.2.1. If an Interrogator uses SSB-ASK modulation, the transmit spectrum shall be centered in the channel during R=>T signaling, and the CW shall be centered in the channel during Tag backscatter. Interrogator transmissions shall satisfy the dense-Interrogator transmit mask in Figure 6.7 with Tari=25μs. Tag backscatter shall be 53.3 or 26.7 kbps data on a 213.3 kHz subcarrier (M=4 or M=8). As an informative example, part B of Figure G.2 shows Interrogator signaling using SSB modulation with Tari=25μs, and 53.3 kbps Tag backscatter on a 213.3 kHz subcarrier (M=4).

FCC environments:

Interrogator transmissions and Tag responses shall be separated spectrally, with Interrogator transmissions centered in channels and Tag responses situated at channel boundaries. The frequency band shall be channelized as in Table 6.10. Interrogator transmissions are unsynchronized in time, hopping among channels. Interrogator signaling (both modulated and CW) shall be centered in a channel with frequency accuracy as specified in 6.3.1.2.1. If an Interrogator uses SSB modulation, the transmit spectrum shall be centered in the channel during R=>T signaling, and the CW shall be centered in the channel during Tag backscatter. Interrogator transmissions shall satisfy the dense-Interrogator transmit mask in Figure 6.7 with Tari=25μs. Tag backscatter shall be 64 or 32 kbps data on a 256 kHz subcarrier (M=4 or M=8). As an informative example, part C of Figure G.2 shows Interrogator signaling using PR-ASK modulation with Tari=25μs, and 64 kbps Tag backscatter on a 256 kHz subcarrier.

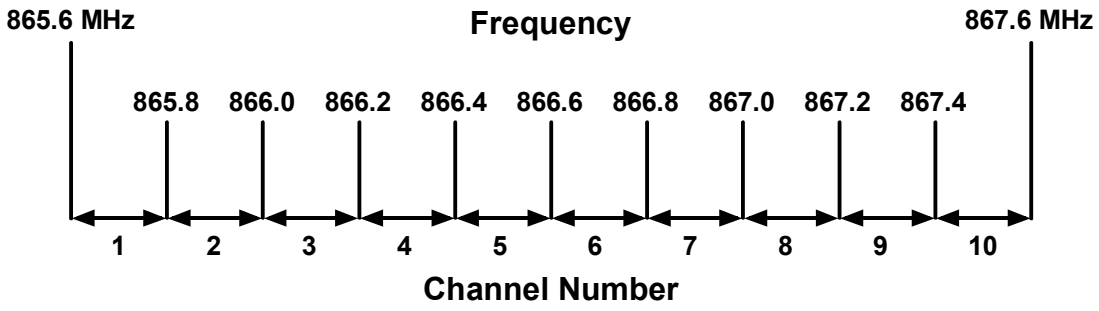


Figure G.1 – Multi-channel CEPT environment

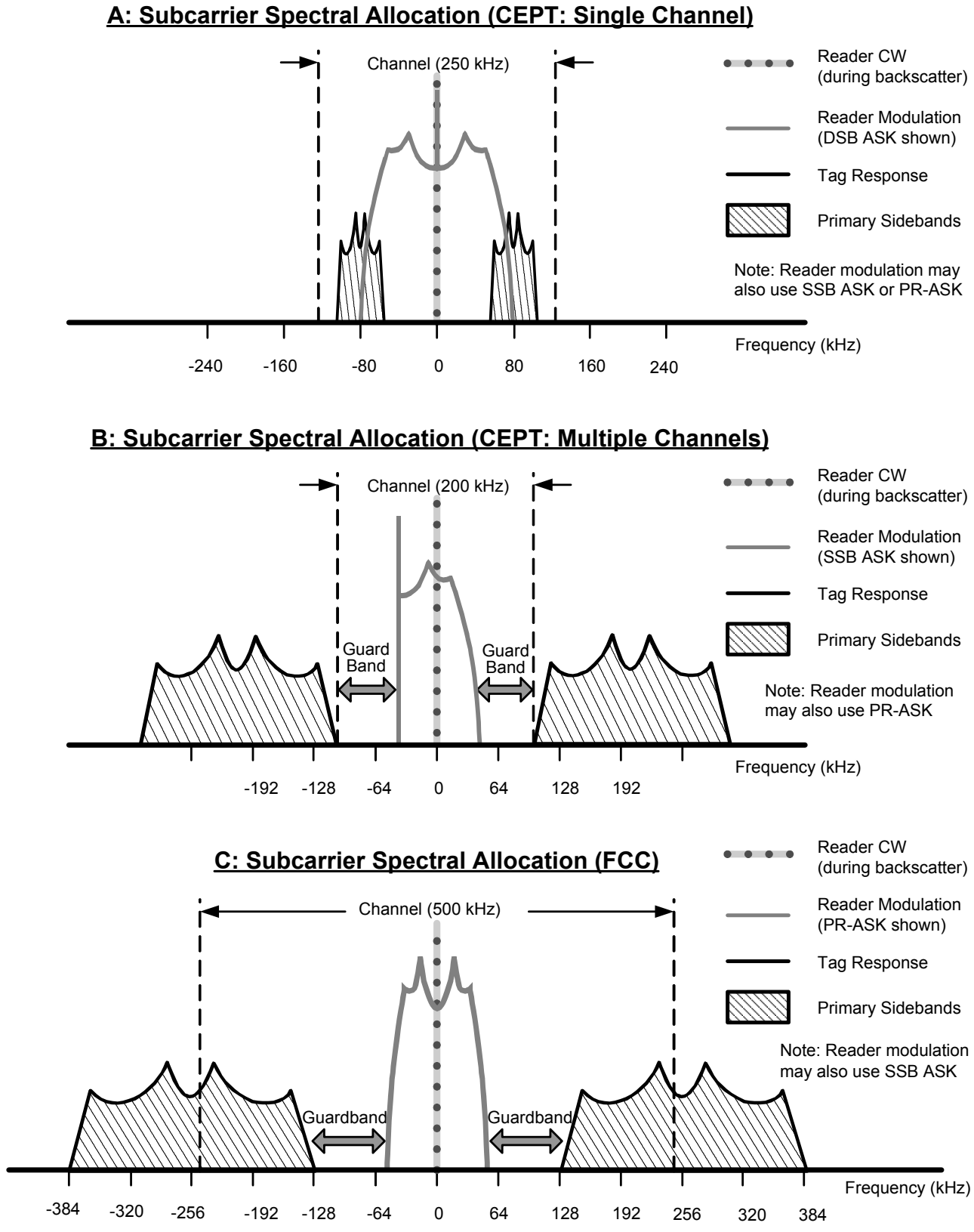


Figure G.2 – Subcarrier spectral allocation for CEPT and FCC environments

Annex H
(informative)
Interrogator-to-Tag link modulation

H.1 Baseband waveforms, modulated RF, and detected waveforms

Figure H.1 shows R=>T baseband and modulated waveforms as generated by an Interrogator, and the corresponding waveforms envelope-detected by a Tag, for DSB- or SSB-ASK modulation, and for PR-ASK modulation.

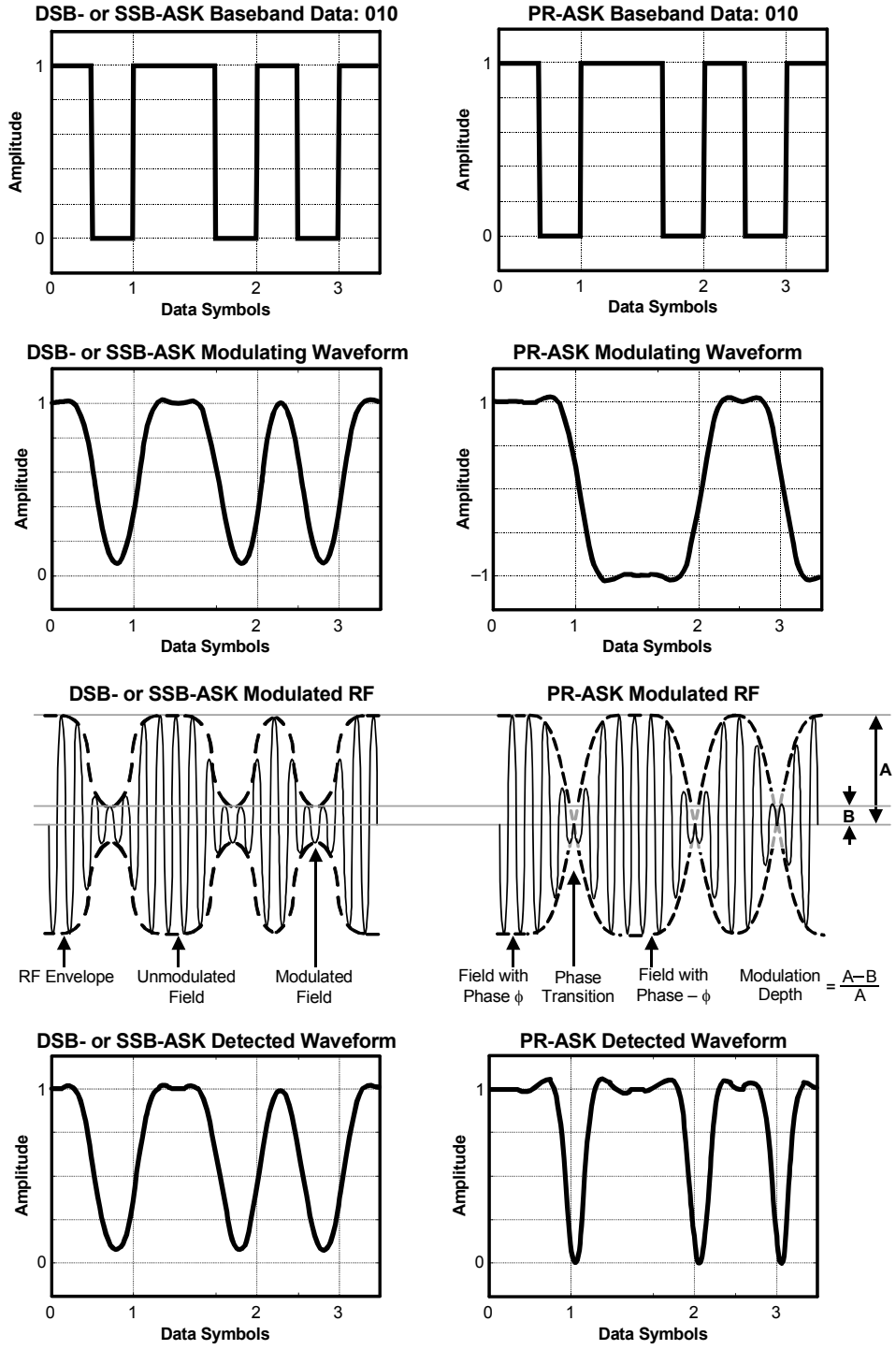


Figure H.1 – Interrogator-to-Tag modulation

Annex I
(Normative)
Error codes

I.1 Tag error codes and their usage

If a Tag encounters an error when executing an access command that reads from or writes to memory, and if the command is a handle-based command (i.e. *Read*, *Write*, *Kill*, *Lock*, *BlockWrite*, or *BlockErase*), then the Tag shall backscatter an error code as shown in Table I.1 instead of its normal reply.

- If the Tag supports error-specific codes, it shall use the error-specific codes shown in Table I.2.
- If the Tag does not support error-specific codes, it shall backscatter error code 00001111_2 (indicating a non-specific error) as shown in Table I.2.
- Tags shall backscatter error codes only from the **open** or **secured** states.
- A Tag shall not backscatter an error code if it receives an invalid access command; instead, it shall ignore the command.
- If an error is described by more than one error code, the more specific error code shall take precedence and shall be the code that the Tag backscatters.
- The header for an error code is a 1-bit, unlike the header for a normal Tag response, which is a 0-bit.

Table I.1 – Tag-error reply format

	Header	Error Code	RN	CRC-16
# of bits	1	8	16	16
description	1	Error code	handle	

Table I.2 – Tag error codes

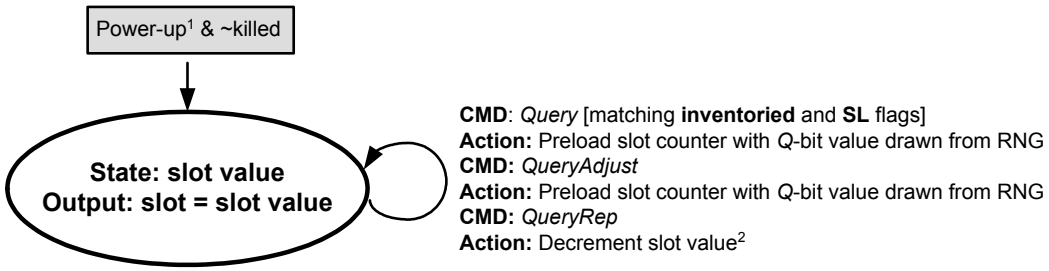
Error-Code Support	Error Code	Error-Code Name	Error Description
Error-specific	00000000_2	Other error	Catch-all for errors not covered by other codes
	00000011_2	Memory overrun or unsupported PC value	The specified memory location does not exist or the PC value is not supported by the Tag
	00000100_2	Memory locked	The specified memory location is locked and/or permalocked and is either not writeable or not readable.
	00001011_2	Insufficient power	The Tag has insufficient power to perform the memory-write operation
Non-specific	00001111_2	Non-specific error	The Tag does not support error-specific codes

Annex J
(normative)
Slot counter

J.1 Slot-counter operation

As described in 6.3.2.4.8, Tags implement a 15-bit slot counter. As described in 6.3.2.8, Interrogators use the slot counter to regulate the probability of a Tag responding to a *Query*, *QueryAdjust*, or *QueryRep* command. Upon receiving a *Query* or *QueryAdjust* a Tag preloads a *Q*-bit value, drawn from the Tag's RNG (see 6.3.2.5), into its slot counter. *Q* is an integer in the range (0,15). A *Query* specifies *Q*; a *QueryAdjust* may modify *Q* from the prior *Query*. Upon receiving a *QueryRep* a Tag decrements its slot value. Tags transition to the **reply** state when their slot value reaches 0000_h. The slot counter implements continuous counting, meaning that, after the slot value decrements to 0000_h, the next *QueryRep* causes it to roll over and begin counting down from $7FFF_h$. Tags that return to **arbitrate** (for example, from the **reply** state) with a slot value of 0000_h decrement their slot value from 0000_h to $7FFF_h$ at the next *QueryRep* (with matching session) and, because their slot value is now nonzero, remain in **arbitrate**.

Annex B and Annex C contain tables describing a Tag's response to Interrogator commands; "slot" is a parameter in these tables.



Notes
1. The slot counter may assume an arbitrary value at Tag power-up
2. If slot value = 0000_h, then decrementing the slot value causes it to roll over to $7FFF_h$

Figure J.1 – Slot-counter state diagram

Annex K
(informative)
Example data-flow exchange

K.1 Overview of the data-flow exchange

The following example describes a data exchange, between an Interrogator and a single Tag, during which the Interrogator reads the kill password stored in the Tag's Reserved memory. This example assumes that:

- The Tag has been singulated and is in the **acknowledged** state.
- The Tag's Reserved memory is locked but not permalocked, meaning that the Interrogator must issue the access password and transition the Tag to the **secured** state before performing the read operation.
- The random numbers the Tag generates (listed in sequence, and not random for reasons of clarity) are:
  - RN16_0 1600_h (the RN16 the Tag backscattered prior to entering **acknowledged**)
  - RN16_1 1601_h (will become the handle for the entire access sequence)
  - RN16_2 1602_h
  - RN16_3 1603_h
- The Tag's EPC is 64 bits in length.
- The Tag's access password is $ACCEC0DE_h$.
- The Tag's kill password is $DEADC0DE_h$.
- The 1^{st} half of the access password EXORed with RN16_2 = $ACCE_h \otimes 1602_h = BACC_h$.
- The 2^{nd} half of the access password EXORed with RN16_3 = $C0DE_h \otimes 1603_h = D6DD_h$.

K.2 Tag memory contents and lock-field values

Table K.1 and Table K.2 show the example Tag memory contents and lock-field values, respectively.

Table K.1 Tag memory contents

Memory Bank	Memory Contents	Memory Addresses	Memory Values
TID	TID[15:0]	10_h–$1F_h$	$54E2_h$
	TID[31:16]	00_h–$0F_h$	$A986_h$
EPC	EPC[15:0]	50_h–$5F_h$	3210_h
	EPC[31:16]	40_h–$4F_h$	7654_h
	EPC[47:32]	30_h–$3F_h$	$BA98_h$
	EPC[63:48]	20_h–$2F_h$	$FEDC_h$
	PC[15:0]	10_h–$1F_h$	2000_h
	CRC-16[15:0]	00_h–$0F_h$	as calculated (see Annex F)
Reserved	access password[15:0]	30_h–$3F_h$	$C0DE_h$
	access password[31:16]	20_h–$2F_h$	$ACCE_h$
	kill password[15:0]	10_h–$1F_h$	$C0DE_h$
	kill password[31:16]	00_h–$0F_h$	$DEAD_h$

Table K.2 – Lock-field values

Kill Password		Access Password		EPC Memory		TID Memory		User Memory	
1	0	1	0	0	0	0	0	N/A	N/A

K.3 Data-flow exchange and command sequence

The data-flow exchange follows the *Access* procedure outlined in Figure 6.25 with a *Read* command added at the end. The sequence of Interrogator commands and Tag replies is:

Step 1: *Req_RN*[RN16_0, CRC-16]
Tag backscatters RN16_1, which becomes the handle for the entire access sequence

Step 2: *Req_RN*[handle, CRC-16]
Tag backscatters RN16_2

Step 3: *Access*[access password[31:16] EXORed with RN16_2, handle, CRC-16]
Tag backscatters handle

Step 4: *Req_RN*[handle, CRC-16]
Tag backscatters RN16_3

Step 5: *Access*[access password[15:0] EXORed with RN16_3, handle, CRC-16]
Tag backscatters handle

Step 6: *Read*[MemBank=Reserved, WordPtr=00_h, WordCount=2, handle, CRC-16]
Tag backscatters kill password

Table K.3 shows the detailed Interrogator commands and Tag replies. For reasons of clarity, the CRC-16 has been omitted from all commands and replies.

Table K.3 – Interrogator commands and Tag replies

Step	Data Flow	Command	Parameter and/or Data	Tag State
1a: *Req_RN* command	R => T	11000001	0001 0110 0000 0000 (RN16_0=1600_h)	**acknowledged → open**
1b: Tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
2a: *Req_RN* command	R => T	11000001	0001 0110 0000 0001 (handle=1601_h)	**open → open**
2b: Tag response	T => R		0001 0110 0000 0010 (RN16_2=1602_h)	
3a: *Access* command	R => T	11000110	1011 1010 1100 1100 ($BACC_h$) 0001 0110 0000 0001 (handle=1601_h)	**open → open**
3b: Tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
4a: *Req_RN* command	R => T	11000001	0001 0110 0000 0001 (handle=1601_h)	**open → open**
4b: Tag response	T => R		0001 0110 0000 0011 (RN16_2=1603_h)	
5a: *Access* command	R => T	11000110	1101 0110 1101 1101 ($D6DD_h$) 0001 0110 0000 0001 (handle=1601_h)	**open → secured**
5b: Tag response	T => R		0001 0110 0000 0001 (handle=1601_h)	
6a: *Read* command	R => T	11000010	00 (MemBank=Reserved) 00000000 (WordPtr=kill password) 00000010 (WordCount=2) 0001 0110 0000 0001 (handle=1601_h)	**secured → secured**
6b: Tag response	T => R		0 (header) 1101 1110 1010 1101 ($DEAD_h$) 1100 0000 1101 1110 ($C0DE_h$)	

Annex L
(informative)
Revision History

Table L.1 – Revision history

Date & Version Number	Section(s)	Change	Approved by
Sept 8, 2004 Version 1.0.4	All	Modified Chicago protocol V1.0.3 as per August 17, 2004 “combo” CRC change template.	
Sept 14, 2004 Version 1.0.5	All	Modified Gen2 protocol V1.0.4 as per September 10, 2004 CRC review.	
Sept 17, 2004 Version 1.0.6	All	Modified Gen2 protocol V1.0.5 as per September 17, 2004 HAG review.	
Sept 24, 2004 Version 1.0.7	All	Modified Gen2 protocol V1.0.6 as per September 21, 2004 CRC review to fix errata. Changed OID to EPC.	
Dec 11, 2004 Version 1.0.8	Multiple	Modified Gen2 protocol V1.0.7 as per the V1.0.7 errata.	
Jan 26, 2005 Version 1.0.9	Multiple	Modified Gen2 protocol V1.0.8 as per the V1.0.8 errata and AFI enhancement requests.	