FINAL
DRAFT

AMENDMENT

ISO/IEC 18000-6:2004 FDAM 1

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2006-03-03

Voting terminates on:
2006-05-03

Information technology — Radio frequency identification for item management —

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

AMENDMENT 1: Extension with Type C and update of Types A and B

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

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

AMENDEMENT 1: Extension avec Type C et mise à jour des Types A et B

Please see the administrative notes on page iii

Reference number
ISO/IEC 18000-6:2004/FDAM 1:2006(E)

In accordance with the provisions of Council Resolution 21/1986, this document is **circulated in the English language only**.

Foreword

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

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

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

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

Amendment 1 to ISO/IEC 18000-6:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

Amendment 1 to ISO/IEC 18000-6:2004 covers the extension of ISO/IEC 18000-6 to Type C, to accommodate the latest development of passive RFID technology in the UHF frequency band from 860 MHz to 960 MHz.

Furthermore, it covers changes in order to achieve an improved collision arbitration and a more robust protocol for Type A.

Information technology — Radio frequency identification for item management —

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

AMENDMENT 1: Extension with Type C and update of Types A and B

Page vii, Introduction

Replace the paragraph after the bulleted list with the following paragraphs:

This International Standard specifies the physical and logical requirements for a passive-backscatter, interrogator-talks-first (ITF), radio frequency identification (RFID) system operating in the 860 MHz to 960 MHz frequency range. The system comprises interrogators, also known as readers, and tags, also known as labels.

An interrogator transmits information to a tag by modulating an RF signal in the 860 MHz to 960 MHz frequency range. The tag receives both information and operating energy from this RF signal. 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.

Page vii, Introduction

Add the following to the table of patent holders:

Contact details	Patent number	Affected clause(s) in this part of ISO/IEC 18000
Alien Technology Corporation ATTN: Dr. John Stephen Smith 18200 Butterfield Blvd Morgan Hill CA 95037 USA Tel: 1-408-782-3900 Fax: 1-408-782-3910 E-mail: ssmith@alientechnology.com	6,933,848 USA, 10/141,489 USA, 11/029,445 USA, 2003/0019929 USA, 10/160,458 USA, 11/132,085 USA, 11/153,030 USA, (divisional of 6,942,155 US), Not yet assigned, (continuation of 11/153,030 USA), 2005/0114326 USA, 10/982,557 USA, Not yet assigned, (continuation of 10/982,557 USA), Not yet assigned, (continuation of 10/982,557 USA), US04/036991 PCT, WO2005048180, 2003/0137403 USA, 10/267,924 USA, WO03032240, 2820082.9 China, 1636039A China, 2801064.3 EU, 2003-535135 Japan, 091123291 Taiwan, 60/681,656 USA, 10/915,725 USA, US04/025883 PCT, 10/140,557 US,	9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.3.4, Fig. Amd.1-22, Fig. Amd.1-26, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.3.4, Fig. Amd.1-22, Fig. Amd.1-26, 9.3.2.7, 9.3.2.10.1.1, 9.3.2.10.2.4, Table Amd.1-28, 9.3.2.7, 9.3.2.10.1.1, 9.3.2.10.2.4, Table Amd.1-28, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.2, 9.3.2.3, 9.3.2.4.7, 9.3.2.10.3.4, Table Amd.1-16, Fig. Amd.1-22, Fig. Amd.1-26, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.2, 9.3.2.3, 9.3.2.4.7, 9.3.2.10.3.4, Table Amd.1-16, Fig. Amd.1-22, Fig. Amd.1-26, 9.3.2.2, 9.3.2.3, 9.3.2.4, 9.3.2.4.1, 9.3.2.4.2, 9.3.2.4.8, 9.3.2.5, 9.3.2.8, 9.3.2.9, 9.3.2.10, 9.3.2.10.1.1, 9.3.2.10.2.1, 9.3.2.10.2.2, 9.3.2.10.2.3, 9.3.2.10.2.4, Table Amd.1-18, Table Amd.1-19, Table Amd.1-20, Fig. Amd.1-21, Fig. Amd.1-22, 9.3.2.10, Table Amd.1-18, 9.3.2.9, Fig. Amd.1-22, 9.3.2.2, 9.3.2.3, 9.3.2.4, 9.3.2.4.1, 9.3.2.4.2, 9.3.2.4.8, 9.3.2.5, 9.3.2.8, 9.3.2.9, 9.3.2.10.1.1, 9.3.2.10.2.1, 9.3.2.10.2.2, 9.3.2.10.2.3, 9.3.2.10.2.4, Table Amd.1-18, Table Amd.1-19, Table Amd.1-20, Fig. Amd.1-21, Fig. Amd.1-22, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.1.1, 9.3.2.10.2.4, 9.3.2.10.3.4, 9.3.2.10.3.5, Table Amd.1-39, Fig. Amd.1-22, Fig. Amd.1-26, Fig. Amd.1-27, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.1.1, 9.3.2.10.2.4, 9.3.2.10.3.4, 9.3.2.10.3.5, Table Amd.1-39, Fig. Amd.1-22, Fig. Amd.1-26, Fig. Amd.1-27, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.1.1, 9.3.2.10.2.4, 9.3.2.10.3.4, 9.3.2.10.3.5, Table Amd.1-39, Fig. Amd.1-22, Fig. Amd.1-26, Fig. Amd.1-27, 9.3.2.1.1.1, 9.3.2.1.2.1, 9.3.2.4.7, 9.3.2.10.1.1, 9.3.2.10.2.4, 9.3.2.10.3.4, 9.3.2.10.3.5, Table Amd.1-39, Fig. Amd.1-22, Fig. Amd.1-26, Fig. Amd.1-27, 9.3.2.10.1.1, 9.3.2.10.2.4, 9.3.1.3.2.3, 9.3.1.3.3, Table Amd.1-12, Fig. Amd.1-15, Fig. Amd.1-16, 9.3.2.2, 9.3.2.3, 9.3.2.10, 9.3.2.10.1.1, Table Amd.1-18, Table Amd.1-19, Table Amd.1-20, Fig. Amd.1-21, 9.3.2.2, 9.3.2.3, 9.3.2.10, 9.3.2.10.1.1, Table Amd.1-18, Table Amd.1-19, Table Amd.1-20, Fig. Amd.1-21, 9.3.2.1

Atmel Germany GmbH ATTN: Dr. Bertram Koch Patent Division IPM Theresienstr. 2 74072 Heilbronn Germany Tel: +49(7131)67-3254 Fax: +49(7131)67-2789 Email: Bertram.Koch@hno.atmel.com	10050878.2-31 DE, 09/929703 US, 1508/01 CH, 0113094 FR, 10138217.0 DE, 10/353298 US, 02767283.1-2210 FR, 02767283.1-2210 FI, 50202435.6-08 DE, 10335009.8 DE, 10/896670 Us, 04017298.3-1237 EP, 200410054468.5 CN, 102004019309.6 DE, 05007835.1-2210 EP, 11/104435 US, 102004014563.6 DE, 11/087871 us, 200510059501.8 CN, 10158442.3-09 DE, 10256099.4 DE, 02026665.6.1248 EP, 10/308248 US, 10301451.9 DE, 10/753859 US, 04000202.41248 EP, 2004-4143 JP, 200410002008.8 CN, 10325399.8 DE, 0 012309 3-241 E, 10/855866 US, 2004-157635 JP, 10357665.7 DE, 10/753849 US, 04000203.2-2207 EP, 200410002047.8 CN, 2004-5689 JP, 10356259.1-35 DE, 11/002852 US, 200410098306.1 CN, 102005032590.4- 35 DE, PCT/EP2005/007 496 WO, 102004007106.3 DE, 11/058056 US, 200510008328.9 CN	6.3.1.2.2, 6.3.1.2.3, 6.3.1.2.5, 6.3.1.2.8, 6.3.1.5, 6.3.1.3.1, 6.3.1.3, 6.3.2.10.3, 6.3.2.2, 6.3.2.3
BTG International Ltd ATTN: Mr Mathew Frankel Senior Legal Advisor 10 Fleet Place Limeburner Lane London EC4M 7SB UK Tel: +44 20 7575 0000 Fax: +44 20 7575 0010 Website: www.btgplc.com Email: info@btgplc.com	CHINA 98117396.9; EUROPE 98306849.5; JAPAN 241748.98; UNITED STATES OF AMERICA 09/138560; SOUTH AFRICA 987635 ; CHINA 98117442.6; CHINA 200410045765.3; EUROPE 98306943.6; JAPAN 10-243364; UNITED STATES OF AMERICA 09/143711; SOUTH AFRICA 987838; CHINA 98119594.6; JAPAN 272945/98; UNITED STATES OF AMERICA 09/160354; SOUTH AFRICA 988956; CHINA 99111121.4; EUROPE 99304836.2; JAPAN 171775/1999; UNITED STATES OF AMERICA 09/334151; UNITED STATES OF AMERICA 10/827814; SOUTH AFRICA 985286; SOUTH AFRICA 994046; AUSTRIA 99308924.2; BELGIUM 99308924.2; SWITZERLAND 99308924.2; CHINA 99122449.3; GERMANY 99308924.2; EUROPE 99308924.2; SPAIN 99308924.2; FRANCE 99308924.2; ITALY 99308924.2; JAPAN 318426/1999; NETHERLANDS 99308924.2; UNITED STATES OF AMERICA 435467; CHINA 00104540.7; EUROPE 00300753.1; JAPAN 2000-023891; UNITED STATES OF AMERICA 09/495456; SOUTH AFRICA 2000/0668; CHINA 00121458.6; EUROPE 00306298.1; JAPAN 2000-222353; UNITED STATES OF AMERICA 09/624006; SOUTH AFRICA 2000/3699; EUROPE 01304980.4; JAPAN 2001-177175; UNITED STATES OF AMERICA 09/877438; SOUTH AFRICA 2001/4484; CHINA 01115964.2; EUROPE 01305116.4; JAPAN 2001-177567; UNITED STATES OF AMERICA 09/878403;	???

	SOUTH AFRICA 2001/4921; CHINA 01129277.6; EUROPE 01305265.9; JAPAN 184904/2001; UNITED STATES OF AMERICA 09/881741; SOUTH AFRICA 2001/4922; CHINA 01145439.3; EUROPE 01310061.5; JAPAN 366552/2001; UNITED STATES OF AMERICA 09/996937; SOUTH AFRICA 2001/9960; EUROPE 02782413.5; SOUTH AFRICA 2004/4464; CHINA ; EUROPE ; JAPAN ; UNITED STATES OF AMERICA ; SOUTH AFRICA 2005/07600 CHINA 1123139; UNITED STATES OF AMERICA 6054925; SOUTH AFRICA 987635; UNITED STATES OF AMERICA 6367697; SOUTH AFRICA 987838; CHINA ZL981195994.6; UNITED STATES OF AMERICA 6198381; SOUTH AFRICA 988956; CHINA 99111121.4; UNITED STATES OF AMERICA 6724895; SOUTH AFRICA 994046; AUSTRIA 1001366; BELGIUM 1001366; SWITZERLAND 1001366; GERMANY 69923645.2; EUROPE 1001366; SPAIN 1001366; FRANCE 1001366; ITALY 1001366; NETHERLANDS 1001366; UNITED STATES OF AMERICA 6480143; UNITED STATES OF AMERICA 6346922; SOUTH AFRICA 2000/0668; UNITED STATES OF AMERICA 6388630; SOUTH AFRICA 2000/3699; UNITED STATES OF AMERICA 6867687 ; SOUTH AFRICA 2001/4484; SOUTH AFRICA 2001/4921; UNITED STATES OF AMERICA 6891466 ; SOUTH AFRICA 2001/4922; CHINA ZL01145439.3; UNITED STATES OF AMERICA 6870460; SOUTH AFRICA 2001/9960	
Impinj, Inc. ATTN: Todd E. Humes, CTO ATTN: Gregory T. Kavounas Sr. P.C. Impinj Patent Licensing Department 701 N. 34th Street, Suite 300 Seattle, WA 98103 USA Tel: +1(206)517-5300 Fax: +1(206)517-5262 Email: todd.humes@impinj.com / greg.kavounas@impinj.com	[USA S/N 10 / 915,930]; [PCT / US 2005/ 028180]; [USA S/N 10 / 890,662]; [EP S/N 5103959.2]; [USA S/N 10 / 824,049]; [PCT / US 2004/ 037668]; [USA S/N 10 / 967,996]; [USA S/N 10 / 985,518]; [PCT / US 2004/ 037387]; [USA S/N 11 / 031,459]; [USA S/N 11 / 031,471]; [USA S/N 11 / 033,028]	9.3.1.2.3; Figure Amd. 1-4; 9.3.1.2.8; Figure Amd. 1-7; Annex J; 9.3.1.3 and subsections (especially 9.3.1.3.2 and 9.3.1.3.3); 9.3.2.10.2.1; 9.3.1.2.8; Figure Amd. 1-7; 9.3.1.3.2.2; Figure Amd. 1-14; Annex J; 9.3.1.3 and subsections (especially 9.3.1.3.2 and 9.3.1.3.3); 9.3.2.10.2.1; 9.3.1.2.8; Figure Amd. 1-7; Annex D; Table D.1; 9.3.2.1; 9.3.2.10.1.1; Table Amd.1-19; 9.3.2.10.3.2; Table Amd. 1-32; 9.3.2.10.3.3; Table Amd. 1-34; 9.3.2.10.3.7; Table Amd. 1-44; 9.3.2.10.3.8; Table Amd. 1-46; 9.3.2.10; Table Amd. 1-18; plus all uses of the header “Code” specified in Table Amd. 1-18 throughout the specification; 2.3, 4.1.5 (“cover-coding”); 9.3.2.5; 9.3.2.9; 9.3.2.10.3.3; Table Amd. 1-34; 9.3.2.10.3.4; Table Amd. 1-36; Figure Amd.1-26; 9.3.2.10.3.6; Table Amd. 1-42; Figure Amd. 1-28

Intermec IP Corp ATTN: Ronald D. Payne Legal Department 6001 36th Avenue West Everett, WA 98203 Tel: +1(425)348.2756 Fax: +1(425)348.2703 Email: ron.payne@intermec.com	5,673,037 USA; 69530547.6 DE; 0702323 EP; EP0702323 FR; EP0702323 GB; 204748 KR; 318306 TW; 6,172,596 USA; 6,400,274 USA; 6,404,325 USA; 4,786,907 USA; 1272787 CA; 0254954 EP; 1435/1996 HK; 2644496 JP; 115902 KR 4,739,328 USA; 5,030,807 USA; 629153 AU; 2033868 CA; 0438250 EP; 0438250 GB; 5,777,561 USA; 5,828,693 USA; 97916151.0 EP; 5,850,181 USA; 5,912,632 USA; 5,995,019 USA; 6,400,274 USA; 6,429,775 USA; 6,288,629 USA; 6,812,841 USA; US03/00755 PC; 03702060.9 EP; 20050179521 USA; 20040189443 USA ; 10/662950 USA;	**Type C** 9.1.1 Physical layer Table 3: Tag:7e Subcarrier Frequency Tag:7g Subcarrier Modulation Tag:7h Duty Cycle Tag:8 Data Coding Tag:9 Bit Rate 9.3.1.3 Tag-to-interrogator (T=>R) communications 9.3.1.3.2 Data encoding 9.3.1.3.2.3 Miller-modulated subcarrier Figure Amd.1-16 — Subcarrier sequences 9.3.1.3.2.4 Miller subcarrier preamble 9.3.1.3.3 Tag supported Tari values and backscatter link rates Table Amd.1-11 — Tag-to-interrogator link frequencies Annex I Dense- and multiple-interrogator channelised signaling **Type A** 7.3.1 Tag memory organization 7.4.7 Command codes and CRC 7.8.9 Write_single_block 7.8.10 Write_multiple_blocks 7.8.11 Lock_blocks 7.8.12 Write_AFI 7.8.13 Lock_AFI 7.8.14 Write_DSFID command 7.8.15 Lock_DSFID **Type B** 8.2.4.2 Detailed Command Processing 8.2.7.6 Command codes and format 8.2.7.9.11 WRITE 8.2.7.9.12 WRITE4BYTE 8.2.7.9.13 LOCK 8.2.7.9.15 WRITE_MULTIPLE 8.2.7.9.16 WRITE4BYTE_MULTIPLE Annex B Memory mapping for Type B **Type C** 9.1.2 Tag-identification layer Table 4 — Tag inventory and access parameters P:5Write Size P:7Write Transaction Time 9.3.2.1.1 Reserved Memory 9.3.2.1.2 UII Memory 9.3.2.1.3 TID memory 9.3.2.1.4 User memory 9.3.2.1.1.1 Kill password 9.3.2.1.1.2 Access password 9.3.2.1.2.2 Protocol-control (PC) bits 9.3.2.1.2.1 CRC-16 9.3.2.1.2.3 UII for EPCglobal™ applications 9.3.2.1.2.4 UII for non-EPCglobal™ applications 9.3.2.9 Accessing individual tags Table Amd.1-17 — Access commands and tag states in which they are permitted

		9.3.2.10.3.3 Write (mandatory) 9.3.2.10.3.4 Kill (mandatory) 9.3.2.10.3.5 Lock (mandatory) 9.3.2.10.3.7 BlockWrite (optional) 9.3.2.10.3.8 BlockErase (optional) F.10 Command response: Write F.11 Command response: Kill F.12 Command response: Lock F.14 Command response: BlockWrite F.15 Command response: BlockErase **Type C** 9.1.1 Physical layer Table 3 — Tag-to-interrogator (T=>R) communications: Tag:7e Subcarrier Frequency Tag:7g Subcarrier Modulation Tag:7h Duty Cycle Tag:8 Data Coding Tag:9 Bit Rate 9.3.1.3 Tag-to-interrogator (T=>R) communications 9.3.1.3.2 Data encoding 9.3.1.3.2.3 Miller-modulated subcarrier Figure Amd.1-16 — Subcarrier sequences 9.3.1.3.2.4 Miller subcarrier preamble 9.3.1.3.3 Tag supported Tari values and backscatter link rates Table Amd.1-11 — Tag-to-interrogator link frequencies Annex I Dense- and multiple-interrogator channelised signaling **Types A, B, C** Table 2 — Interrogator-to-tag (R=>T) communications: Int:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Int:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) Table 3 — Tag-to-interrogator (T=>R) communications: Tag:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Tag:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) **Types A, B** 6.4 Frequency hopping carrier rise and fall times **Type C** 9.3.1.2.9 Frequency-hopping spread-spectrum waveform 9.3.1.2.10 Frequency-hopping spread-spectrum channelisation Annex I Dense- and multiple-interrogator channelised signaling Bibliography: ETSI EN 300 220 ETSI EN 302 208-1

		ETSI EN 302 208-2 US Code of Federal Regulations (CFR) Title 47, Chapter I, Part 15 **Types A, B, C** Table 2 — Interrogator-to-tag (R=>T) communications: Int:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Int:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) Table 3 — Tag-to-interrogator (T=>R) communications: Tag:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Tag:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) **Types A, B** 6.4 Frequency hopping carrier rise and fall times **Type C** 9.3.1.2.9 Frequency-hopping spread-spectrum waveform 9.3.1.2.10 Frequency-hopping spread-spectrum channelisation Annex I Dense- and multiple-interrogator channelised signaling Bibliography: ETSI EN 300 220 ETSI EN 302 208-1 ETSI EN 302 208-2 US Code of Federal Regulations (CFR) Title 47, Chapter I, Part 15 **Type A, B** 6.5.5 Message Format 6.5.6 Return preamble **Type C** 9.3.1.3.3 Tag supported Tari values and backscatter link rates **Type C** 9.1.1 Physical layer Table 3: Tag:7e Subcarrier Frequency Tag:7g Subcarrier Modulation Tag:7h Duty Cycle Tag:8 Data Coding Tag:9 Bit Rate 9.3.1.3 Tag-to-interrogator (T=>R) communications 9.3.1.3.2 Data encoding 9.3.1.3.2.3 Miller-modulated subcarrier Figure Amd.1-16 — Subcarrier sequences 9.3.1.3.2.4 Miller subcarrier preamble 9.3.1.3.3 Tag supported Tari values and backscatter link rates Table Amd.1-11 — Tag-to-interrogator link frequencies Annex I Dense- and multiple-

		interrogator channelised signaling CLAIMS 11-et seq **Types A, B, C** Table 2: Int:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Int:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) Table 3: Tag:1d Frequency Hop Rate (frequency-hopping [FHSS] systems) Tag:1e Frequency Hop Sequence (frequency-hopping [FHSS] systems) **Types A, B** 6.4 Frequency hopping carrier rise and fall times **Type C** 9.3.1.2.9 Frequency-hopping spread-spectrum waveform 9.3.1.2.10 Frequency-hopping spread-spectrum channelisation Annex I Dense- and multiple-interrogator channelised signaling Bibliography: ETSI EN 300 220 ETSI EN 302 208-1 ETSI EN 302 208-2 US Code of Federal Regulations (CFR) Title 47, Chapter I, Part 15 **Type A** 7.8.9 Write single block 7.8.10 Write multiple blocks 7.8.11 Lock single block 7.8.12 Write AFI 7.8.13 Lock AFI 7.8.14 Write DSFID command 7.8.15 Lock DSFID **Type B** 8.2.1.3.3 WRITE_OK 8.2.7.9.11 WRITE 8.2.7.9.12 WRITE4BYTE 8.2.7.9.13 LOCK 8.2.7.9.14 QUERY_LOCK 8.2.7.9.15 WRITE_MULTIPLE 8.2.7.9.16 WRITE4BYTE_MULTIPLE **Type C** 9.3.2.10.3.3 Write (mandatory) 9.3.2.10.3.4 Kill (mandatory) 9.3.2.10.3.5 Lock (mandatory) 9.3.2.10.3.7 BlockWrite (optional) 9.3.2.10.3.8 BlockErase (optional) Annex E State-transition tables **Type A** Tag state storage **Type B** 8.2.1.3.2 Data exchange status bit (DE_SB)

		Type C persistent memory or persistent flag 9.3.2.2 Sessions and inventoried flags 9.3.2.3 Selected flag 9.3.2.7 Selecting tag populations 9.3.2.8 Inventorying tag populations 9.3.2.10.1.1 Select (mandatory) 9.3.2.10.2.1 Query (mandatory) 9.3.2.10.2.2 QueryAdjust (mandatory) 9.3.2.10.2.3 QueryRep (mandatory) Annex E State-transition tables Annex F Command-Response Tables
Magellan Technology Pty Limited ATTN: Ms E. J. Angus Company Secretary 65 Johnston St ANNANDALE NSW 2038 Australia Tel: +61 2 9562 9800 Fax: +61 2 9518 7620 E-mail : jeana@magtech.com.au	US10/204159 USA , AU3711301 Australia , WO0165712 International PCT, EP1266458 Europe , 2003526148 Japan	6, 7, 8, 9
OMRON Corporation ATTN: Atsuhsi Hisano Business Development Group Planning Division Business Management Department Gate City Ohsaki West Tower 14F, 1-11-1, Ohsaki, Shinagawa-ku, Tokyo, 141-0032 Japan Tel: +81(3)5435-2004 Fax: +81(3)5435-2025 Email: atsushi_hisano@omron.co.jp	2949728/ Japan	6.3.1.3.3
Koninklijke Philips Electronics N.V. ATTN: Harald Röggla Triester Strasse 64 1101 Vienna Austria Tel: 43(1)60101 1469 Fax: +43(1)60101 1101 Email: harald.roeggla@philips.com	AT 401 127/ AT;EP 0 669 591 / BE, CH, De, DK, ES, FR, GB, IT, LI, NL, SE;EP 1 038 257 /AT, DE, ES, FR, GB, IT;IN /PCT/00/00034 / IN;US10/382262;JP00-561579 / JP;	2.5.1
Symbol Technologies, Inc. ATTN: Aaron Bernstein, VP of IP, Legal Department One Symbol Plaza, MS A6 Holtsville, NY 11742 USA Tel: 1-800-927-9626 Fax 1-631-738-4110 E-mail aaron.bernstein@symbol.com	6,784,813 (USA); 10/926,269 (USA); 10/932,279 (USA); 761843 (AU); 2,310,623 (CA); 98812462.9 (CN); 1031120 (DE); 1031120 (EP); 1320062 (EP); 1031120 (FR); 1031120 (GB); 1032468 (HK); 136,220 (IL); 1031120 (IT); 2000-521687 (JP); 10/688,535 (USA); 2003282941 (AU); TBA (CN); TBA (EP); TBA (JP); TBA (KR); 10/725,010 (USA); 10/073,000 (USA); 10/072,885 (USA); 10/693,687 (USA); 2003286702 (AU); TBA (CA); TBA (CN); 03777912.1 (EP); TBA (JP); 10-2005-7007116 (KR); TBA (SG); 10/927,775 (USA); US04/027999(WO); 2,503,407 (CA); TBA (EP); TBA (JP); 0480001172.5 (CN); 1020057007371 (KR); 2004269728 (AU)	Type C

TAGSYS, S.A. ATTN: Alastair McArthur, CTO 180 Chemin de Saint Lambert 13821 La Penne sur Huveaune France Tel: +33(0)4 91 27 57 01 Fax: +33(0)4 91 27 57 00 E-mail: alastair.mcarthur@tagsys.net	US 6,641,036; EP 1232471; WO 02054365; WO 0141043; EP 0953181; US 6,538,564	???
Texas Instruments Incorporated ATTN: Mrs. Scharlene Franks Contract Manager, RFID Business Group 6550 Chase Oaks Boulevard, MS 8470 Plano, Texas 75023 USA Tel: +1(214) 567-8830 Fax: +1(214) 567-2409 Email: s-franks@ti.com	NONE	Not applicable
Zebra Technologies Corporation ATTN: Eric McAlpine IP Counsel Legal Dept. 333 Corporate Woods Parkway Vernon Hills, IL 60061 USA Phone: 1.847.793.5640 Fax: 1.847.955.4514 E-mail: emcalpine@zebra.com	92/10006 AUSTRALIA; BW/A/97/00142 BOTSWANA; 2058692 CANADA; 02017523.8 EPC; 95112753.9 EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IT, LU, NL, PT, SE); 92300041.8 EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IT, LU, NL, PT, SE); 91/346821 JAPAN; 9609578-1 SINGAPORE; 92/0039 SOUTH AFRICA; 08/976949 USA; 07/816893 USA; 08/976948 USA; 326618 ARGENTINA; 93/50781 AUSTRALIA; BW/A/97/00141 BOTSWANA; PI9304761-4 BRAZIL; 2103288 CANADA; 93121410 CHINA; 01117486.9 EPC; 97104997.8 EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, NL, PT, SE); 97104998.6 EPC (AT, BE, CH, DK, DE, ES, FR, GB, GR, IE, IT, NL, PT, SE); 93309232.2 EPC (AT, BE, CH, ES, FR, GB, GR, IE, IT, NL, PT, SE); 0598624UKREG GUERNSEY; 98109976.2 HONG KONG; 93/01312 INDIA; 107636 ISRAEL; 47219/03 JAPAN; 93/289591 JAPAN; 0598624UKREG JERSEY; 93/02408 MALAYSIA; 9307214 MEXICO; 250219 NEW ZEALAND; 314270 NEW ZEALAND; 314269 NEW ZEALAND; 19934177 NORWAY; 19993092 NORWAY; 19993091 NORWAY; 93051222 RUSSIAN FEDERATION; 9609092-3 SINGAPORE; 93/8624 SOUTH AFRICA; 93/024833 SOUTH KOREA; 82109793 TAIWAN; 93003717 UKRAINE; 08/154329 USA; 08/580913 USA; 95/16530 AUSTRALIA; 93/44940 AUSTRALIA; BW/A/97/00144 BOTSWANA; 2104829 CANADA; 93118806 CHINA; 93306790.2 EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MN, NL, PT, SE); 93/00956 INDIA; 108621 ISRAEL; 93/211454 JAPAN; 93049090 RUSSIAN FEDERATION; 9608519-6	???

	SINGAPORE; 93/6267 SOUTH AFRICA; 94/03632 SOUTH KOREA; 83101248 TAIWAN; 93003224 UKRAINE; 08/665363 USA; 08/111430 USA; 99/10433 AUSTRALIA; 2310241 CANADA; 02021446.6 EPC; 05017862.3 EPC; 98952886.4 EPC (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MN, NL, PT, SE); 2000-521391 JAPAN; 504144 NEW ZEALAND; 98/10356 SOUTH AFRICA; 09/570951 USA; 98/74401 AUSTRALIA; PI9808714-2 BRAZIL; 2289207 CANADA; 98805116.8 CHINA; 04012409.1 EPC; 98921613.0 EPC (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, NL, SE); 02024117.0 EPC (BE, DE, DK, ES, FI, FR, GB, IE, IT, NL, SE); 132297 ISRAEL; 98/548931 JAPAN; 9910381 MEXICO; 337966 NEW ZEALAND; 99126440 RUSSIAN FEDERATION; 9904942-1 SINGAPORE; 98/10257 SOUTH AFRICA; 1999-7010475 SOUTH KOREA; 10/635683 USA; 09/415234 USA; PCT/GB04/004505 PCT; PCT/GB05/000110 PCT; 0400968.4 UNITED KINGDOM; PCT/GB05/000345 PCT; 0402667.0 UNITED KINGDOM; 2,148,145 CANADA; 08/430825 USA BOTSWANA BW/P03/00031; MALAYSIA MY-109809-A; BOTSWANA BW/P/03/00033; TAIWAN NI-84612; TAIWAN NI-67003; SOUTH AFRICA 98/10356; SOUTH AFRICA 98/10257; SOUTH AFRICA 93/8624; SOUTH AFRICA 93/6267; SOUTH AFRICA 92/0039; JERSEY 622; UKRAINE 37182; UKRAINE 44215; SINGAPORE 48262; SINGAPORE 48423; SINGAPORE 55818; SINGAPORE 68358; ISRAEL 107636; ISRAEL 108621; ISRAEL 132297; MEXICO 186161; INDIA 187341; INDIA 188258; NEW ZEALAND 250219; ARGENTINA 250637; NORWAY 307590; SOUTH KOREA 309651; NORWAY 310261; NORWAY 310262; NEW ZEALAND 314269; NEW ZEALAND 314270; SOUTH KOREA 316754; NEW ZEALAND 337966; EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IT, LU, NL, PT, SE) 0494114; NEW ZEALAND 504144; EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MN, NL, PT, SE) 0585132; EPC (AT, BE, CH, ES, FR, GB, GR, IE, IT, NL, PT, SE) 0598624; GUERNSEY 0598624UKREG; AUSTRALIA 656088; AUSTRALIA 658857; AUSTRALIA 670402; AUSTRALIA 676853; EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IT, LU, NL, PT, SE) 0685825; AUSTRALIA	

	745638; AUSTRALIA 764975; EPC (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, NL, PT, SE) 0789253; EPC (AT, BE, CH, DK, DE, ES, FR, GB, GR, IE, IT, NL, PT, SE) 0789254; EPC (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, NL, SE) 0983569; HONG KONG 1009295; EPC (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MN, NL, PT, SE) 1031046; EPC (BE, DE, DK, ES, FI, FR, GB, IE, IT, NL, SE) 1280092; EPC 1291671; CANADA 2058692; CANADA 2103288; CANADA 2104829; RUSSIAN FEDERATION 2126165; RUSSIAN FEDERATION 2156540; RUSSIAN FEDERATION 2238585; JAPAN 3100716; JAPAN 3290003; USA 5519381; USA 5537105; USA 5557280; USA 5680459; USA 5699066; USA 5726630; USA 5966083; USA 5995017; USA 6661336; USA 6784787; BRAZIL 9304761-4; CHINA 93118806.7; CHINA 93121410.6	

Page 1, clause 1

Replace paragraph 2 with the following paragraphs:

This part of ISO/IEC 18000 contains one mode with three types. All three types are ITF. Type A uses Pulse-Interval Encoding (PIE) in the forward link and an adaptive ALOHA collision-arbitration algorithm. Type B uses Manchester in the forward link and an adaptive binary-tree collision-arbitration algorithm. Type C uses Pulse-Interval Encoding (PIE) in the forward link and a random slotted collision-arbitration algorithm. The detailed technical differences between the three types are shown in the associated parameter tables.

This part of ISO/IEC 18000 specifies

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

— interrogator and tag operating procedures and commands, and

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

Page 1, clause 2

Replace the whole clause with the following clause:

2 Conformance

2.1 Claiming conformance

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

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

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

2.2 Interrogator conformance and obligations

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

— support at least one type of A, B or C – it may optionally support two or all three;

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

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

— operate within applicable local regulations.

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

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

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

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

— implement any command that conflicts with this part of ISO/IEC 18000, or

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

2.3 Tag conformance and obligations

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

— support at least one type of A, B or C – t may optionally support two or all three;

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

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

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

— conform to local radio regulations.

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

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

— implement any proprietary and/or custom commands as defined in clauses 7, 8 and 9 of this part of ISO/IEC 18000.

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

— implement any command that conflicts with this part of ISO/IEC 18000;

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

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

Page 2, clause 3

Insert the following paragraph as first paragraph:

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

Page 2, clause 3

Delete the following normative reference and associated footnote:

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

Page 2, clause 3

Insert the following normative references:

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

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

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

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

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

EPCglobal™ Tag Data Standards version 1.3 and above, EPCglobal™

Page 2, clause 4.1

Replace the introductory paragraph with the following:

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

Page 2, clause 4.1

Insert the following terms and definitions:

4.1.3
command set
set of commands used to explore and modify a tag population

4.1.4
continuous wave
typically 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

4.1.5
cover-coding
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, then performs a bit-wise EXOR of the data or password with the received random number, and, finally, 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

4.1.6
dense-interrogator environment
operating environment (4.1.13) within which most or all of the available channels are occupied by active interrogators

EXAMPLE 25 active interrogators operating in 25 available channels.

4.1.7
dense-interrogator mode

set of interrogator-to-tag and tag-to-interrogator signalling parameters used in dense-interrogator environments

4.1.8
EPCglobal™ Application

application whose usage denotes an acceptance of EPCglobal™ standards and policies

NOTE Compare non-EPCglobal™ Application (4.1.12).

4.1.9
inventoried flag

flag that indicates whether a tag may respond to an interrogator; tags maintain a separate **inventoried** flag for each of four sessions, where each flag has symmetric *A* and *B* values within any given session, and in which interrogators typically inventory tags from *A* to *B* followed by a re-inventory of tags from *B* back to *A* (or vice versa)

4.1.10
inventory round

period initiated by a *Query* command and terminated by either a subsequent *Query* command (which also starts a new inventory round) or a *Select* command

4.1.11
multiple-interrogator environment

operating environment (4.1.13) within which a modest number of the available channels are occupied by active interrogators

EXAMPLE 5 active interrogators operating in 25 available channels.

4.1.12
non-EPCglobal™ Application

application whose usage does not denote an acceptance of EPCglobal™ standards and policies

NOTE Compare EPCglobal™ Application (4.1.8).

4.1.13
operating environment

region within which an interrogator's RF transmissions are attenuated by less than 90 dB

NOTE 1 In free space, the operating environment is a sphere whose radius is approximately 1000 m, with the interrogator located at the centre.

NOTE 2 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 1000 m in certain directions and greater than 1000 m in other directions

4.1.14
operating procedure

tag-identification layer
set of functions and commands used by an interrogator to identify and modify tags

4.1.15
permalock or permalocked

memory location whose lock status is unchangeable (i.e. the memory location is permanently locked or permanently unlocked)

4.1.16
persistent memory or persistent flag

memory or flag value whose state is maintained during a brief loss of tag power

4.1.17
physical layer

data coding and modulation waveforms used in interrogator-to-tag and tag-to-interrogator signalling

4.1.18
random-slotted collision arbitration

collision-arbitration 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

4.1.19
session

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 (4.1.10); 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

4.1.20
single-interrogator environment

operating environment (4.1.13) within which there is a single active interrogator at any given time

4.1.21
singulation

identifying an individual tag in a multiple-tag environment

4.1.22
slot

point in an inventory round at which a tag may respond, which corresponds to the value output by a tag's slot counter

NOTE Tags reply when their slot (i.e. the value in their slot counter) is zero.

Page 2, clause 4.2

Insert the following symbols:

BLF	Backscatter-link frequency (BLF = $1/T_{pri}$ = DR/TRcal)
DR	Divide ratio
FDM	Frequency-division multiplexing
FT	Frequency tolerance
M_h	RF signal envelope ripple (overshoot)
M_{hh}	FHSS signal envelope ripple (overshoot)
M_{hl}	FHSS signal envelope ripple (undershoot)
M_{hs}	FHSS signal level during a hop
M_l	RF signal envelope ripple (undershoot)
M_s	RF signal level when OFF

Q	Slot-count parameter. *Q* is an integer in the range (0,15)
R=>T	Interrogator-to-tag (reader-to-tag)
RTcal	Interrogator-to-tag (reader-to-tag) calibration symbol
T=>R	Tag-to-interrogator (tag-to-reader)
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}	Time for an FHSS signal to settle to within a specified percentage of its final value
T_{pri}	Backscatter-link pulse-repetition interval (T_{pri} = 1/BLF = TRcal/DR)
T_r or $T_{r,10\text{-}90\%}$	RF signal envelope rise time
TRcal	Tag-to-interrogator (tag-to-reader) calibration symbol
T_s	Time for an RF signal to settle to within a specified percentage of its final value
UII	Unique Item Identifier
x_{fp}	floating-point value
$xxxx_2$	binary notation
$xxxx_h$	hexadecimal notation

Page 2, clause 4.2

Replace the following symbols:

Tari	Reference time interval for a data-0 in interrogator-to-tag signalling

Page 3, clause 4.3

Insert the following abbreviated terms:

Ciphertext	Information that is cover-coded
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
FCC	Federal Communications Commission
Handle	16-bit tag-authentication number
ITF	Interrogator-talks-first (Note: the common usage is RTF (Reader-talks-first) but the more precise term is ITF, which is used throughout the document.)
NSI	Numbering system identifier
Pivot	Decision threshold differentiating an R=>T data-0 symbol from a data-1 symbol

Plaintext	Information that is not cover-coded
ppm	parts per million
PR-ASK	Phase-reversal amplitude shift keying
RFU	Reserved for future use
RN16	16-bit random or pseudo-random number
RNG	Random or pseudo-random number generator
SSB	Single sideband
SSB-ASK	Single-sideband amplitude-shift keying
TID	Tag-identification or tag identifier, depending on context
Word	16 bits

Page 4, before clause 5

Insert the following new subclause:

4.4 Notation

This part of ISO/IEC 18000 uses the following notational conventions for type C:

— 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 signalling from an interrogator to a tag (reader-to-tag).

— The symbol, T=>R, refers to commands or signalling from a tag to an interrogator (tag-to-reader).

Page 4, clause 5.1

Replace clause 5.1 by the following clause:

5.1 General

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

Figure 1, Figure 2, and Figure 3 show the architecture of each of the three interrogator types. Figure Amd.1-1, Figure Amd.1-2, and Figure Amd.1-3 show the architecture of each of the three tag types.

Table 1 provides an overview of the three communication types showing the protocol blocks from interrogator input (demodulator) to interrogator output (modulator) independent of the communication flow.

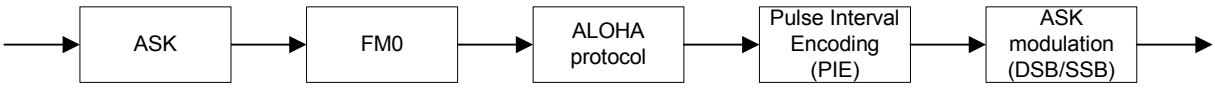


Figure 1 — Type A interrogator architecture

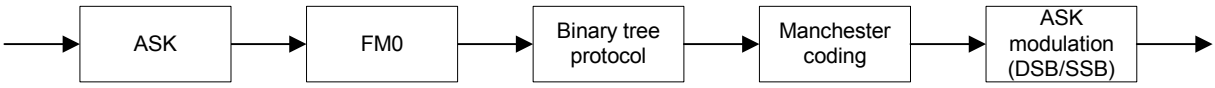


Figure 2 — Type B interrogator architecture

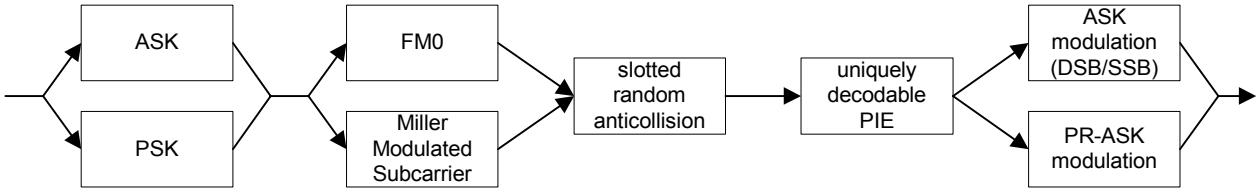


Figure 3 — Type C interrogator architecture

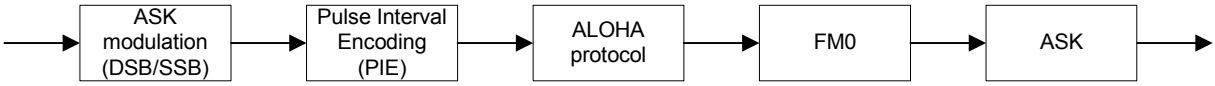


Figure Amd.1-1 — Type A tag architecture

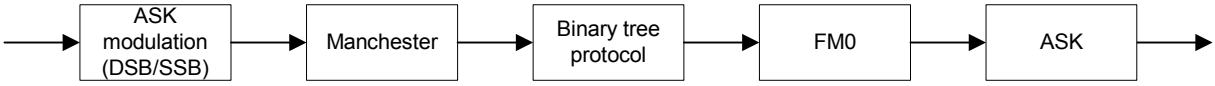


Figure Amd.1-2 — Type B tag architecture

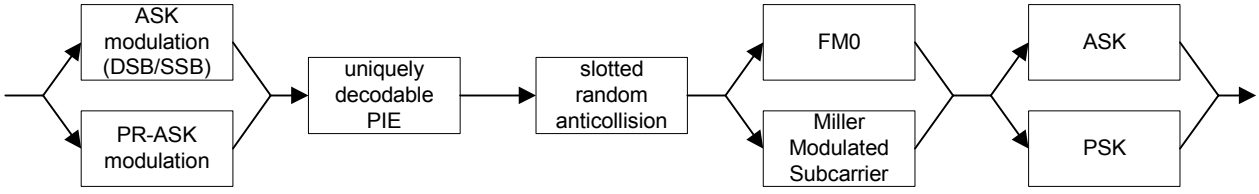


Figure Amd.1-3 — Type C tag architecture

Table 1 — Overview of Types A, B and C

Parameter	Type A	Type B	Type C
Forward link encoding	PIE	Manchester	Uniquely decodeable PIE
Modulation depth	27% to 100%	30.5% or 100%	80% to 100%
Data rate	33 kbit/s (assuming equiprobable data)	10 or 40 kbit/s (according to local regulations)	26.7 kbit/s to 128 kbit/s (assuming equiprobable data)
Return link encoding	FM0	FM0	FM0, Miller subcarrier
Collision arbitration	ALOHA / FST	Binary Tree	random slotted collision-arbitration
Tag unique identifier	64 bits (40 bit SUID)	64 bits	Variable: minimum 16 bits, maximum 496 bits
Memory addressing	Blocks up to 256 bits	Byte blocks, 1,2,3 or 4 byte writes	Bit, word (16-bit), or vendor-defined block boundaries (depending on command)
Error detection, forward link	5 bit CRC for all commands (with an additional 16-bit CRC appended for all long commands)	16-bit CRC	16-bit CRC, except a 5-bit CRC for the *Query* command
Error detection, return link	16-bit CRC	16-bit CRC	16-bit CRC, except no error check for RN16
Collision arbitration linearity	2048	Up to 2^{256}	Linear up to 2^{15} tags in the interrogator's RF field; above that number, N log(N) for tags with unique UIIs

Page 6, Table 2

Replace Table 2 with the following table:

Table 2 — Interrogator-to-tag link parameters

Ref.	Parameter Name	Description	Ref Type C
Int:1	Operating Frequency Range	**Types A and B:** 860 – 960 MHz The interrogator operating frequency range shall be determined by the radio regulations in force in a particular regulatory jurisdiction and by the type approval requirements of the particular jurisdiction. Before an interrogator may be used, it shall meet the local radio regulations of the country in which it is to be used. It is envisaged that there will be more than one version of interrogator having different frequency and power characteristics in order to comply with local regulations. NOTE Performance will vary according to bandwidth and power output permitted by local regulations. **Type C:** 860 MHz – 960 MHz, as required by local regulations	See clause 9.3.1.1
Int:1a	Default Operating Frequency	**Types A and B:** In accordance with the local radio regulations. See Int: 1 **Type C:** Determined by local radio regulations and by the radio-frequency environment at the time of the communication	See clause 9.3.1.1
Int:1b	Operating Channels (spread-spectrum systems)	**Types A and B:** In accordance with the local radio regulations. See Int: 1 **Type C:** In accordance with local regulations; if the channelisation is unregulated, then as specified	See clause 9.3.1.2.10, Annex I
Int:1c	Operating Frequency Accuracy	**Types A and B:** In accordance with the local radio regulations See Int: 1 **Type C:** As specified	See clause 9.3.1.2.1
Int:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	**Types A and B:** Not applicable for single fixed frequency or channelized Adaptive Frequency Agile systems. Where FHSS is permitted, the hop rate shall be in accordance with the local radio regulations. **Type C:** In accordance with local regulations	See clause 9.3.1.2.9

Ref.	Parameter Name	Description	Ref Type C
Int:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	**Types A and B:** In accordance with the local radio regulations. Where not specified by such regulations a pseudo-random hopping sequence shall be used that ensures an even distribution of transmissions over the available channels. **Type C:** In accordance with local regulations	See clause 9.3.1.2.9
Int:2	Occupied Channel Bandwidth	In accordance with local regulations	
Int:2a	Minimum Receiver Bandwidth	In accordance with local regulations	
Int:3	Interrogator Transmit Maximum EIRP	In accordance with local regulations	
Int:4	Interrogator Transmit Spurious Emissions	**Types A and B:** In accordance with the local radio regulations. **Type C:** As specified; tighter emission limits may be imposed by local regulations	See clause 9.3.1.2.11
Int:4a	Interrogator Transmit Spurious Emissions, In-Band (spread-spectrum systems)	**Types A and B:** In accordance with the local radio regulations. **Type C:** As specified; tighter emission limits may be imposed by local regulations	See clause 9.3.1.2.11
Int:4b	Interrogator Transmit Spurious Emissions, Out-of-Band	**Types A and B:** In accordance with the local radio regulations. **Type C:** As specified; tighter emission limits may be imposed by local regulations	See clause 9.3.1.2.11
Int:5	Interrogator Transmitter Spectrum Mask	**Types A and B:** As per Int: 2 and Int: 4a. **Type C:** As specified; tighter emission limits may be imposed by local regulations	Figure Amd.1-9, Figure Amd.1-10
Int:6	Timing	**Types A and B:** See below Int: 6x. **Type C:** As specified	See clause 9.3.1.6, Figure Amd.1-19, Table Amd.1-15
Int:6a	Transmit-to-Receive Turn-Around Time	**Types A and B:** The interrogator transmit/receive settling time shall not exceed 85 µs. **Type C:** MAX(RTcal,$10T_{pri}$) nominal	See clause 9.3.1.6, Figure Amd.1-19, Table Amd.1-15
Int:6b	Receive-to-Transmit Turn-Around Time	**Types A and B:** As determined by the communication protocol – refer Tag: 6a. **Type C:** $3T_{pri}$ minimum; $20T_{pri}$ maximum when tag is in **reply** & **acknowledged** states; no limit otherwise	See clause 9.3.1.6, Figure Amd.1-19, Table Amd.1-15
Int:6c	Dwell Time or interrogator Transmit Power-On Ramp	1500 µs, maximum settling time	See clause 9.3.1.2.6, Figure Amd.1-6, Table Amd.1-8
Int:6d	Decay Time or interrogator Transmit Power-Down Ramp	**Types A and B:** Maximum 1 ms. **Type C:** 500 µs, maximum	Figure Amd.1-6, Table Amd.1-9
Int:7	Modulation	**Types A and B:** Amplitude Modulation. **Type C:** DSB-ASK, SSB-ASK, or PR-ASK	See clause 9.3.1.2.2

Ref.	Parameter Name	Description	Ref Type C
Int:7a	Spreading Sequence (direct-sequence [DSSS] systems)	Not applicable.	
Int:7b	Chip Rate (spread-spectrum systems)	Not applicable.	
Int:7c	Chip Rate Accuracy (spread-spectrum systems)	Not applicable.	
Int:7d	Modulation Depth	**Type A**: Nominal 30 % to 100 %. See Table 11 and clause 7.1.1. **Type B**: Nominal 18% or 100% . **Type C:** 90% nominal	See clause 9.3.1.2.5, Figure Amd.1-5, Table Amd.1-7
Int:7e	Duty Cycle	**Types A and B:** In accordance with local regulations. **Type C:** 48% – 82.3% (time the waveform is high)	See clause 9.3.1.2.3, Figure Amd.1-4, Table Amd.1-7
Int:7f	FM Deviation	Not applicable.	
Int:8	Data Coding	**Type A**: Pulse Interval Encoding **Type B**: Manchester bi-phase **Type C:** PIE	See clause 9.3.1.2.3, Figure Amd.1-4
Int:9	Bit Rate	**Type A**: 33 kbit/s as constrained by the local radio regulations **Type B**: 10 kbit/s or 40 kbit/s as constrained by the local radio regulations. **Type C:** 26.7 kbit/s to 128 kbit/s (assuming equiprobable data)	See clause 9.3.1.2.4
Int:9a	Bit Rate Accuracy	**Types A and B:** 100 ppm **Type C:** +/– 1%, minimum	See clause 9.3.1.2.4
Int:10	Interrogator Transmit Modulation Accuracy	**Types A and B:** Not applicable. **Type C:** As specified	See clause 9.3.1.2.4
Int:11	Preamble	**Type A**: has no preamble **Type B**: See clause 8.1.4.3 **Type C:** Required	See clause 9.3.1.2.8
Int:11a	Preamble Length	**Type A**: not applicable. **Type B**: 9 bits. See clause 8.1.4.3. **Type C:** As specified	See clause 9.3.1.2.8
Int:11b	Preamble Waveform(s)	**Type A**: not applicable . **Type B**: See clause 8.1.4.3. **Type C:** As specified	See Figure Amd.1-7
Int:11c	Bit Sync Sequence	**Type A**: not applicable. **Type B**: see clause 8.1.4.3 **Type C:** None	
Int:11d	Frame Sync Sequence	**Types A and B:** Not applicable. **Type C:** Required	See clause 9.3.1.2.8, Figure Amd.1-7
Int:12	Scrambling (spread-spectrum systems)	Not Applicable.	
Int:13	Bit Transmission Order	MSB is transmitted first	See clause 9.3.1.4

Ref.	Parameter Name	Description	Ref Type C
Int:14	Wake-up process	**Types A and B:** Presence of an appropriate RF signal at the tag followed by a wake-up command as required by the tag type. See relevant clauses. **Type C:** As specified	See clause 9.3.1.3.4
Int:15	Polarization	**Types A and B:** Interrogator dependent. Not defined in this part of ISO/IEC 18000. **Type C:** Not specified	

Page 8, Table 3

Replace Table 3 with the following table:

Table 3 — Tag-to-interrogator link parameters

Ref.	Parameter Name	Description	Ref Type C
Tag:1	Operating Frequency Range	860 MHz – 960 MHz, inclusive	See clause 9.3.1.1
Tag:1a	Default Operating Frequency	**Types A and B:** The tag shall respond to an interrogator signal within the frequency range specified in Tag: 1. **Type C:** Tags respond to interrogator signals that satisfy Int:1a	See clause 9.3.1.1
Tag:1b	Operating Channels (spread-spectrum systems)	**Types A and B:** The tag shall respond to an interrogator signal within the frequency range specified in Tag: 1. **Type C:** Tags respond to interrogator signals that satisfy Int:1b	See clause 9.3.1.2.10
Tag:1c	Operating Frequency Accuracy	**Types A and B:** The tag shall respond to an interrogator signal within the frequency range specified in Tag: 1. **Type C:** As specified	See clause 9.3.1.3.3 Table Amd.1-11
Tag:1d	Frequency Hop Rate (frequency-hopping [FHSS] systems)	**Types A and B:** Not applicable **Type C:** Tags respond to interrogator signals that satisfy Int:1d	See clause 9.3.1.2.9
Tag:1e	Frequency Hop Sequence (frequency-hopping [FHSS] systems)	**Types A and B:** Not applicable **Type C:** Tags respond to interrogator signals that satisfy Int:1e	See clause 9.3.1.2.9
Tag:2	Occupied Channel Bandwidth	In accordance with local regulations	
Tag:3	Transmit Maximum EIRP	In accordance with local regulations	
Tag:4	Transmit Spurious Emissions	In accordance with local regulations	
Tag:4a	Transmit Spurious Emissions, In-Band (spread spectrum systems)	In accordance with local regulations	
Tag:4b	Transmit Spurious Emissions, Out-of-Band	In accordance with local regulations	
Tag:5	Transmit Spectrum Mask	In accordance with local regulations	

Ref.	Parameter Name	Description	Ref Type C
Tag:6a	Transmit-to-Receive Turn-Around Time	**Type A:** The tag shall open its receive command window within 2 bit periods of the end of its response. **Type B:** 400 µs **Type C:** $3T_{pri}$ minimum, $20T_{pri}$ maximum in **reply** & **acknowledged** states; no limit otherwise	See clause 9.3.1.6, Figure Amd.1-19, Table Amd.1-15
Tag:6b	Receive-to-Transmit Turn-Around Time	**Type A :** Range from 150 to 1150 µs (see clause 7.5.3 and Table 28) **Type B :** Range 85 to 460 µs (see clause 8.1.5.2) **Type C:** MAX(RTcal,$10T_{pri}$) nominal	See clause 9.3.1.6, Figure Amd.1-19, Table Amd.1-15
Tag:6c	Dwell Time or Transmit Power-On Ramp	**Types A and B:** Not applicable. **Type C:** Receive commands 1.5 ms after power-up	See clause 9.3.1.3.4
Tag:6d	Decay Time or Transmit Power-Down Ramp	Not applicable.	
Tag:7	Modulation	**Types A and B:** Bi-state amplitude modulated backscatter. **Type C:** ASK and/or PSK modulation (selected by tag)	See clause 9.3.1.3.1
Tag:7a	Spreading Sequence (direct sequence [DSSS] systems)	Not applicable.	
Tag:7b	Chip Rate (spread spectrum systems)	Not applicable.	
Tag:7c	Chip Rate Accuracy (spread spectrum systems)	Not applicable.	
Tag:7d	On-Off Ratio	**Types A and B:** The tag Delta RCS (Varying Radar Cross Sectional area) affects system performance. A typical value is greater than $0.005m^2$. **Type C:** Not specified	
Tag:7e	Subcarrier Frequency	**Types A and B:** Not applicable. **Type C:** 40 kHz to 640 kHz	See clause 9.3.1.3.3 Table Amd.1-11
Tag:7f	Subcarrier Frequency Accuracy	**Types A and B:** Not applicable. **Type C:** As specified	Table Amd.1-11
Tag:7g	Subcarrier Modulation	**Types A and B:** Not applicable. **Type C:** Miller, at the data rate	See clause 9.3.1.3.2.3, Figure Amd.1-16
Tag:7h	Duty Cycle	**Types A and B:** The tag shall transmit its response when commanded to do so by the interrogator. **Type C:** FM0: 50%, nominal Subcarrier: 50%, nominal	See clause 9.3.1.3.2.1, 9.3.1.3.2.3
Tag:7i	FM Deviation	Not applicable.	
Tag:8	Data Coding	**Types A and B:** Bi-phase space (FM0) **Type C:** Baseband FM0 or Miller-modulated subcarrier (selected by the interrogator)	See clause 9.3.1.3.2

Ref.	Parameter Name	Description	Ref Type C
Tag:9	Bit Rate	**Types A and B:** Typical 40 kbit/s or 160 kbit/s (subject to tag clock tolerance see Table 9), The return bit rate selection for 160 kbit/s for Type B is defined in clause 8.1.4.4.5. **Type C:** FM0: 40 kbps to 640 kbps Subcarrier modulated: 5 kbps to 320 kbps	See clause 9.3.1.3.3 Table Amd.1-11
Tag:9a	Bit Rate Accuracy	**Types A and B:** +/- 15% (refer to Table 9) **Type C:** Same as Subcarrier Frequency Accuracy; see Tag:7f	See clause 9.3.1.3.3, Table Amd.1-11, Table Amd.1-12
Tag:10	Tag Transmit Modulation Accuracy (frequency-hopping [FHSS] systems	Not applicable.	
Tag:11	Preamble	**Types A and B:** The preamble is defined in clause 6.5.6 **Type C:** Required	See clause 9.3.1.3.2.2, See clause 9.3.1.3.2.4
Tag:11a	Preamble Length	**Types A and B:** 16 bits made up of a quiet period, followed by sync, followed by a code violation followed by an orthogonal code. **Type C:** As specified	See Table Amd.1-11, See Figure Amd.1-18
Tag:11b	Preamble Waveform	**Types A and B:** Bi-phase encoded data '1'. **Type C:** As specified	See Table Amd.1-11, See Figure Amd.1-18
Tag:11c	Bit-Sync Sequence	**Types A and B:** Included in the preamble. **Type C:** None	
Tag:11d	Frame-Sync Sequence	**Types A and B:** Included in the preamble. **Type C:** None	
Tag:12	Scrambling (spread-spectrum systems)	Not applicable.	
Tag:13	Bit Transmission Order	MSB is transmitted first	See clause 9.3.1.4
Tag:14	Reserved	Deliberately left blank.	
Tag:15	Polarization	**Types A and B:** Product design feature. Not defined in this part of ISO/IEC 18000. **Type C:** Tag dependent; not specified by this document	
Tag:16	Minimum Tag Receiver Bandwidth	**Types A and B:** 860 – 960 MHz **Type C:** Tag dependent; not specified by this document.	

Page 9, Table 4

Replace Table 4 with the following table:

Table 4 — Protocol parameters

Ref.	Parameter Name	Description	Ref Type C
P:1	Who talks first	Interrogator	See clause 9.3
P:2	Tag addressing capability	**Type A** see clause 7.3.1 **Type B** see clause 8.2.2 **Type C:** As specified	See clause 9.3.2.1
P:3	Tag UII	**Types A and B:** Contained in tag memory and accessible by means of a command. **Type C:** Contained in Tag memory	See clause 9.3.2.1
P:3a	UII Length	**Type A:** 64 bits. An SUID of 40 bits is used during census or collision arbitration transactions. **Type B:** 64 bits **Type C:** As specified	9.3.2.1.2.2
P:3b	UII Format	See clause 7.2.1 for **type A**. See clause B.2 for **type B**. **Type C:** NSI < 100_h: As specified in EPCglobal™ Tag Data Standards (Version 1.3 and above), NSI ≥ 100_h: As specified in ISO/IEC 15961	9.3.2.1.2.2 9.3.2.1.2.3 9.3.2.1.2.4
P:4	Read size	**Type A:** Addressable in blocks of up to 256 bits, but always an integer multiple of bytes. **Type B**: Addressable in byte blocks. **Type C:** Multiples of 16 bits	See clause 9.3.2.10.3.2 Table Amd.1-32
P:5	Write Size	**Type A:** Addressable in blocks of up to 256 bits, but always an integer multiple of bytes. **Type B:** Addressable in byte blocks. Writing in blocks of 1, 2, 3 or 4 bytes. See details in relevant clauses. **Type C:** Multiples of 16 bits	See clause 9.3.2.10.3.3, Table Amd.1-34, See clause 9.3.2.10.3.7, Table Amd.1-44
P:6	Read Transaction Time	**Types A and B:** A single tag can typically be identified and have its first 128 bits of user memory read in less than 10 ms. This time may vary depending on the data rate used as constrained by the local radio regulations. **Type C:** Varied with R=>T and T=>R link rate and number of bits being read	See clause 9.3.2.10.3.2
P:7	Write Transaction Time	**Types A and B:** Once a tag has been identified and selected, a 32-bit data block can typically be written in less than 20 ms. This time may vary depending on the data rate used as constrained by the local radio regulations. **Type C:** 20 ms (maximum) after end of *Write* command	See clause 9.3.2.10.3.3, See clause 9.3.2.10.3.7, Figure Amd.1-25

Ref.	Parameter Name	Description	Ref Type C
P:8	Error detection	**Type A:** Interrogator to tag: 16-bit commands: CRC-5. Commands of more than 16 bits have an additional CRC-16, bit protection. Tag to interrogator: CRC-16 **Type B:** Interrogator to tag: CRC-16 Tag to interrogator: CRC-16 **Type C:** Interrogator-to-tag: *Select* command: CRC-16 *Query* command: CRC-5 Other Inventory commands: Command length Access commands: CRC-16 Tag-to-interrogator: PC, UII: CRC-16 RN16: None or CRC-16 (varies with command) handle: CRC-16 All other: CRC-16	See clause 9.3.2.10 and its subsections
P:9	Error correction	**Types A and B:** No forward error correction code used. Errors are handled by signalling an error to the interrogator that then repeats its last transmission. **Type C:** None	
P:10	Memory size	**Types A and B:** No minimum user memory size is specified, but if user memory is provided it shall be an integer multiples of 4 bytes. **Type C:** Tag dependent, extensible (size is neither limited nor specified by this document)	
P:11	Command structure and extensibility	**Type A:** Several command codes are reserved for future use. In addition, a Protocol extension flag allows for further extensions. **Type B:** Several command codes are reserved for future use. **Type C:** As specified	Table Amd.1-18

Page 10, Table 5

Replace Table 5 with the following table:

Table 5 — Anti-collision parameters

Ref.	Parameter Name	Description	Ref Type C
A:1	Type (Probabilistic or Deterministic)	Probabilistic	See clause 9.3.2.6
A:2	Linearity	**Type A**: Essentially linear using adaptive slot allocation up to 256 slots for 250 tags in an interrogator zone. **Type B**: Essentially linear up to 2^{256} tags depending on size of data content. **Type C:** Linear up to 2^{15} tags in the interrogator's RF field; above that number, NlogN for tags with unique UIIs	See clause 9.3.2.8
A:3	Tag inventory capacity	**Types A and B:** The algorithm permits the reading of not less than 250 tags in the reading zone of the interrogator. **Type C:** Unlimited for tags with unique UIIs	See clause 9.3.2.8

Page 11, Figure 4

Move the figure into clause 6.2 as it is referenced in clause 6.2.

Page 11, Table 6
Move the table into clause 6.2 as it is referenced in clause 6.2.

Page 13, Clause 6.5.3

Replace the wording of clause 6.5.3 with the following wording:

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

Page 13, Table 9

Replace Table 9 with the following table:

Table 9 — Return link parameters

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

Page 19, Table 12

Replace Table 12 with the following table:

Table 12 — Reference interval timing

Tari	Stability
10 to 20 µs	±100 ppm

Page 28, Table 20

Replace the text in row"7", column "Round size" with the following text:

1024

Page 28, Table 21

Replace Table 21 with the following modified table:

Table 21 — Command classes

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

Page 28, Clause 7.4.6.3

Replace the first paragraph with the following:

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

Page 29, Clause 7.4.6.5

Replace the first paragraph with the following:

The proprietary command codes range from '3A' to '3F'. Proprietary commands may be enabled by this part of ISO/IEC 18000, but they shall not be specified in this part of ISO/IEC 18000. A proprietary command shall not solely duplicate the functionality of any mandatory or optional command defined in this part of ISO/IEC 18000 by a different method.

Page 30, Table 22

Replace the text in row "Next_Slot", column "Comments" with the following text:

The signature may be one of 16 values between 0x00 and 0x0F as transmitted by the tag in its last transmission.

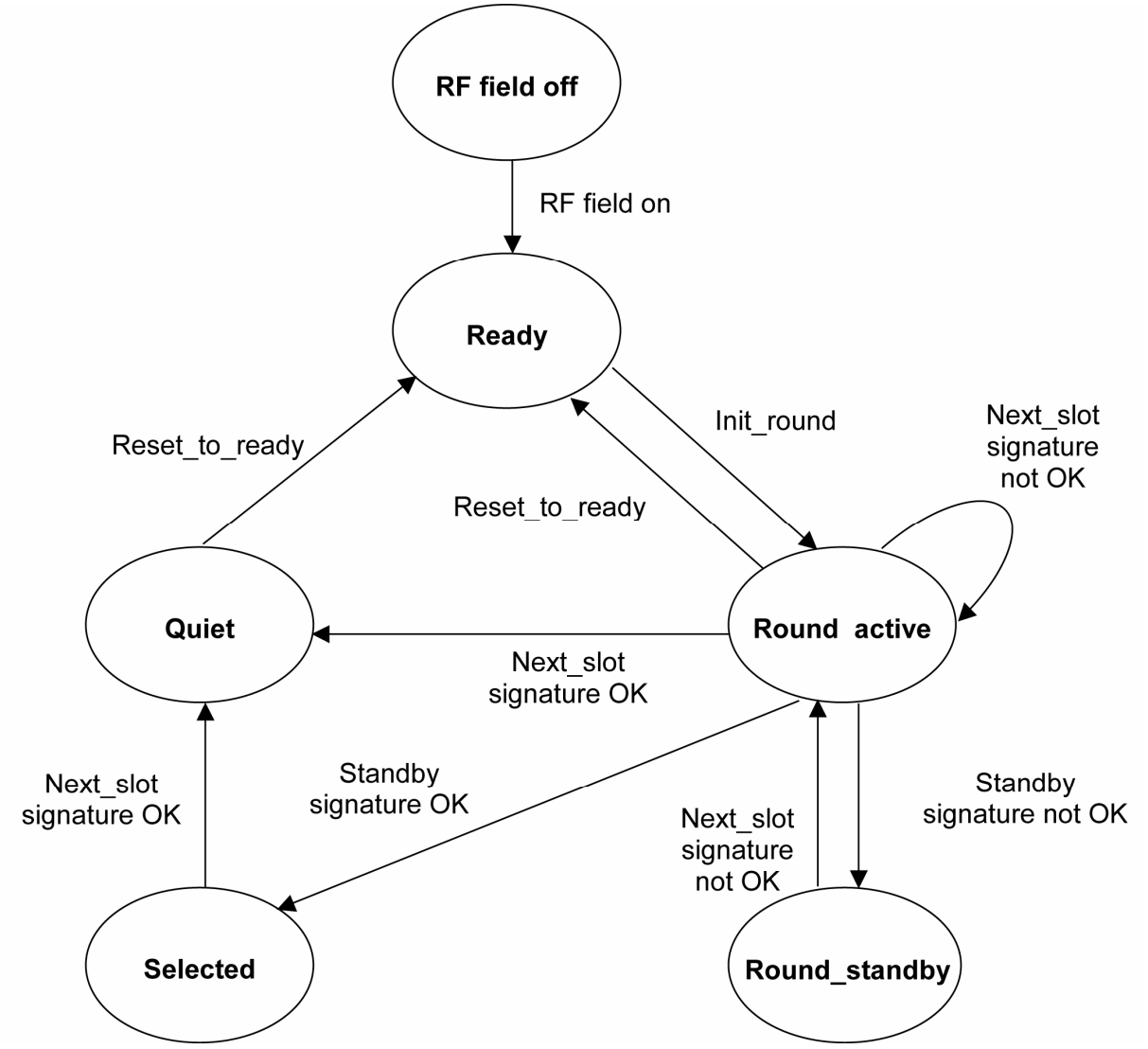
Page 30, Table 22

Insert the following row:

Command 6 bits	Op code	Parameter / flags 4 bits		CRC-5	Extended parameters	CRC-16	Comments
RFU	0x38	RFU	RFU	5 bits			
Init_Fast_Slot	0x39	SUID 1 bit	Round Size 3 bits	5 bits			SUID=0 tag responds with 1st page of user data; SUID=1 tag responds with SUID

Page 35, Figure 16

Replace Figure 16 with the following new Figure 16.



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

Figure 16 — Tag state transition diagram

Page 37, Clause 7.4.11

Insert the following text at the end of clause 7.4.11.

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

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

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

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

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

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

Page 39, Clause 7.5.5.3

Replace clause 7.5.5.3. with the following text:

7.5.5.3 Tag

When a tag, in the Round active state (and not in the FST mode), has just responded in the current slot and receives a Next_slot command, which was not received within the window specified in clause 7.5.5, the tag shall not move to the quiet state, but shall remain in the Round active state and shall increment its slot counter as specified in Clause 7.7.2 and Table 31.

When a tag in the Round active state, has just responded in the current slot and receives a Standby_round command, which was not received within the window specified in clause 7.5.5, the tag shall not move to the selected state, but shall transition to the Round_standby state as specified in Clause 7.7.3 and Table 33.

Page 41, Clause 7.7.2

Insert the following paragraph at the end of clause 7.7.2:

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

Page 41, Clause 7.7.3

Replace clause 7.7.3 with the following text:

7.7.3 Standby_round

Command code = '04'

The Standby_round command has two functions:

- It acknowledges a valid response from a tag and instructs this tag to enter the Selected state if the tag supports the optional Selected state (individual commands can then be sent to this tag such as Read and Write), or

- It instructs all other tags in the Round_active state to enter the Round_standby state. For these tags, their participation in the round is suspended. tags that do not support the optional Selected state will move from the Round_active state to the Round_standby state irrespective of whether the Standby_round command acknowledged a valid response from the tag or not. The round will be resumed by a Next_slot command, or a Close_slot command. In addition, a new round will be initiated by tags in the Round_standby state by a New_round command, an Init_round command or an Init_round_all command.

The command contains:

The Standby_round command code, and

The tag signature.

No flags are used by this command.

Table 32 shows the Standby_round command format.

Table 32 — Standby_round command format

Protocol Extension	Standby _round	Tag signature	CRC-5
1 bit	6 bits	4 bits	5 bits

Table 33 — Tag state transition for Standby_round

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

Page 47, Clause 7.8.3

Insert the following paragraph after the first paragraph:

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

Page 48, Table 45

Replace Table 45 with the following table:

Table 45 — Tag state transition for Close_slot

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

Page 76, Clause 7.8.16

Insert the following clause after clause 7.8.16:

7.8.17 Init_Fast_Slots

Command code = '39'

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

The command contains:

The Init_Fast_Slots command code of 6 bits,

The SUID Flag of 1 bit, and

The Round size of 3 bits.

Table Amd.1-1 shows the Init_Fast_Slots command format.

Table Amd.1-1 — Init_Fast_Slots command format

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

The response, as shown in Table Amd.1-2 or Table Amd.1-3 shall contain:

- The DSFID if the SUID flag is set in the command,
- The tag signature,
- The tag type (battery-less or battery-assisted),
- The battery status flags,
- The SUID if the SUID flag is set in the command, and
- The first n bits of the tag memory if the SUID flag is NOT set in the command.

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

Table Amd.1-2— Init_Fast_Slots response format when the SUID flag is NOT set

Preamble	Flags see Table 24	Tag type see Table Amd.1-4	Battery status see Table Amd.1-5	Signature	Random number See NOTES 1 and 2	First n bits of memory See NOTE 3	CRC-16
	2 bits	1 bit	1 bit	4 bits	8 bits	n = 32, 64, 96 or 128 bits	16 bits

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

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

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

This International Standard does not specify how the 8 bit random number shall be generated (It could for instance be the last 8 significant bits of the UID, or a random number, etc.).

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

Table Amd.1-3— Init_Fast_Slots response format when the SUID flag is set

Preamble	Flags see Table 24	Tag type see Table Amd.1-4	Battery status see Table Amd.1-5	Signature	DSFID	SUID	CRC-16
	2 bits	1 bit	1 bit	4 bits	8 bits	40 bits	16 bits

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

Table Amd.1-4 shows the Tag type field, while Table Amd.1-5 shows the Battery Status field.

Table Amd.1-4 — Tag type (battery-assisted or not)

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

Table Amd.1-5 — Battery status (for battery-assisted tag)

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

The tag shall perform the actions specified in Table Amd.1-6.

Table Amd.1-6 — Tag state transition for Init_Fast_Slots

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

NOTE The slots are numbered from 1 to Round_size.

Page 111, Clause 8.2.7.9.1

Add the following paragraph after the first paragraph:

For memory lock functionality a tag shall provide the opportunity to mark an address lockable. An ADDRESS is marked lockable by commands as described in this clause. No ADDRESS shall be marked lockable after the tag has been in the POWER-OFF state. A tag shall not support more than one ADDRESS to be marked lockable at the same time.

Page 114, Table 154

Replace Table 154 with the following table:

Table 154 — READ_VARIABLE response in the case of NO error

Preamble	(Length + 1) * BYTE_DATA	CRC-16
	(Length + 1) * 8 bits	16 bits

Page 118, Clause 8.2.7.9.14

Replace the second paragraph with the following paragraph:

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

Page 121, Clause 8

Insert the following new clause after Clause 8:

9 Type C

9.1 Protocol overview

9.1.1 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 to a mandatory or optional command using full-duplex communications.

9.1.2 Tag-identification layer

An interrogator manages tag populations using three basic operations:

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.

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, UII, and CRC-16 from the tag. Inventory comprises multiple commands. An inventory round operates in one and only one session at a time.

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.

9.2 Command types and command structure

9.2.1 General

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

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

9.2.2 Mandatory

Conformant tags shall support all mandatory commands.

Conformant interrogators shall support all mandatory commands.

9.2.3 Optional

Tags may or may not support optional commands.

Interrogators may or may not support optional commands.

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

9.2.4 Custom

Custom commands are not specified in this document.

An interrogator shall issue a custom command only after (i) singulating a tag, and (ii) 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 tag ID.

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

9.2.5 Proprietary

Proprietary commands are not specified in this document.

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

9.3 Description of operating procedure

The operating procedure defines the physical and logical requirements for an ITF, random-slotted collision arbitration, RFID system operating in the 860 MHz – 960 MHz frequency range.

9.3.1 Signalling

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

9.3.1.1 Operational frequencies

Tags shall receive power from and communicate 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 that are claimed to operate in dense-interrogator environments shall support, but are not required to always use, the optional dense-interrogator mode described in Annex I.

9.3.1.2 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 4.1. The interrogator sets the data rate by means of the preamble that initiates the round.

The high values in Figure Amd.1-4, Figure Amd.1-5, Figure Amd.1-6, Figure Amd.1-7 and Figure Amd.1-8, correspond to emitted CW (i.e. an interrogator delivering power to the tag or tags) whereas the low values correspond to attenuated CW.

9.3.1.2.1 Interrogator frequency accuracy

Interrogators that claim to operate in single- or multiple-interrogator environments shall have a frequency accuracy that meets local regulations. Interrogators that are claimed to operate 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) while transmitting, unless local regulations specify tighter accuracy, in which case the interrogator frequency accuracy shall meet the local regulations.

9.3.1.2.2 Modulation

Interrogators shall communicate using DSB-ASK, SSB-ASK, or PR-ASK modulation, detailed in Annex J. Tags shall demodulate all three modulation types.

9.3.1.2.3 Data encoding

The R=>T link shall use PIE, shown in Figure Amd.1-4. Tari is the reference time interval for interrogator-to-tag signalling, and is the duration of a data-0. High values represent transmitted CW; low values represent attenuated CW. Pulse modulation depth, rise time, fall time, and PW shall be as specified in Table Amd.1-7, 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, Tari, data-0 length, and data-1 length for the duration of an inventory round. The RF envelope shall be as specified in Figure Amd.1-5.

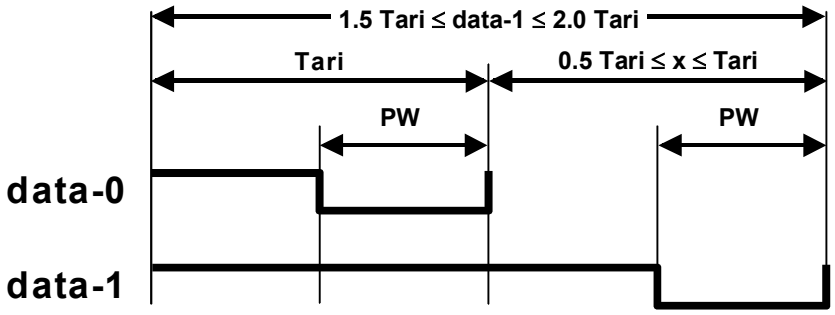


Figure Amd.1-4 — PIE Symbols

9.3.1.2.4 Tari values

Interrogators shall communicate using Tari values in the range of 6.25µs to 25µs. Interrogator compliance shall be evaluated using at least one Tari value between 6.25µs and 25µs with at least one value of the parameter *x*. The tolerance on all parameters specified in units of Tari shall be +/–1%. The choice of Tari value and *x* shall be in accordance with local radio regulations.

9.3.1.2.5 R=>T RF envelope

The R=>T RF envelope shall comply with Figure Amd.1-5 and Table Amd.1-7. The electric field strength A is the maximum amplitude of the RF envelope. Tari is defined in Figure Amd.1-4. 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 9.3.1.2.7).

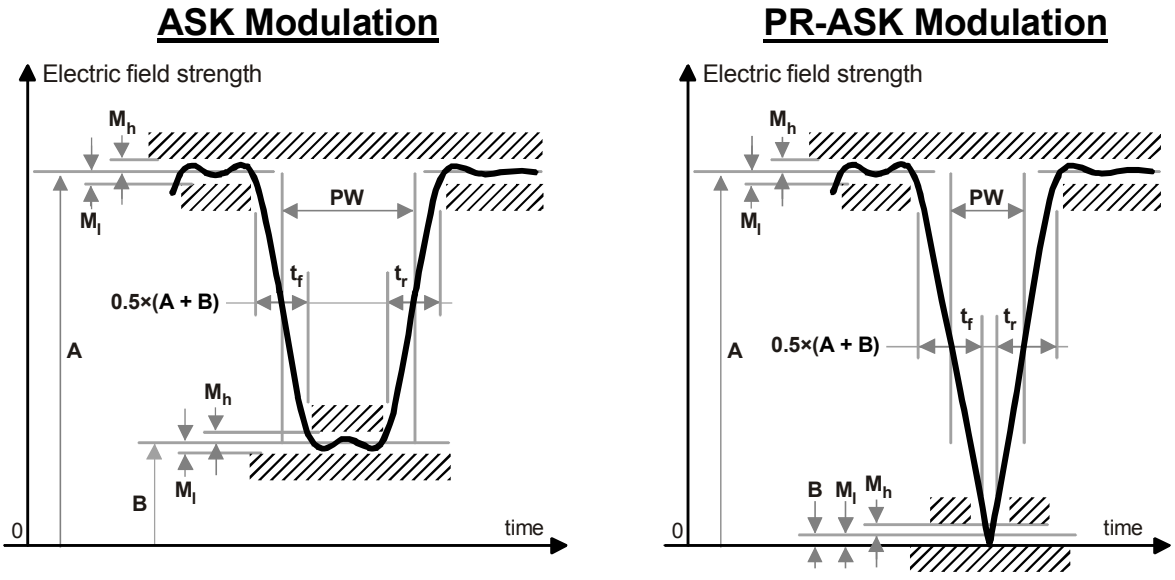


Figure Amd.1-5 — Interrogator-to-tag RF Envelope

Table Amd.1-7 — RF envelope parameters

Tari	Parameter	Symbol	Minimum	Nominal	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

9.3.1.2.6 Interrogator power-up waveform

The interrogator power-up RF envelope shall comply with Figure Amd.1-6 and Table Amd.1-8. 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 Amd.1-6 during interval T_s. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table Amd.1-8 (i.e. before T_s).

9.3.1.2.7 Interrogator power-down waveform

The interrogator power-down RF envelope shall comply with Figure Amd.1-6 and Table Amd.1-9. 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.

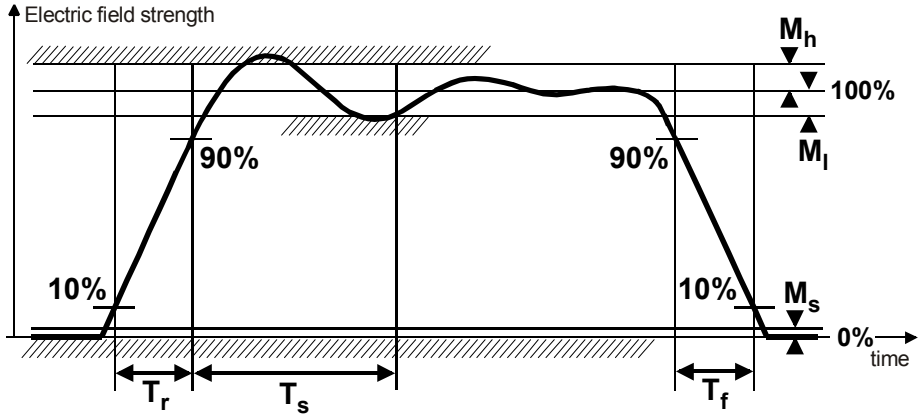


Figure Amd.1-6 — Interrogator power-up and power-down RF envelope

Table Amd.1-8 — Interrogator power-up waveform parameters

Parameter	Definition	Minimum	Nominal	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 Amd.1-9 — Interrogator power-down waveform parameters

Parameter	Definition	Minimum	Nominal	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

9.3.1.2.8 R=>T preamble and frame-sync

An interrogator shall begin all R=>T signalling with either a preamble or a frame-sync, both of which are shown in Figure Amd.1-7. A preamble shall precede a *Query* command (see 9.3.2.10.2.1) and denotes the start of an inventory round. All other signalling 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 Amd.1-7. The RF envelope shall be as specified in Figure Amd.1-5.

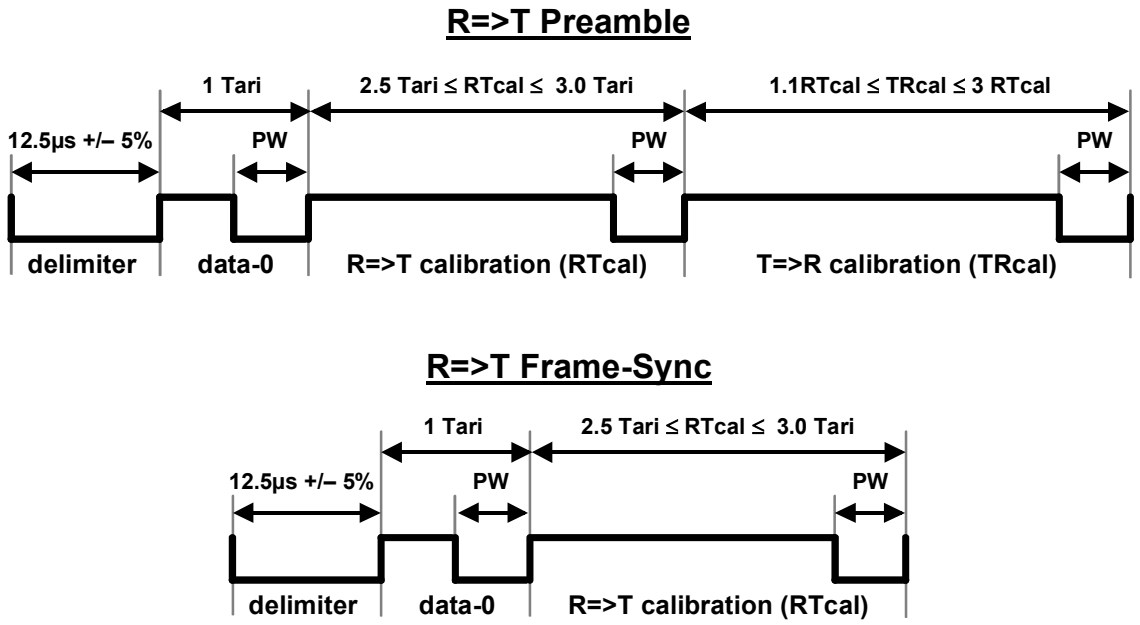


Figure Amd.1-7 — R=>T preamble and frame-sync

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 (BLF), TRcal, and DR. A tag shall measure the length of TRcal, compute BLF, and adjust its T=>R link rate to be equal to BLF (Table Amd.1-11

shows BLF values and tolerances). The TRcal and RTcal that an interrogator uses in any inventory round shall meet the constraints in Equation (2):

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

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

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.

9.3.1.2.9 Frequency-hopping spread-spectrum waveform

When an interrogator uses frequency-hopping spread spectrum (FHSS) signalling, the interrogator's RF envelope shall comply with Figure Amd.1-8 and Table Amd.1-10. The RF envelope shall not fall below the 90% point in Figure Amd.1-8 during interval T_{hs}. Interrogators shall not issue commands before the end of the maximum settling-time interval in Table Amd.1-10 (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.

9.3.1.2.10 Frequency-hopping spread-spectrum channelisation

Interrogators that are claimed to operate in single-interrogator environments shall meet local regulations for spread-spectrum channelisation. Interrogators that are claimed to operate in multiple- or dense-interrogator environments shall meet local regulations for spread-spectrum channelisation, unless the channelisation is unregulated, in which case interrogators shall adopt the channelisation described by the algorithm in Figure I.1 (Annex I describes multiple- and dense-interrogator channelised signalling).

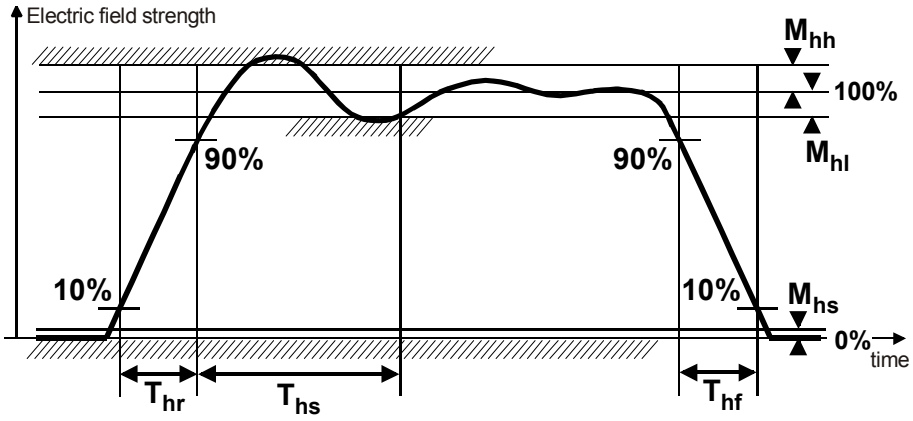


Figure Amd.1-8 — FHSS interrogator RF envelope

Table Amd.1-10 — FHSS waveform parameters

Parameter	Definition	Minimum	Nominal	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

9.3.1.2.11 Transmit mask

Interrogators that are claimed to operate according to this International Standard shall meet local regulations for out-of-channel and out-of-band spurious radio-frequency emissions.

Interrogators that are claimed to operate in multiple-interrogator environments, in addition to meeting local regulations, shall also meet the Multiple-Interrogator Transmit Mask specified in this International Standard:

Multiple-Interrogator Transmit Mask: For an interrogator transmitting random data 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 Amd.1-9, 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 an exception.

Interrogators that are claimed to operate in dense-interrogator environments shall meet both local regulations and the Transmit Mask shown in Figure Amd.1-9 of this International Standard, except when operating in the optional dense-interrogator mode described in Annex I, in which case they shall instead meet the Dense-interrogator Transmit Mask described below and shown in Figure Amd.1-10. Regardless of the mask used, interrogators certified for operation in dense-interrogator environments shall not be permitted the two exceptions to the transmit mask that are allowed for interrogators certified for operation in multiple-interrogator environments.

Dense-Interrogator Transmit Mask: For interrogator transmissions centred at a frequency f_c, a 2.5/Tari bandwidth R_{BW} also centred at f_c, an offset frequency f_o = 2.5/Tari, and a 2.5/Tari bandwidth S_{BW} centred at $(n \times f_o) + f_c$ (integer n), the ratio of the integrated power $P()$ in S_{BW} to that in R_{BW} with the Interrogator transmitting random data 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 Amd.1-10, with dBch defined as dB referenced to the integrated power in the reference channel.

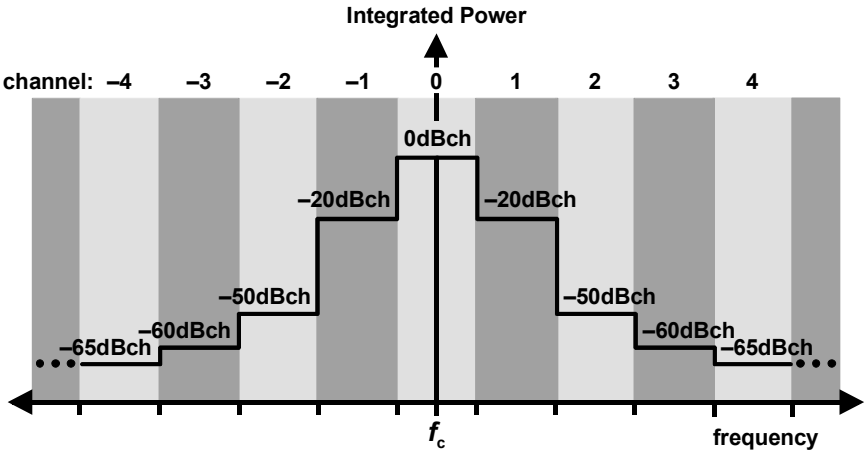


Figure Amd.1-9 — Transmit mask for multiple-interrogator environments

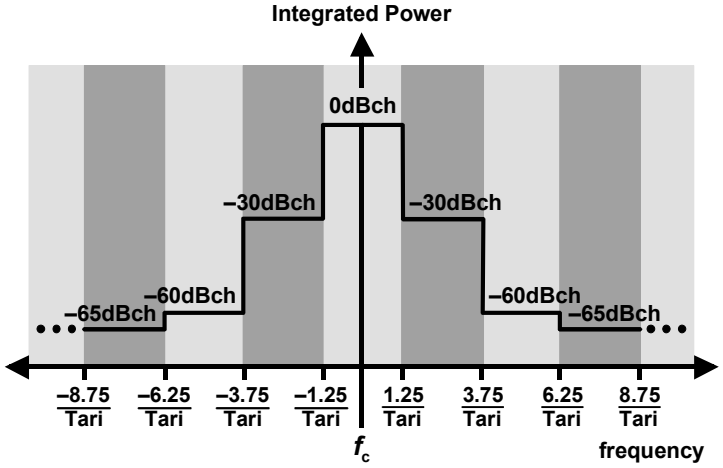


Figure Amd.1-10 — Transmit mask for dense-interrogator environments

9.3.1.3 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 9.3.2.8. 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 Amd.1-12, Figure Amd.1-13, Figure Amd.1-14, Figure Amd.1-16, Figure Amd.1-17 and Figure Amd.1-18 correspond to the antenna-reflectivity state the tag exhibits during the CW period prior to a T=>R preamble (e.g. an ASK 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 (e.g. an ASK tag reflecting power).

9.3.1.3.1 Modulation

Tag backscatter shall use ASK and/or PSK modulation. The tag vendor selects the modulation format. Interrogators shall demodulate both modulation types.

9.3.1.3.2 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.

9.3.1.3.2.1 FM0 baseband

Figure Amd.1-11 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 Amd.1-11 maps a logical data sequence to the FM0 basis functions that are transmitted. The state 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.

Figure Amd.1-12 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 Amd.1-12 depends on prior transmissions. FM0 signalling shall always end with a "dummy" data-1 bit at the end of a transmission, as shown in Figure Amd.1-13.

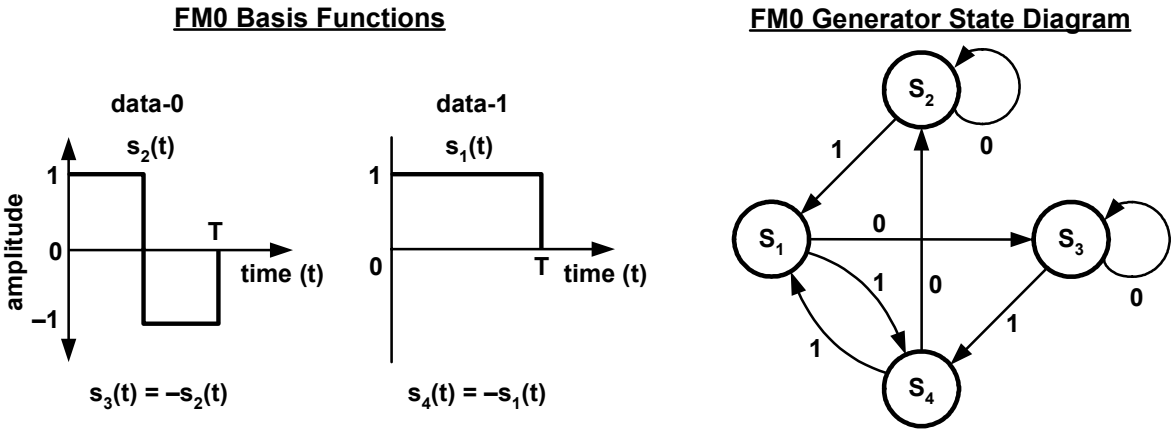


Figure Amd.1-11 — FM0 basis functions and generator state diagram

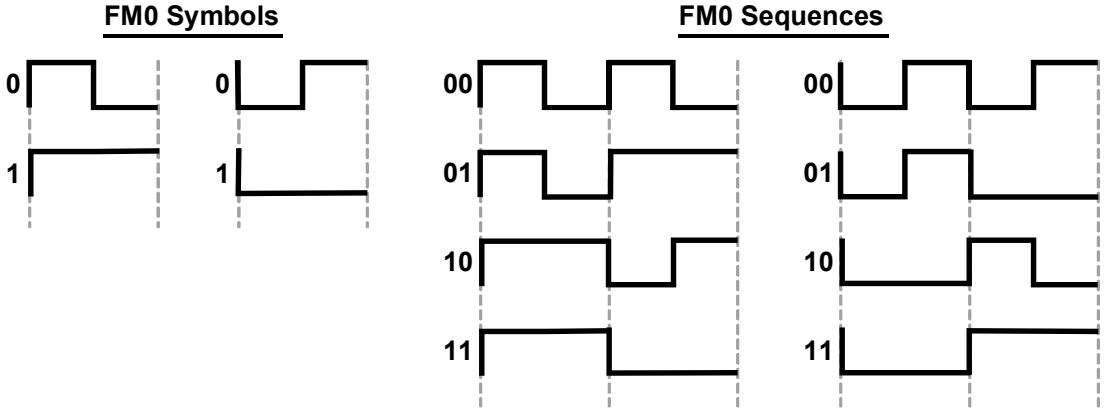


Figure Amd.1-12 — FM0 symbols and sequences

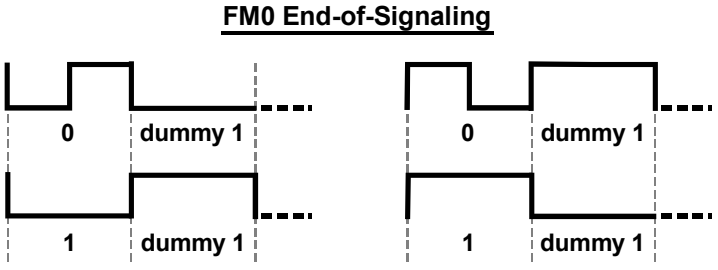


Figure Amd.1-13 — Terminating FM0 transmissions

9.3.1.3.2.2 FM0 preamble

T=>R FM0 signalling shall begin with one of the two preambles shown in Figure Amd.1-14. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round,

unless a tag is replying to a command that writes to memory, in which case a tag shall use the extended preamble regardless of TRext (i.e. the tag replies as if TRext=1 regardless of the TRext value specified in the *Query*—see 9.3.2.10.3). The "v" shown in Figure Amd.1-14 indicates an FM0 violation (i.e. a phase inversion should have occurred but did not).

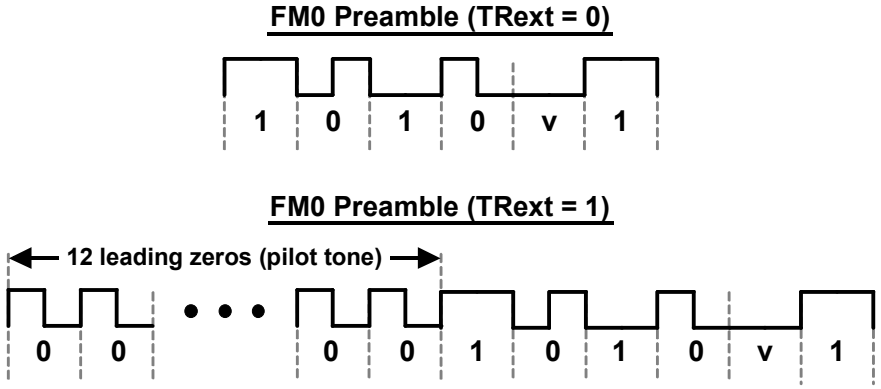


Figure Amd.1-14 — FM0 T=>R preamble

9.3.1.3.2.3 Miller-modulated subcarrier

Figure Amd.1-15 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 Amd.1-15 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.

Figure Amd.1-16 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 inventory round (see Figure Amd.1-15). 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 Amd.1-16 depends on prior transmissions. Miller signalling shall always end with a "dummy" data-1 bit at the end of a transmission, as shown in Figure Amd.1-17.

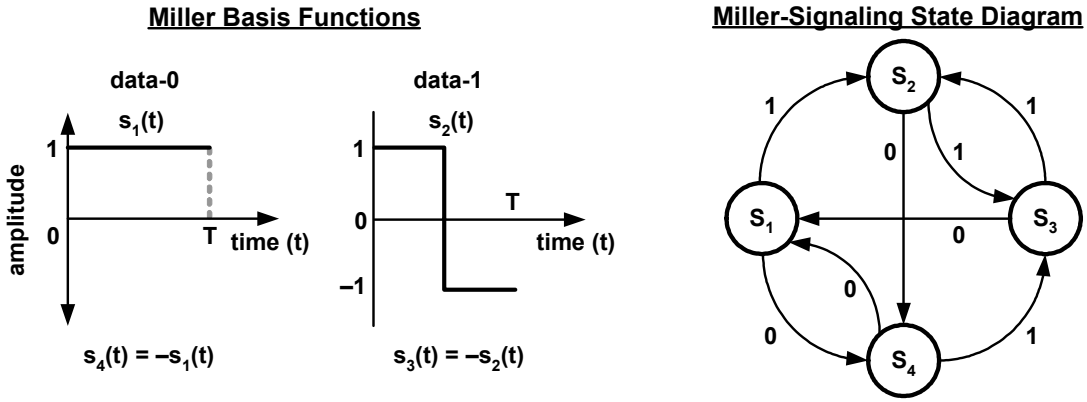


Figure Amd.1-15 — Miller basis functions and generator state diagram

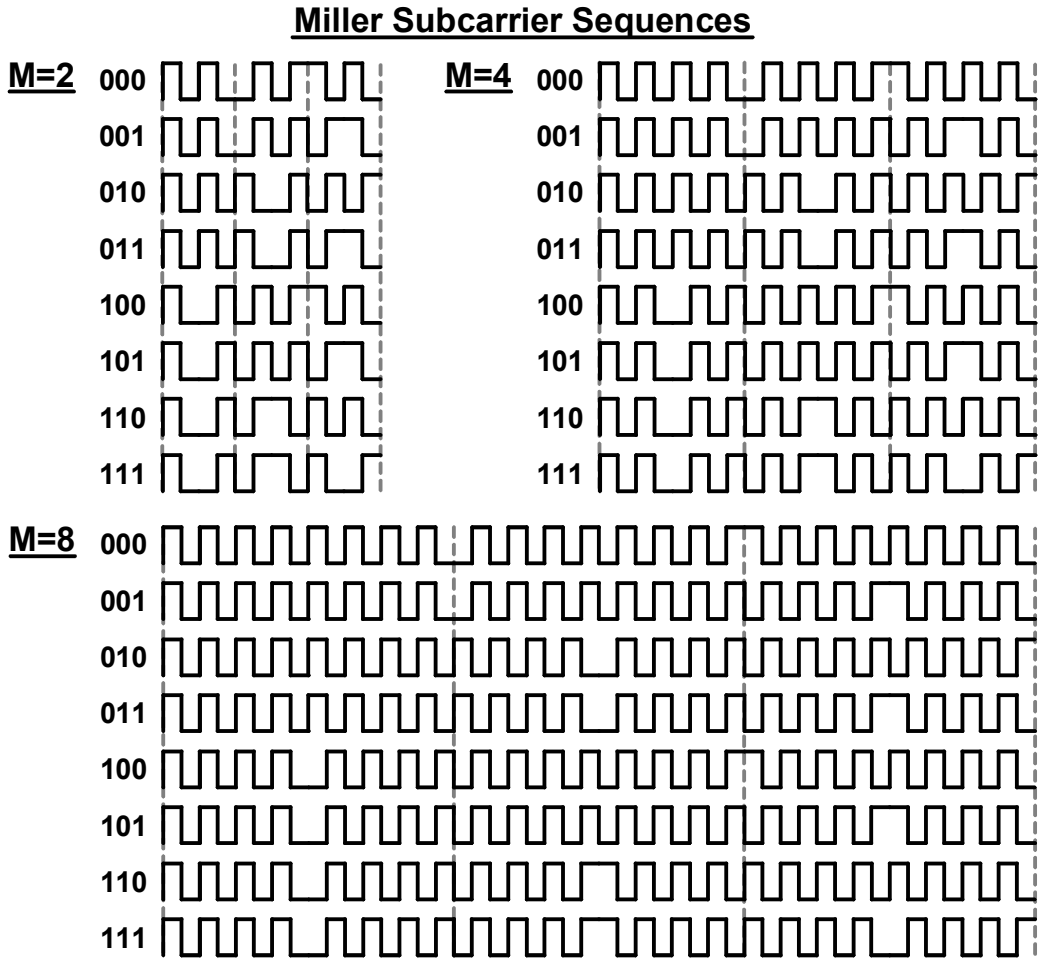


Figure Amd.1-16 — Subcarrier sequences

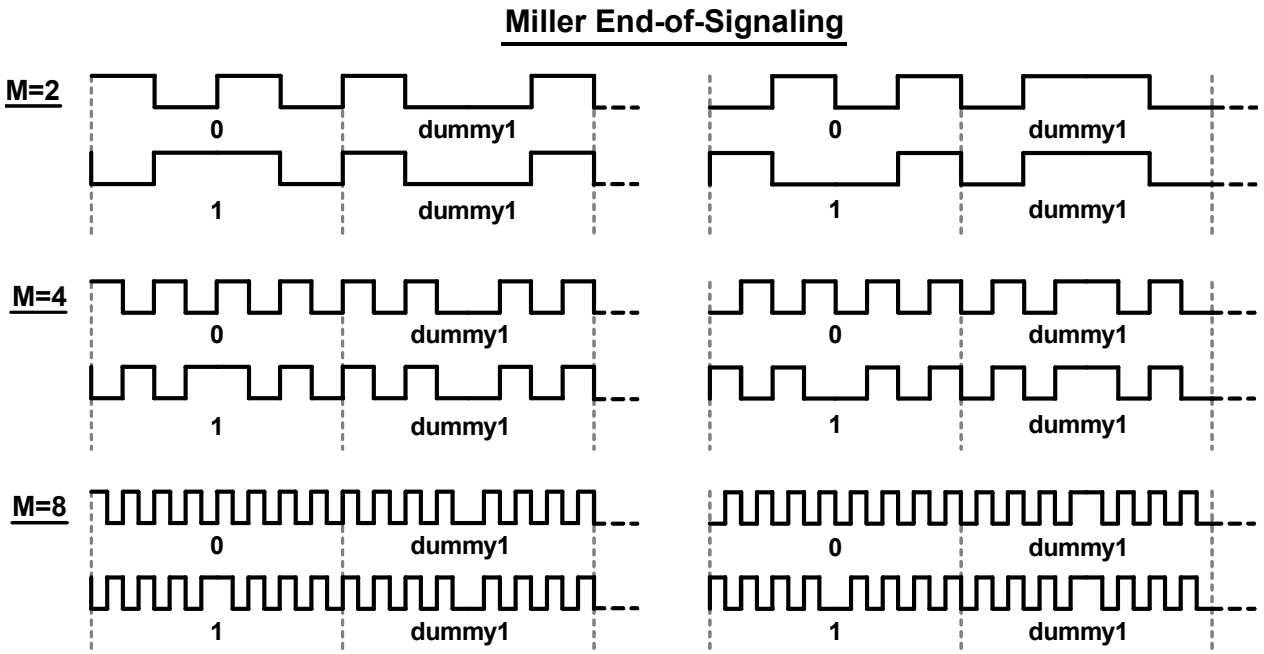


Figure Amd.1-17 — Terminating subcarrier transmissions

9.3.1.3.2.4 Miller subcarrier preamble

T=>R subcarrier signalling shall begin with one of the two preambles shown in Figure Amd.1-18. The choice depends on the value of the TRext bit specified in the *Query* command that initiated the inventory round, unless a tag is replying to a command that writes to memory, in which case a tag shall use the extended preamble regardless of TRext (i.e. the tag replies as if TRext=1 regardless of the TRext value specified in the *Query*—see 9.3.2.10.3).

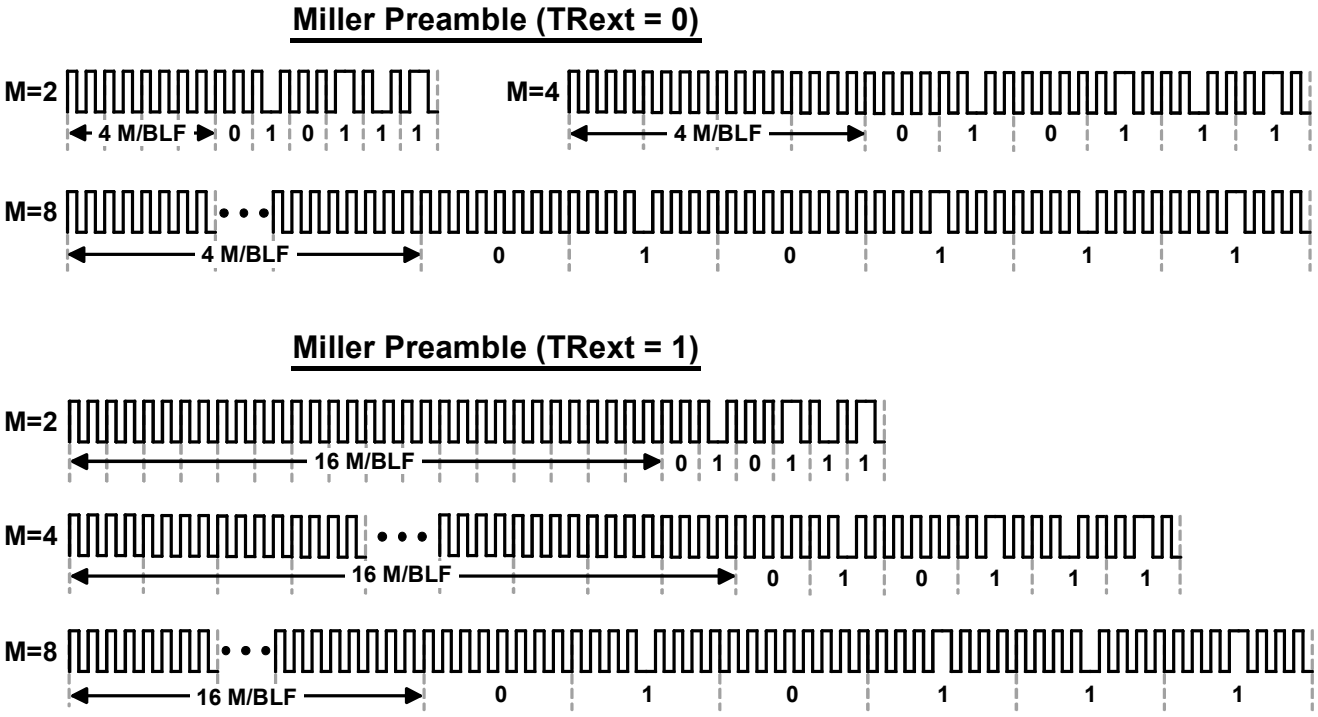


Figure Amd.1-18 — Subcarrier T=>R preamble

9.3.1.3.3 Tag supported Tari values and backscatter link rates

Tags shall support all R=>T Tari values in the range of 6.25μs to 25μs, over all parameters allowed by 9.3.1.2.3.

Tags shall support the T=>R link frequencies and tolerances specified in Table Amd.1-11, and the T=>R data rates specified in Table Amd.1-12. The frequency-variation requirement in Table Amd.1-11 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 Amd.1-11 and M in Table Amd.1-12; the preamble that precedes the *Query* specifies TRcal. BLF 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 9.3.1.2.8).

Table Amd.1-11 — Tag-to-interrogator link frequencies

DR: Divide Ratio	Trcal[a] (µs +/– 1%)	BLF: Link Frequency (kHz)	Frequency Tolerance FT (-25 °C to 40 °C)	Frequency Tolerance FT (-40 °C to 65 °C)	Frequency variation during backscatter
64/3	33.3	640	+ / – 15%	+ / – 15%	+ / – 2.5%
	33.3 < TRcal < 66.7	320 < BLF < 640	+ / – 22%	+ / – 22%	+ / – 2.5%
	66.7	320	+ / – 10%	+ / – 15%	+ / – 2.5%
	66.7 < TRcal < 83.3	256 < BLF < 320	+ / – 12%	+ / – 15%	+ / – 2.5%
	83.3	256	+ / – 10%	+ / – 10%	+ / – 2.5%
	83.3 < TRcal ≤ 133.3	160 ≤ BLF < 256	+ / – 10%	+ / – 12%	+ / – 2.5%
	133.3 < TRcal ≤ 200	107 ≤ BLF < 160	+ / – 7%	+ / – 7%	+ / – 2.5%
	200 < TRcal ≤ 225	95 ≤ BLF < 107	+ / – 5%	+ / – 5%	+ / – 2.5%
8	17.2 ≤ TRcal < 25	320 < BLF ≤ 465	+ / – 19%	+ / – 19%	+ / – 2.5%
	25	320	+ / – 10%	+ / – 15%	+ / – 2.5%
	25 < TRcal < 31.25	256 < BLF < 320	+ / – 12%	+ / – 15%	+ / – 2.5%
	31.25	256	+ / – 10%	+ / – 10%	+ / – 2.5%
	31.25 < TRcal < 50	160 < BLF < 256	+ / – 10%	+ / – 10%	+ / – 2.5%
	50	160	+ / – 7%	+ / – 7%	+ / – 2.5%
	50 < TRcal ≤ 75	107 ≤ BLF < 160	+ / – 7%	+ / – 7%	+ / – 2.5%
	75 < TRcal ≤ 200	40 ≤ BLF < 107	+ / – 4%	+ / – 4%	+ / – 2.5%

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

Table Amd.1-12 — Tag-to-interrogator data rates

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

9.3.1.3.4 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 Amd.1-8 or Table Amd.1-10, as appropriate (i.e. within T_s or T_{hs}, respectively).

9.3.1.3.5 Minimum operating field strength and backscatter strength

The tag manufacturer shall specify:

— the free-space sensitivity,

— the minimum relative backscattered modulated power (ASK modulation) or change in radar cross-section or equivalent (phase modulation), and

— the manufacturer's normal operating conditions,

for the tag mounted on one or more manufacturer-selected materials.

9.3.1.4 Transmission order

The transmission order for all R=>T and T=>R communications shall be most-significant bit (MSB) first.

Within each message, the most-significant word shall be transmitted first

Within each word, the MSB shall be transmitted first.

9.3.1.5 Cyclic Redundancy Check (CRC)

A CRC is a cyclic-redundancy check that a tag uses to ensure the validity of certain R=>T commands, and an interrogator uses to ensure the validity of certain backscattered T=>R replies. This International Standard uses two CRC types: (i) a CRC-16 and (ii) a CRC-5.

To generate a CRC-16 a tag or interrogator shall first generate the CRC-16 precursor shown in Table Amd.1-13, and then take the ones-complement of the generated precursor to form the CRC-16.

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

The CRC-16 that protects a tag's UII is computed and stored by a tag at power up - See Clause 9.3.2.10.

Tag shall append a CRC-16 to those replies that use a CRC-16. – See Clause 9.3.2.10 for command specific reply formats.

To generate a CRC-5 an interrogator shall use the definition in Table Amd.1-14.

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

Interrogator shall append the appropriate CRC to R=>T transmissions as specified in Table Amd.1-18.

Table Amd.1-13 — CRC-16 precursor

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

Table Amd.1-14 — CRC-5 definition

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

9.3.1.6 Link timing

Figure Amd.1-19 illustrates R=>T and T=>R link timing. The figure (not drawn to scale) defines interrogator interactions with a tag population. Table Amd.1-15 shows the timing requirements for Figure Amd.1-19, while 9.3.2.10 describes the commands. Tags and interrogators shall meet all timing requirements shown in

Table Amd.1-15. RTcal is defined in 9.3.1.2.8; T_{pri} is the T=>R link period (T_{pri} = 1 / BLF). As described in 9.3.1.2.8, 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.

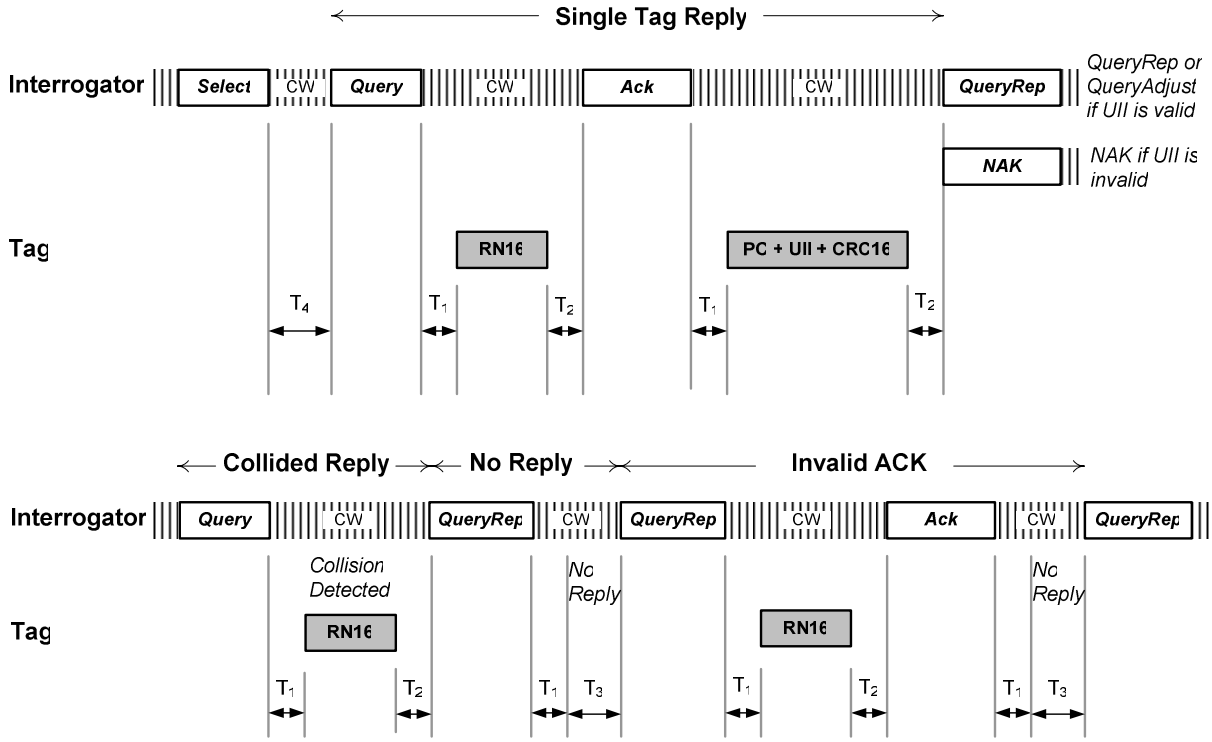


Figure Amd.1-19 — Link timing

Table Amd.1-15 — Link timing parameters

Parameter	Minimum	Nominal	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}	Interrogator response 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

The following items apply to the requirements specified in Table Amd.1-15:

1) T_{pri} denotes either the commanded period of an FM0 symbol or the commanded period of a single subcarrier cycle, as appropriate.

2) A tag may exceed the maximum value for T_1 when responding to commands that write to memory — see, for example, 9.3.2.10.3.3.

3) The maximum value for T_2 shall apply only to tags in the **reply** or **acknowledged** states (see 9.3.2.4.3 and 9.3.2.4.4). 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 9.3.2.4.2),

   — 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 9.3.2.10.

4) 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.

5) FT is the frequency tolerance specified in Table Amd.1-11.

6) T_1+T_3 shall not be less than T_4.

9.3.2 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.

9.3.2.1 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 Amd.1-20. The memory banks are:

— **Reserved memory** shall contain the kill and/or access passwords, if passwords are implemented on the tag. 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 $3F_h$. See 9.3.2.1.1.

— **UII 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 UII, and hereafter referred to as an UII) that identifies the object to which the tag is or will be attached beginning at address 20_h. See 9.3.2.1.2.

— **TID memory** shall contain an 8-bit ISO/IEC 15963 allocation class identifier at memory locations 00_h to 07_h. TID memory shall contain sufficient identifying information above 07_h for an interrogator to uniquely identify the custom commands and/or optional features that a tag supports. See 9.3.2.1.3.

— **User memory** allows user-specific data storage. See 9.3.2.1.4.

The logical addressing of all memory banks shall begin at zero (00_h). The physical memory map is vendor-specific. Commands that access memory have a MemBank parameter that selects the bank, and an address parameter, specified using the EBV format described in Annex D, to select a particular memory location within that bank. When tags backscatter memory contents, this backscatter shall fall on word boundaries (except in the case of a truncated reply – see 9.3.2.10.1.1).

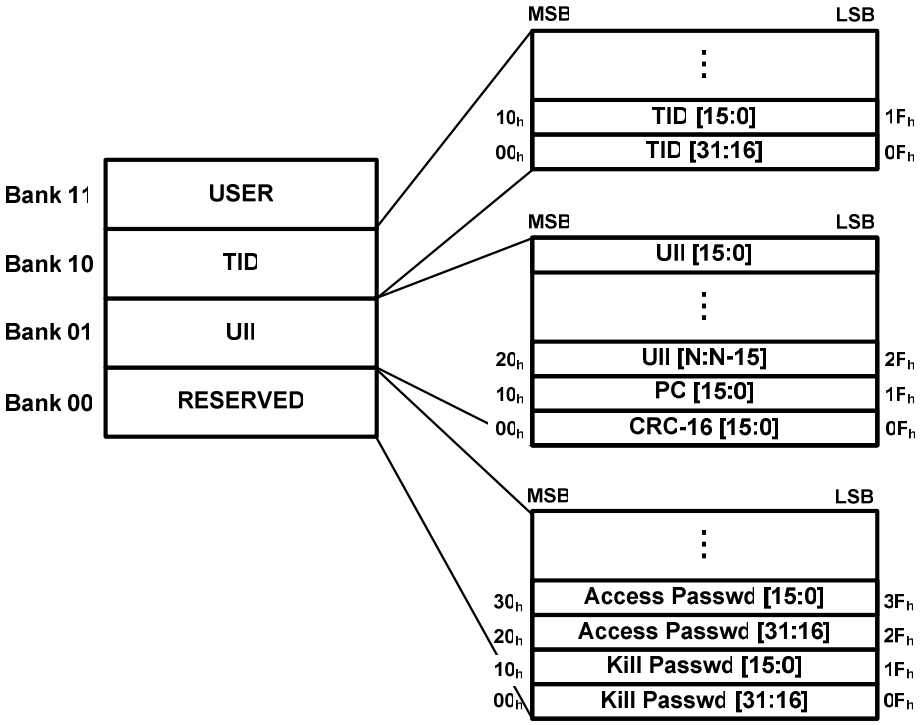


Figure Amd.1-20 — Logical memory map

MemBank is defined as follows:

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

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

Memory writes, detailed in 9.3.2.9, 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 9.3.2.9 and 9.3.2.10.3.5 for a detailed description of memory locking and unlocking. The kill and access passwords are individually lockable, as are UII, TID, and User memory. If the kill and/or access passwords are locked they are usable by only the *Kill* and *Access* commands, respectively, and are rendered both unwriteable and unreadable by any other command. The UII, TID, and User memory banks are always readable regardless of their lock status.

9.3.2.1.1 Reserved Memory

Reserved memory contains the kill (see 9.3.2.1.1.1) and/or access (see 9.3.2.1.1.2) passwords, if passwords are implemented on the tag. If a tag does not implement the kill and/or access password(s), the tag shall logically operate as though it has zero-valued password(s) that are permanently read/write locked (see 9.3.2.10.3.5), and the corresponding physical memory locations in Reserved memory need not exist.

9.3.2.1.1.1 Kill password

The kill password is a 32-bit value stored in Reserved memory 00_h to $1F_h$, MSB first. The default (unprogrammed) value shall be zero. An interrogator shall use a kill password once, to kill the tag and render it nonresponsive thereafter. A tag shall not execute a kill operation if its kill password is zero. A tag that does

not implement a kill password operates as if it has a zero-valued kill password that is permanently read/write locked.

9.3.2.1.1.2 Access password

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

9.3.2.1.2 UII Memory

UII memory contains a CRC-16 at memory addresses 00_h to $0F_h$, PC bits at memory locations 10_h to $1F_h$, and a UII beginning at address 20_h. The CRC-16, PC, and UII shall be stored MSB first (the UII's MSB is stored in location 20_h).

The CRC-16 is described in 9.3.2.1.2.1

The PC, as described in 9.3.2.1.2.2, is subdivided into a UII 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 UII is a code that identifies the object to which a tag is affixed. The UII for EPCglobal™ Applications is described in 9.3.2.1.2.3; the UII for non-EPCglobal™ Applications is described in 9.3.2.1.2.4. Interrogators may issue a *Select* command that includes all or part of the UII in the mask. Interrogators may issue an *ACK* command to cause a tag to backscatter its PC, UII, and CRC-16 (under certain circumstances the tag may truncate its reply — see 9.3.2.10.1.1). An Interrogator may issue a *Read* command to read all or part of the UII.

9.3.2.1.2.1 CRC-16

The PC and UII are protected by the CRC-16 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 UII memory. At power-up a tag shall compute this CRC-16 over UII memory location 10_h to the end of the UII (not necessarily to the end of UII memory, but to the end of the UII specified by the length field in the PC — see 9.3.2.1.2.2) and map the computed CRC-16 into UII memory 00_h to $0F_h$, MSB first. Because the {PC+ UII} is stored in UII 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 Amd.1-6 or Figure Amd.1-8, respectively. tags shall not recalculate this CRC-16 for a truncated reply (see 9.3.2.10.1.1).

9.3.2.1.2.2 Protocol-control (PC) bits

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

Bits 10_h – 14_h: The length of the (PC + UII) that a tag backscatters, in words:
- 00000_2: One word (addresses 10_h to $1F_h$ in UII memory).
- 00001_2: Two words (addresses 10_h to $2F_h$ in UII memory).
- 00010_2: Three words (addresses 10_h to $3F_h$ in UII memory).
- •
- •
- •
- 11111_2: 32 words (addresses 10_h to $20F_h$ in UII memory).

Bits 15_h – 16_h: RFU (shall be set to 00_2).

Bits 17_h – $1F_h$: A numbering system identifier (NSI). The MSB of the NSI is stored in memory location 17_h. If bit 17_h contains a logical 0, then the Application is referred to as an EPCglobal™ Application and PC bits 18_h – $1F_h$ shall be as defined in the EPCglobal™ Tag Data Standards. If bit 17_h contains a logical 1, then the Application is referred to as a non-EPCglobal™ Application and PC bits 18_h – $1F_h$

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

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

During truncated replies a tag substitutes 00000_2 for the PC bits — see 9.3.2.10.1.1.

If an interrogator modifies the UII length during a memory write, and it wishes the tag to subsequently backscatter the modified UII, then it must write the length of the new or updated (PC + UII) into the first 5 bits of the tag's PC. A tag shall backscatter an error code (see Annex K) if an interrogator attempts to write a (PC + UII) 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 + UII) words designated by the first 5 bits of the PC rather than over the length of the entire UII memory (see 9.3.2.1.2.1).

9.3.2.1.2.3 UII for an EPCglobal™ Application

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

9.3.2.1.2.4 UII for a non-EPCglobal™ Application

The UII structure for a non-EPCglobal™ Application shall be as defined in ISO/IEC 15961.

9.3.2.1.3 TID memory

TID memory locations 00_h to 07_h shall contain one of two ISO/IEC 15963 class-identifier values — either $E0_h$ or $E2_h$. TID memory locations above 07_h shall be defined according to the registration authority defined by this class-identifier value and shall contain, at a minimum, sufficient identifying information for an Interrogator to uniquely identify the custom commands and/or optional features that a tag supports. TID memory may also contain tag- and vendor-specific data (for example, a tag serial number).

NOTE The tag manufacturer assigns the class-identifier value (i.e. $E0_h$ or $E2_h$), for which ISO/IEC 15963 defines the registration authorities. The class-identifier does not specify the Application. If the class identifier is $E0_h$, TID memory locations 08_h to $0F_h$ contain an 8-bit manufacturer identifier, TID memory locations 10_h to $3F_h$ contain a 48-bit tag serial number (assigned by the tag manufacturer), the composite 64-bit tag ID (i.e. TID memory 00_h to $3F_h$) is unique among all classes of tags defined in ISO/IEC 15693, and TID memory is permalocked at the time of manufacture. If the class identifier is $E2_h$, TID memory locations 08_h to 13_h contain a 12-bit tag mask-designer identifier (obtainable from the registration authority), TID memory locations 14_h to $1F_h$ contain a vendor-defined 12-bit tag model number, and the usage of tag memory locations above $1F_h$ is defined in version 1.3 and above of the EPCglobal™ Tag Data Standards.

9.3.2.1.4 User memory

A tag may contain User memory. User memory allows user-specific data storage.

9.3.2.1.4.1 User memory for an EPCglobal™ Application

If User memory is included on a tag then its encoding shall be as defined in the EPCglobal™ Tag Data Standards.

9.3.2.1.4.2 User memory for a non-EPCglobal™ Application

If User memory is included on a tag then User memory locations 00_h to 07_h shall be the DSFID as defined in ISO/IEC 15961. The encoding of User memory locations above 07_h shall be as defined in ISO/IEC 15962.

9.3.2.2 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 Amd.1-21.

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.

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.

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.

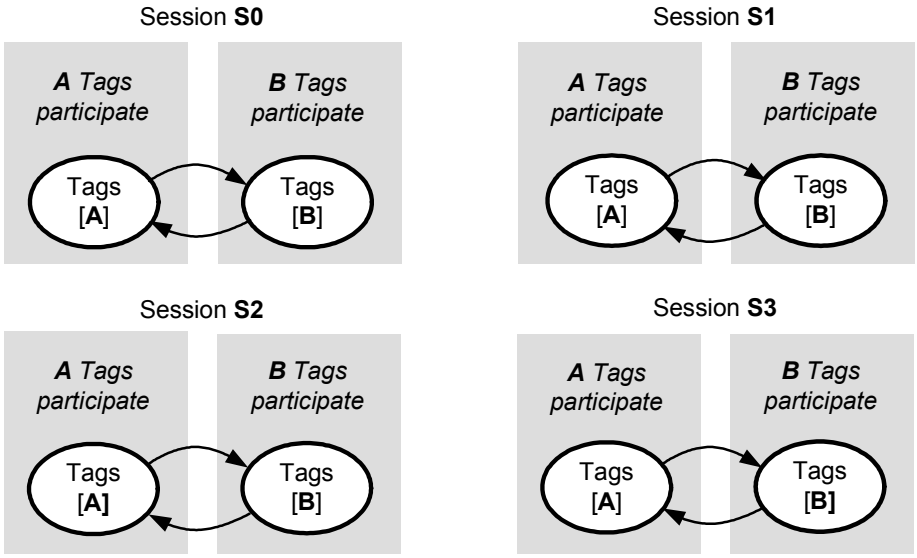


Figure Amd.1-21 — Session diagram

A tag's **inventoried** flags shall have the persistence times shown in Table Amd.1-16. 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 set 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 Amd.1-16. The value of the S1 **inventoried** flag shall not change as a result of a persistence timeout while a tag is participating in an inventory round. If the S1 persistence time expires during an inventory round then the tag shall change its S1 flag to *A* at the end of the round.

9.3.2.3 Selected flag

The tag shall implement a **selected** flag, **SL**, which an interrogator may assert or de-assert using a *Select* command. The Sel parameter in the *Query* command allows an interrogator to inventory tags that have **SL** either asserted or de-asserted (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** may be used in any session and is common to all sessions.

A tag's **SL** flag shall have the persistence times shown in Table Amd.1-16. A tag shall power-up with its **SL** flag either asserted or de-asserted, 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 de-asserted (set to ~**SL**). A tag shall be capable of asserting or de-asserting 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 Amd.1-16.

Table Amd.1-16 — Tag flags and persistence values

Flag	Required persistence
S0 **inventoried** flag	Tag energized: Indefinite Tag not energized: None
S1 **inventoried** flag[a]	Tag energized: –25 °C to +40 °C: 500ms < persistence < 5s –40 °C to +65 °C: Not specified Tag not energized: –25 °C to +40 °C: 500ms < persistence < 5s –40 °C to +65 °C: Not specified
S2 **inventoried** flag[a]	Tag energized: Indefinite Tag not energized: –25 °C to +40 °C: 2s < persistence –40 °C to +65 °C: Not specified
S3 **inventoried** flag[a]	Tag energized: Indefinite Tag not energized: –25 °C to +40 °C: 2s < persistence –40 °C to +65 °C: Not specified
Selected (**SL**) flag[a]	Tag energized: Indefinite Tag not energized: –25 °C to +40 °C: 2s < persistence –40 °C to +65 °C: Not specified

The following requirement applies to those items denoted with a superscript "a" in Table Amd.1-16: For a randomly chosen and sufficiently large tag population, 95% of the tag persistence times shall meet the persistence requirements in Table Amd.1-16, with a 90% confidence interval.

9.3.2.4 Tag states and slot counter

Tags shall implement the states and the slot counter shown in Figure Amd.1-22. Annex E shows the associated state-transition tables; Annex F shows the associated command-response tables.

9.3.2.4.1 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 9.3.2.10.2.1) 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 9.3.2.5), 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.

9.3.2.4.2 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 9.3.2.4.8) hold nonzero values. A tag in **arbitrate** shall decrement its slot counter every time it receives a *QueryRep* command (see 9.3.2.10.2.3) whose session parameter matches the session for the inventory round currently in progress, and it shall transition to the **reply** state and backscatter an RN16 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**.

9.3.2.4.3 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, UII and CRC-16. If the tag fails to receive an *ACK* within time $T_{2(max)}$, or receives an invalid *ACK* or an *ACK* with an erroneous RN16, it shall return to **arbitrate**. Tag and interrogator shall meet all timing requirements specified in Table Amd.1-15.

9.3.2.4.4 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 Amd.1-22). If a tag in the **acknowledged** state receives a valid *ACK* containing the correct RN16 it shall re-backscatter its PC, UII, and CRC-16. If a tag in the **acknowledged** state fails to receive a valid command within time $T_{2(max)}$ it shall return to **arbitrate**. Tag and interrogator shall meet all timing requirements specified in Table Amd.1-15.

9.3.2.4.5 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 Amd.1-22). If a tag in the **open** state receives a valid *ACK* containing the correct handle it shall re-backscatter its PC, UII, and CRC-16. Tag and interrogator shall meet all timing requirements specified in Table Amd.1-15 except $T_{2(max)}$; in the **open** state the maximum delay between tag response and interrogator transmission is unrestricted.

9.3.2.4.6 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 sequence, 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 Amd.1-22). If a tag in the **secured** state receives a valid *ACK* containing the correct handle it shall re-backscatter its PC, UII, and CRC-16. Tag and interrogator shall meet all timing requirements specified in Table Amd.1-15 except $T_{2(max)}$; in the **secured** state the maximum delay between tag response and interrogator transmission is unrestricted.

9.3.2.4.7 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 sequence (see 9.3.2.10.3.4) 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.

9.3.2.4.8 Slot counter

Tags shall implement a 15-bit slot counter. Upon receiving a *Query* or *QueryAdjust* command a tag shall preload into its slot counter a value between 0 and 2^Q–1, drawn from the tag's RNG (see 9.3.2.5). *Q* is an integer in the range (0,15). A *Query* specifies *Q*; a *QueryAdjust* may modify *Q* from the prior *Query*.

Tags in the **arbitrate** state shall 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 an 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*. The slot counter shall be capable of continuous counting, meaning that, after the slot counter rolls over to $7FFF_h$ it begins counting down again, thereby effectively preventing subsequent replies until the tag loads a new random value into its slot counter. See also Annex L.

9.3.2.5 Tag random or pseudo-random number generator

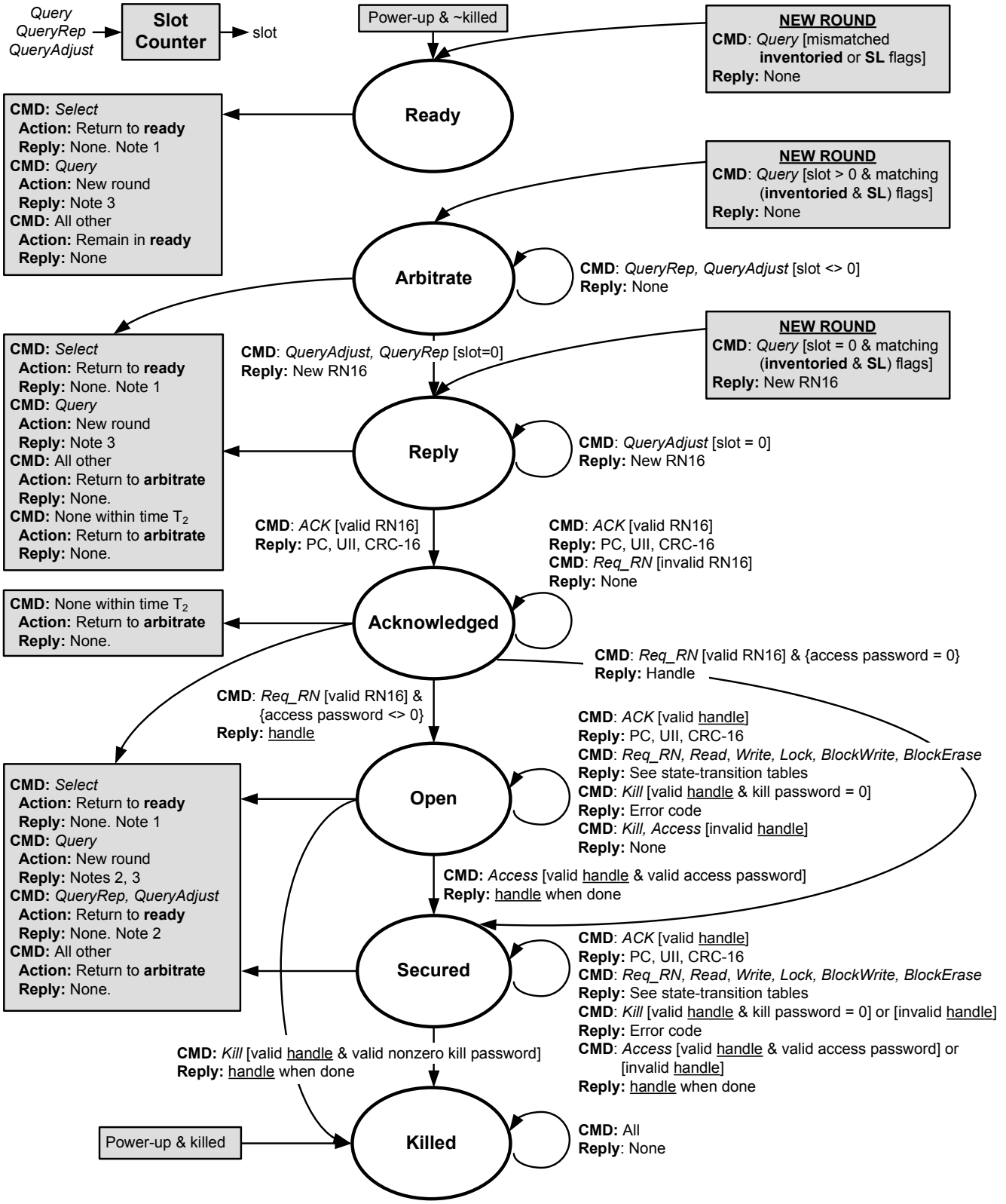
Tags shall implement a random or pseudo-random number generator (RNG). The RNG shall meet the following randomness criteria independent of the strength of the energizing field, the R=>T link rate, and the data stored in the tag (including the PC, UII, and CRC-16). Tags shall generate 16-bit random or 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 9.3.2.4.8). 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 Amd.1-26 or Figure Amd.1-28).

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 Amd.1-6 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 document recommends that interrogators wait 10ms after T_r in Figure Amd.1-6 or T_{hr} in Figure Amd.1-8 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 Amd.1-22 — Tag state diagram

9.3.2.6 Managing tag populations

Interrogators manage tag populations using the three basic operations shown in Figure Amd.1-23. Each of these operations comprises one or more commands. The operations are defined as follows:

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.

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, UII, and CRC-16 from the tag. An inventory round operates in one and only one session at a time. Annex H shows an example of an interrogator inventorying and accessing a single tag.

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.

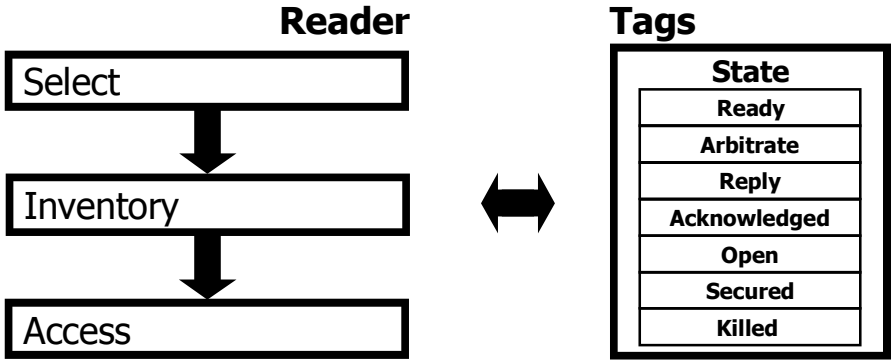


Figure Amd.1-23 — Interrogator/tag operations and tag state

9.3.2.7 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 de-assert a tag's **SL** flag, or it can set a tag's **inventoried** flag to either *A* or *B* in any one of the four sessions. *Select* contains the parameters Target, Action, MemBank, Pointer, Length, Mask, and Truncate.

Target and Action indicate whether and how a *Select* modifies a tag's **SL** or **inventoried** flag, and in the case of the **inventoried** flag, for which session. A *Select* that modifies **SL** shall not modify **inventoried**, and vice versa.

MemBank specifies if the mask applies to UII, TID, or User memory. *Select* commands apply to a single memory bank. Successive *Selects* may apply to different memory banks.

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 UII, or only that portion of the UII immediately following Mask. Truncated replies are always followed by the CRC-16 in UII 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.

9.3.2.8 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 (“inventory round” is defined in clause 4.1).

Query contains a slot-count parameter *Q*. Upon receiving a *Query* participating tags pick a random value in the range (0, 2^Q-1), inclusive, and load this value into their slot counter. Tags that pick a zero transition to the **reply** state and reply immediately. Tags that pick a nonzero value 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:

The tag backscatters an RN16 as it enters **reply**,

The interrogator acknowledges the tag with an *ACK* containing this same RN16,

The acknowledged tag transitions to the **acknowledged** state, backscattering its PC, UII, and CRC-16,

The interrogator issues a *QueryAdjust* or *QueryRep,* causing the identified tag to invert its **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$) 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 Amd.1-19), or receives the *ACK* with an erroneous RN16, it returns 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, UII, 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 re-backscatters its PC, UII, and CRC-16.

At any point the interrogator may issue a *NAK*, in response to which all tags in the inventory round that receive the *NAK* 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 transition to the **reply** state and reply immediately. Tags that pick a nonzero value 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) return to **arbitrate** with a slot value of 0000_h and 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.

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

The scenario outlined above assumed a single interrogator operating in a single session. However, as described in 9.3.2.2, an interrogator can inventory a tag population in one of four sessions. Furthermore, as

described in 9.3.2.10.2, 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** execute a *Query*, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure Amd.1-22).

If a tag in the **acknowledged**, **open**, or **secured** states receives a *Query* whose session parameter matches the prior session it inverts its **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$) 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 leaves 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** execute a *QueryAdjust* or *QueryRep* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the tag is not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively). Tags 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*, and the tag is not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively), it inverts its **inventoried** flag (i.e. $A \rightarrow B$ or $B \rightarrow A$) for the current session then transitions 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:

No tags reply: The interrogator may issue another *Query*, or it may issue a *QueryAdjust* or *QueryRep*.

One tag replies (see Figure Amd.1-24): 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, UII, 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* with matching session parameter 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 $A \rightarrow B$).

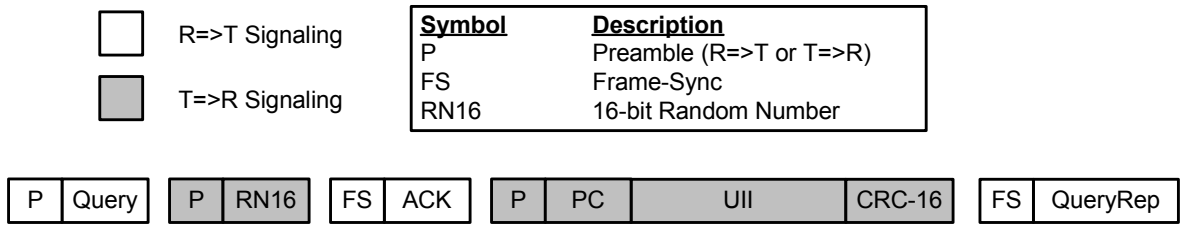


Figure Amd.1-24 — One tag reply

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 Amd.1-19), return to **arbitrate** and await the next *Query* or *QueryAdjust* command.

9.3.2.9 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*. The tag executes access commands in the states according to Table Amd.1-17.

Table Amd.1-17 — Access commands and tag states in which they are permitted

Command	Acknowledged State	Open State	Secured State	Remark
Req_RN	Permitted	Permitted	Permitted	
Read		Permitted	Permitted	
Write		Permitted	Permitted	Requires prior *Req_RN*
Kill		Permitted	Permitted	Requires prior *Req_RN*
Lock			Permitted	
Access		Permitted	Permitted	Optional command; Requires prior *Req_RN*
BlockWrite		Permitted	Permitted	Optional command
BlockErase		Permitted	Permitted	Optional command

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 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 entire duration of a tag access.

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, UII, 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 9.3.2.8. 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.

If an interrogator issues a command containing cover-coded data or a half-password and fails to receive a response from the tag, the interrogator may reissue the command unchanged. If the interrogator issues a command with new data or half-password, then it shall first issue a *Req_RN* to obtain a new RN16 and shall use this RN16 for the cover-coding.

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

The *BlockWrite* command (see 9.3.2.10.3.7) 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 K), indicating that the operation was unsuccessful. A tag reply to all access commands that write memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite*, and *BlockErase*) uses the extended preamble shown in Figure Amd.1-14 or Figure Amd.1-18, as appropriate (i.e. the tag replies as if TRext=1 regardless of the TRext value specified in the *Query* command that initiated the inventory round).

Killing a tag is a multi-step procedure, described in 9.3.2.10.3.4 and outlined in Figure Amd.1-26.

Issuing an access password to a tag is a multi-step procedure, described in 9.3.2.10.3.6 and outlined in Figure Amd.1-28.

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 Amd.1-41 for a detailed description of memory lock, permalock, and the tag state required to modify memory.

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.

9.3.2.10 Interrogator commands and tag replies

Interrogator-to-tag commands shall have the format shown in Table Amd.1-18.

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 Amd.1-18. 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 Amd.1-14 and detailed in Annex A.

Select, *Req_RN*, *Read*, *Write*, *Kill*, *Lock*, *Access*, *BlockWrite* and *BlockErase* are protected by a CRC-16, defined in 9.3.1.5 and detailed in Annex A.

R=>T commands begin with either a preamble or a frame-sync, as described in 9.3.1.2.8. The command-code lengths specified in Table Amd.1-18 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 E and Annex F.

Table Amd.1-18 — 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	–	–	–

9.3.2.10.1 Select commands

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

9.3.2.10.1.1 *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 de-assert 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 Amd.1-19. 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 Amd.1-20. 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 UII 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. Tags shall ignore *Select* commands whose Target is 101_2, 110_2, or 111_2.

Action indicates whether matching tags assert or de-assert **SL**, or set their **inventoried** flag to *A* or to *B*. Tags conforming to the contents of the MemBank, Pointer, Length, and Mask fields are considered matching. Tags not conforming to the contents of these fields are considered non-matching.

MemBank specifies whether Mask applies to UII, TID, or User memory. *Select* commands shall apply to a single memory bank. Successive *Selects* may apply to different banks. MemBank shall not specify 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 D). 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 or Truncate = 1 and Pointer is outside the UII specified in the PC bits, 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 UII immediately following Mask, followed by the CRC-16 stored in UII memory 00_h to $0F_h$. If an interrogator asserts Truncate, it shall assert it:

in the last *Select* that the interrogator issues prior to sending a *Query*,

only if the *Select* has Target = 100_2, and

only if Mask ends in the UII.

These constraints *do not* preclude an interrogator from issuing multiple *Select* commands that target the **SL** and/or **inventoried** flags. They *do* require that an interrogator that is requesting tags to truncate their replies assert Truncate in the last *Select*, and that this last *Select* targets the **SL** flag. Tags shall power-up with Truncate de-asserted.

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

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

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

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 9.3.1.2.8). The CRC-16 is calculated over the first command-code bit to the Truncate bit.

Tags shall not reply to a *Select*.

Table Amd.1-19 — *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 Amd.1-20	00: RFU 01: UII 10: TID 11: User	Starting Mask address	Mask length (bits)	Mask value	0: Disable truncation 1: Enable truncation	

Table Amd.1-20 — Tag response to Action parameter

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

9.3.2.10.2 Inventory commands

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

9.3.2.10.2.1 *Query* (mandatory)

Interrogators and tags shall implement the *Query* command shown in Table Amd.1-21. *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 9.3.1.2.8 and Table Amd.1-11.

M (cycles per symbol) sets the T=>R data rate and modulation format as shown in Table Amd.1-12.

TRext chooses whether the T=>R preamble is pre-pended with a pilot tone as described in 9.3.1.3.2.2 and 9.3.1.3.2.4, however, a tag's reply to access commands that write to memory (i.e. *Write*, *Kill*, *Lock*, *BlockWrite* and *BlockErase*) always use a pilot tone regardless of the value of TRext.

Sel chooses which tags respond to the *Query* (see 9.3.2.10.1.1 and 9.3.2.8).

Session chooses a session for the inventory round (see 9.3.2.8).

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 9.3.2.8).

Interrogators shall prepend a *Query* with a preamble (see 9.3.1.2.8).

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 shall pick a random value in the range (0, 2^Q-1), inclusive, and shall 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 Amd.1-22; 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 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 when beginning the new round.

Tags shall support all DR and M values specified in Table Amd.1-11 and Table Amd.1-12, respectively.

Tags in any state other than **killed** shall execute a *Query* command, starting a new round in the specified session and transitioning to **ready**, **arbitrate**, or **reply**, as appropriate (see Figure Amd.1-22). Tags in the **killed** state shall ignore a *Query*.

Table Amd.1-21 — *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 Amd.1-22 — Tag reply to a *Query* command

	Response
# of bits	16
description	RN16

9.3.2.10.2.2 *QueryAdjust* (mandatory)

Interrogators and tags shall implement the *QueryAdjust* command shown in Table Amd.1-23. *QueryAdjust* adjusts *Q* (i.e. the number of slots in an inventory round – see 9.3.2.8) without changing any other round parameters.

QueryAdjust includes the following fields:

Session corroborates the session number for the inventory round (see 9.3.2.8 and 9.3.2.10.2.1). 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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-24; otherwise, the tag shall remain silent. Tags shall respond to a *QueryAdjust* only if they received a prior *Query*.

Tags in any state except **ready** or **killed** shall execute a *QueryAdjust* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the tag is not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively).

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryAdjust* whose session parameter matches the session parameter in the prior *Query*, and who are not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively), shall invert their **inventoried** flag (i.e. *A*→*B* or *B*→*A*, as appropriate) for the current session and transition to **ready**.

Table Amd.1-23 — *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 Amd.1-24 — Tag reply to a *QueryAdjust* command

	Response
# of bits	16
description	RN16

9.3.2.10.2.3 *QueryRep* (mandatory)

Interrogators and tags shall implement the *QueryRep* command shown in Table Amd.1-25. *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 9.3.2.8 and 9.3.2.10.2.1). 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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-26; otherwise the tag shall remain silent. Tags shall respond to a *QueryRep* only if they received a prior *Query*.

Tags in any state except **ready** or **killed** shall execute a *QueryRep* command if, and only if, (i) the session parameter in the command matches the session parameter in the *Query* that started the round, and (ii) the tag is not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively).

Tags in the **acknowledged**, **open**, or **secured** states that receive a *QueryRep* whose session parameter matches the session parameter in the prior *Query*, and who are not in the middle of a *Kill* or *Access* command sequence (see 9.3.2.10.3.4 or 9.3.2.10.3.6 respectively), shall invert their **inventoried** flag (i.e. *A*→*B* or *B*→*A*, as appropriate) for the current session and transition to **ready**.

Table Amd.1-25 — *QueryRep* command

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

Table Amd.1-26 — Tag reply to a *QueryRep* command

	Response
# of bits	16
description	RN16

9.3.2.10.2.4 *ACK* (mandatory)

Interrogators and tags shall implement the *ACK* command shown in Table Amd.1-27. 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 9.3.2.10.3.1).

An *ACK* shall be pre-pended with a frame-sync (see 9.3.1.2.8).

The tag reply to a successful *ACK* shall be as shown in Table Amd.1-28. As described in 9.3.2.10.1.1, the reply may be truncated, in which case the tag reply is 00000_2 followed by the truncated UII followed by the CRC-16 computed by the tag at power-up. 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 Amd.1-27 — *ACK* command

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

Table Amd.1-28 — Tag reply to a successful *ACK* command

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

9.3.2.10.2.5 *NAK* (mandatory)

Interrogators and tags shall implement the *NAK* command shown in Table Amd.1-29. Any tag that receives a *NAK* shall return to the **arbitrate** state without changing its **inventoried** flag, unless the tag is in **ready** or **killed**, in which case it shall ignore the *NAK* and remain in its current state.

A *NAK* shall be pre-pended with a frame-sync (see 9.3.1.2.8).

Tags shall not reply to a *NAK*.

Table Amd.1-29 — *NAK* command

	Command
# of bits	8
description	11000000

9.3.2.10.3 Access commands

The set of access commands comprises *Req_RN, Read, Write*, *Kill*, *Lock*, *Access*, *BlockWrite*, and *BlockErase*. As described in 9.3.2.9, 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, the tag shall verify that the handle is correct prior to executing an access command, and the tag shall ignore access commands with an incorrect handle. The handle value is fixed for the entire duration of a tag access.

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 Amd.1-14 or Figure Amd.1-18, 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 UII, the CRC-16 stored in tag UII 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, because the tag computes the CRC-16 stored in tag UII memory 00_h to $0F_h$ at power-up

If an Interrogator attempts to write directly to UII memory 00_h to $0F_h$ using either a *Write* or a *BlockWrite* command, the tag shall respond with an error code (see Annex L for error-code definitions and for the reply format).

Req_RN, Read, Write, *Kill*, and *Lock* are required commands; *Access*, *BlockWrite*, and *BlockErase* are optional. A tag shall ignore optional access commands that it does not support.

See Annex M for an example of a data-flow exchange during which an interrogator accesses a tag and reads its kill password.

9.3.2.10.3.1 *Req_RN* (mandatory)

Interrogators and tags shall implement the *Req_RN* command shown in Table Amd.1-30. *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 pre-pended with a frame-sync (see 9.3.1.2.8).

The tag reply to a *Req_RN* shall be as shown in Table Amd.1-31. The RN16 or handle are protected by a CRC-16.

Table Amd.1-30 — *Req_RN* command

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

Table Amd.1-31 — Tag reply to a *Req_RN* command

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

9.3.2.10.3.2 *Read* (mandatory)

Interrogators and tags shall implement the *Read* command shown in Table Amd.1-32. *Read* allows an interrogator to read part or all of a tag's Reserved, UII, TID, or User memory. *Read* has the following fields:

MemBank specifies whether the *Read* accesses Reserved, UII, TID, or User memory. *Read* commands shall apply to a single memory bank. Successive *Reads* may apply to different banks.

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 D).

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 UII memory contents starting at WordPtr and ending at the length of the UII specified by the first 5 bits of the PC if WordPtr lies within the UII, and shall backscatter the UII memory contents starting at WordPtr and ending at the end of UII memory if WordPtr lies outside the UII.

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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-33. 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 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 Amd.1-15, rather than the reply shown in Table Amd.1-33 (see Annex K for error-code definitions and for the reply format).

Table Amd.1-32 — *Read* command

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

Table Amd.1-33 — Tag reply to a successful *Read* command

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

9.3.2.10.3.3 *Write* (mandatory)

Interrogators and tags shall implement the *Write* command shown in Table Amd.1-34. *Write* allows an interrogator to write a word in a tag's Reserved, UII, TID, or User memory. *Write* has the following fields:

MemBank specifies whether the *Write* occurs in Reserved, UII, 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 D).

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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-35 and Figure Amd.1-25 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 Amd.1-35 (see Annex K 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 *Write* shall use the extended preamble shown in Figure Amd.1-14 or Figure Amd.1-18, 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 Amd.1-34 — *Write* command

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

Table Amd.1-35 — Tag reply to a successful *Write* command

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

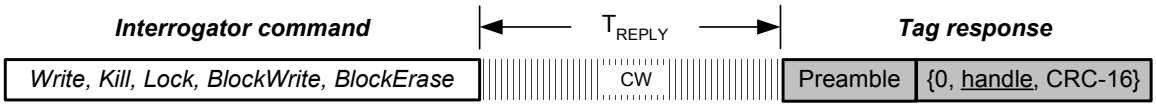


Figure Amd.1-25 — Successful *Write* sequence

9.3.2.10.3.4 *Kill* (mandatory)

Interrogators and tags shall implement the *Kill* command shown in Table Amd.1-36. *Kill* allows an interrogator to permanently disable a tag.

Kill contains 3 RFU bits. Interrogators shall set these bits to 000_2. 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 Amd.1-26. 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 valid 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).

Tags 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 Amd.1-26).

The tag reply to the first *Kill* shall be as shown in Table Amd.1-37. 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 Amd.1-37 and Figure Amd.1-25 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 Amd.1-37 (see Annex K 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 Amd.1-26.

A *Kill* shall be pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-14 or Figure Amd.1-18, 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 Amd.1-36 — *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 Amd.1-37 – Tag reply to the first *Kill* command

	RN	CRC-16
# of bits	16	16
description	handle	

Table Amd.1-38 — Tag reply to a successful *Kill* procedure

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

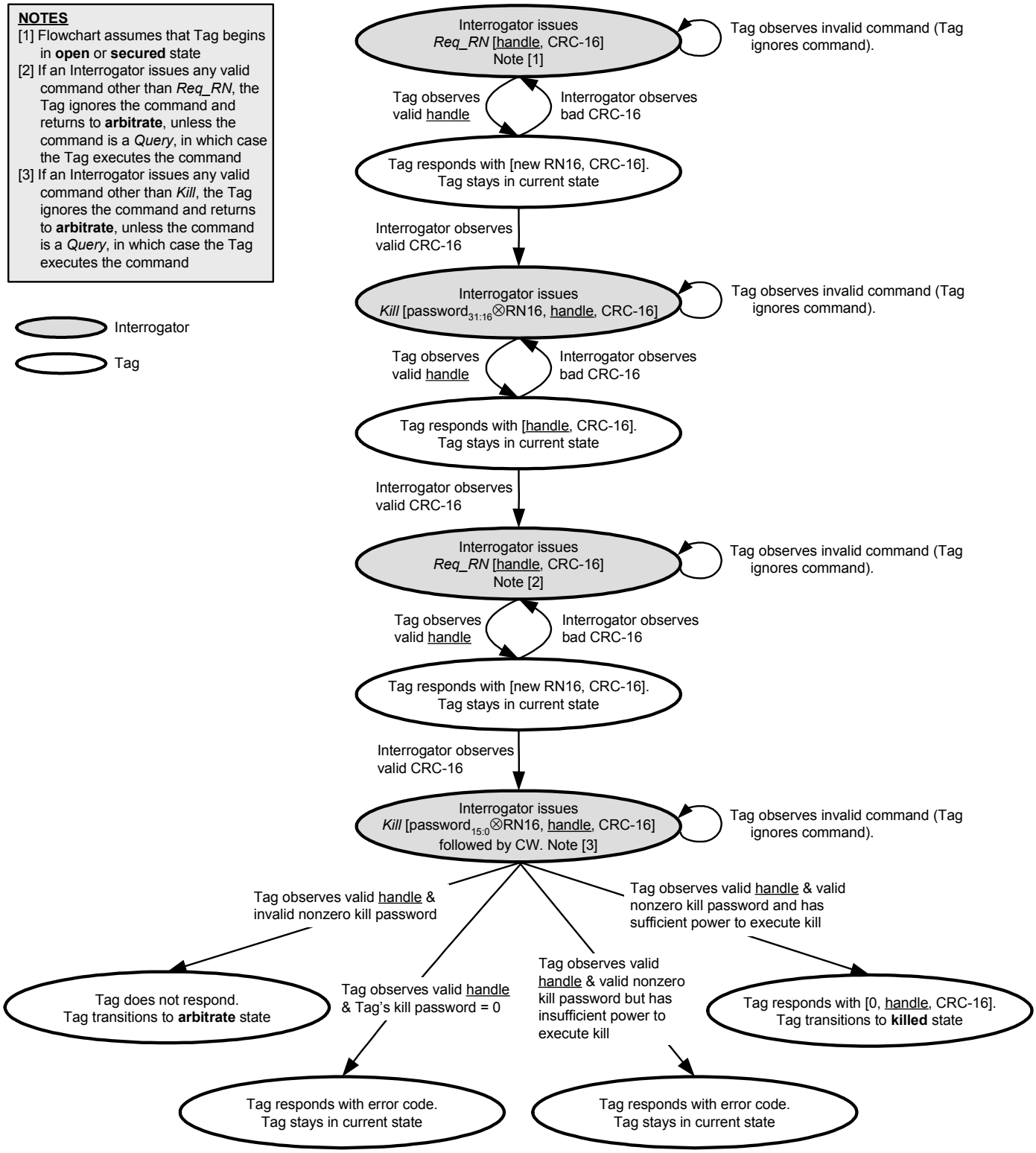


Figure Amd.1-26 — *Kill* procedure

9.3.2.10.3.5 *Lock* (mandatory)

Interrogators and tags shall implement the *Lock* command shown in Table Amd.1-39 and Figure Amd.1-27. 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: De-assert 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 Amd.1-41.

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 K).

Permalock bits, once asserted, cannot be de-asserted. If a tag receives a *Lock* whose payload attempts to de-assert a previously asserted permalock bit, the tag shall ignore the *Lock* and backscatter an error code (see Annex K). 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 Amd.1-27, 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 K). 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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-40 and Figure Amd.1-25 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 Amd.1-40 (see Annex K 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 Amd.1-14 or Figure Amd.1-18, 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 Amd.1-39 — Lock command

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

Table Amd.1-40 — Tag reply to a Lock command

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

Lock-Command Payload

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Kill Mask		Access Mask		UII Mask		TID Mask		User Mask		Kill Action		Access Action		UII Action		TID Action		User Action	

Masks and Associated Action Fields

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

Figure Amd.1-27 — Lock payload and usage

Table Amd.1-41 — 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.

9.3.2.10.3.6 *Access* (optional)

Interrogators and tags may implement an *Access* command; if they do, the command shall be as shown in Table Amd.1-42. *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 Amd.1-22) 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 Amd.1-28*.* 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 valid 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 pre-pended with a frame-sync (see 9.3.1.2.8).

The tag reply to an *Access* command shall be as shown in Table Amd.1-43. 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 and returns to **arbitrate**. The reply includes a CRC-16 calculated over the handle.

Table Amd.1-42 — *Access* command

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

Table Amd.1-43 — Tag reply to an *Access* command

	RN	CRC-16
# of bits	16	16
description	handle	

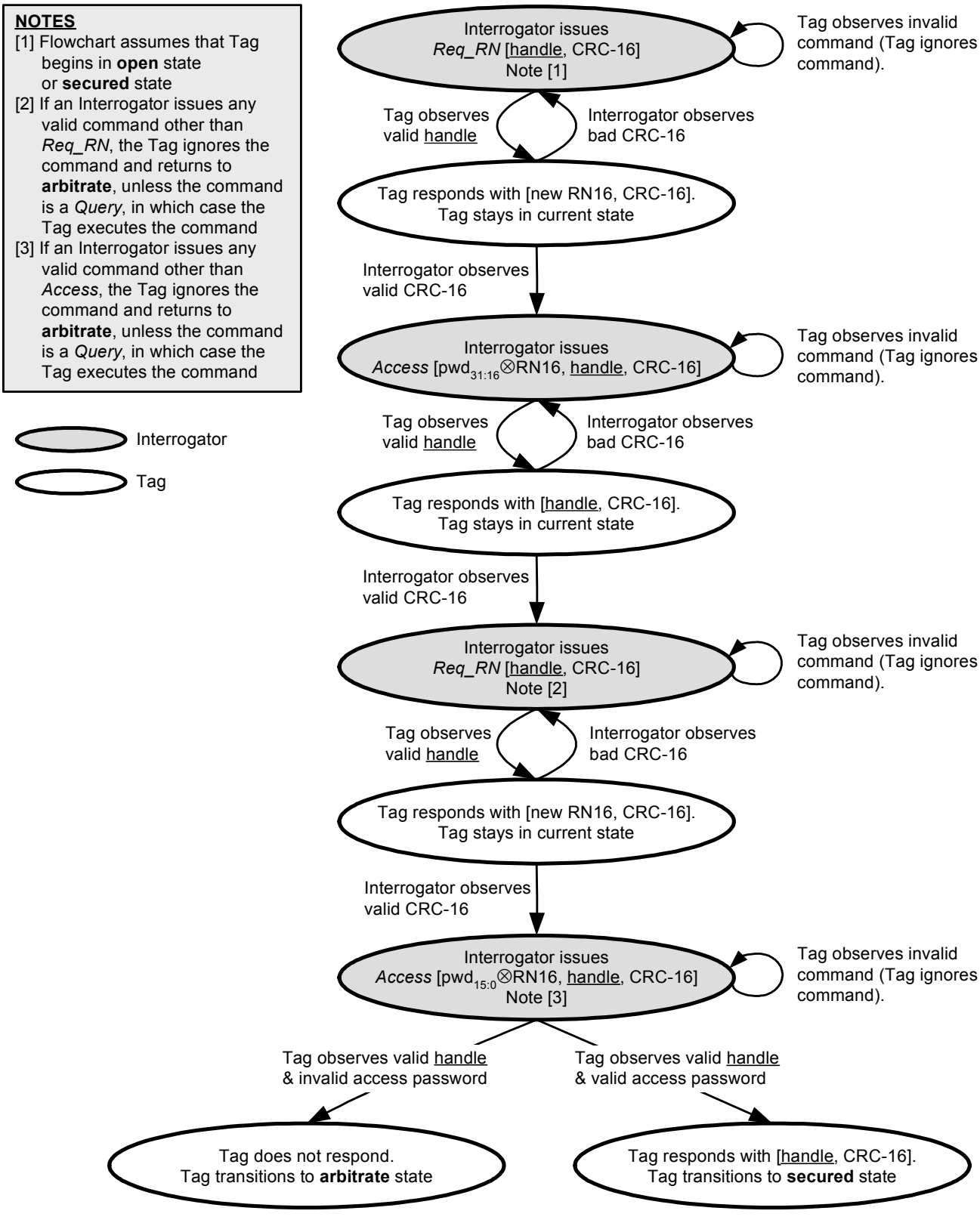


Figure Amd.1-28 — *Access* procedure

9.3.2.10.3.7 *BlockWrite* (optional)

Interrogators and tags may implement a *BlockWrite* command; if they do, they shall implement it as shown in Table Amd.1-44. *BlockWrite* allows an interrogator to write multiple words in a tag's Reserved, UII, TID, or User memory using a single command. *BlockWrite* has the following fields:

MemBank specifies whether the *BlockWrite* occurs in Reserved, UII, TID, or User memory. *BlockWrite* commands shall apply to a single memory bank. Successive *BlockWrites* may apply to different banks.

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 D).

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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 Amd.1-45 and Figure Amd.1-25 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 Amd.1-45 (see Annex K 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 Amd.1-14 or Figure Amd.1-18, 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 Amd.1-44 — *BlockWrite* command

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

Table Amd.1-45 — Tag reply to a successful *BlockWrite* command

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

9.3.2.10.3.8 *BlockErase* (optional)

Interrogators and tags may implement a *BlockErase* command; if they do, they shall implement it as shown in Table Amd.1-46. *BlockErase* allows an interrogator to erase multiple words in a tag's Reserved, UII, TID, or User memory using a single command. *BlockErase* has the following fields:

MemBank specifies whether the *BlockErase* occurs in Reserved, UII, TID, or User memory. *BlockErase* commands shall apply to a single memory bank. Successive *BlockErases* may apply to different banks.

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 D).

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 pre-pended with a frame-sync (see 9.3.1.2.8).

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 47 and Figure Amd.1-25 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 47 (see Annex K 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 Amd.1-14 or Figure Amd.1-18, 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 Amd.1-46 — *BlockErase* command

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

Table 47 — Tag reply to a successful *BlockErase* command

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

Page 122, Annex A

Replace Annex A with the following:

Annex A
(informative)

Calculation of 5-bit and 16-bit cyclic redundancy checks

A.1 Example CRC-5 encoder/decoder

An exemplary schematic diagram for a CRC-5 encoder/decoder is shown in Figure A.1, using the polynomial and preset defined in Table Amd.1-14.

To compute a CRC-5, first preload the entire CRC register (i.e. Q[4:0], Q4 being the MSB and Q0 the LSB) with the value 01001_2 (see Table A.1), then clock the data bits to be encoded into the input labelled DATA, MSB first. After clocking in all the data bits, Q[4:0] holds the CRC-5 value.

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

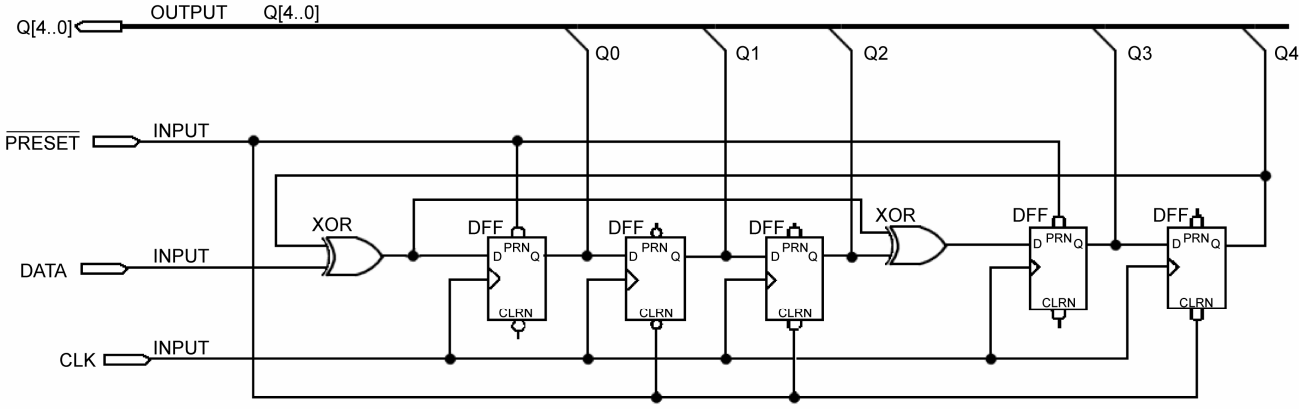


Figure A.1 — Example CRC-5 circuit

Table A.1 — CRC-5 register preload values

Register	Preload value
Q0	1
Q1	0
Q2	0
Q3	1
Q4	0

A.2 Example CRC-16 encoder/decoder

An exemplary schematic diagram for a CRC-16 encoder/decoder is shown in Figure A.2, using the polynomial and preset defined in Table Amd.1-13 (the polynomial used to calculate the CRC-16, $x^{16} + x^{12} + x^5 + 1$, is the CRC-CCITT International Standard).

To compute a CRC-16, first preload the entire CRC register (i.e. Q[15:0], Q15 being the MSB and Q0 the LSB) with the value $FFFF_h$. Second, clock the data bits to be encoded into the input labelled DATA, MSB first. After clocking in all the data bits, Q[15:0] holds the ones-complement of the CRC-16. Third, invert all the bits of Q[15:0] to produce the CRC-16.

There are two methods to check a CRC-16:

Method 1: First preload the entire CRC register (Q[15:0]) with the value $FFFF_h$, then clock the received data and CRC-16 {data, CRC-16} bits into the input labelled DATA, MSB first. The CRC-16 check passes if the value in Q[15:0]=$1D0F_h$.

Method 2: First preload the entire CRC register (Q[15:0]) with the value $FFFF_h$. Second, clock the received data bits into the input labelled DATA, MSB first. Third, invert all bits of the received CRC-16, and clock the inverted CRC-16 bits into the input labelled DATA, MSB first. The CRC-16 check passes if the value in Q[15:0]=0000_h.

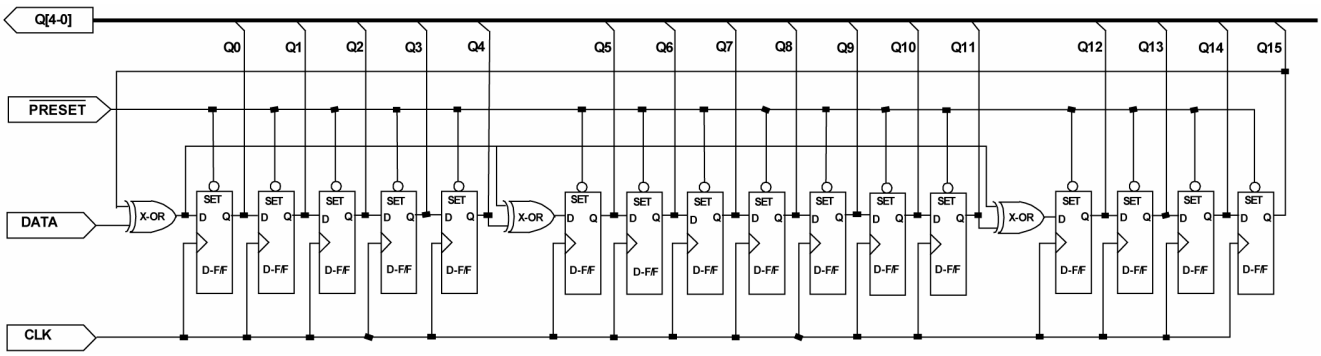


Figure A.2 — Example CRC-16 circuit

A.3 Example CRC-16 calculations

Example 1: This example shows the CRC-16 that a Type C tag would compute at powerup. As shown in Figure Amd.1-20, UII memory in Type C tags contains a CRC-16 starting at address 00_h, PC bits starting at address 10_h, and zero or more UII words starting at address 20_h. Table A.2 shows the CRC-16 that a tag would compute and logically map into UII memory at powerup, for the indicated example PC and UII word values. In each successive column, one more word of UII memory is written, with the entire UII memory written in the rightmost column. The indicated PC values correspond to the number of UII words written, with PC bits 15_h–$1F_h$ set to zero. Entries marked N/A mean that that word of UII memory is not included as part of the CRC calculation.

Table A.2 – UII memory contents for an example Type C tag

UII word starting address	UII word contents	UII word values						
00_h	CRC-16	$E2F0_h$	$CCAE_h$	$968F_h$	$78F6_h$	$C241_h$	$2A91_h$	1835_h
10_h	PC	0000_h	0800_h	1000_h	1800_h	2000_h	2800_h	3000_h
20_h	UII word 1	N/A	1111_h	1111	1111_h	1111_h	1111_h	1111_h
30_h	UII word 2	N/A	N/A	2222_h	2222_h	2222_h	2222_h	2222_h
20_h	UII word 3	N/A	N/A	N/A	3333_h	3333_h	3333_h	3333_h
40_h	UII word 4	N/A	N/A	N/A	N/A	4444_h	4444_h	4444_h
50_h	UII word 5	N/A	N/A	N/A	N/A	N/A	5555_h	5555_h
60_h	UII word 6	N/A	N/A	N/A	N/A	N/A	N/A	6666_h

Example 2: This example (a) computes the CRC that a Type B interrogator would use when sending a SUCCESS command (code 09_h) to a tag, then (b) performs the check a tag would perform when verifying the received command. The interrogator sends the packet shown in Table A.3; only the SUCCESS command (09_h) is used in the CRC-16 calculation.

Computing the CRC-16: Table A.4 shows the bit values in the 16 CRC registers of Figure A.2 as the command code 09_h is shifted, bit-by-bit, into the CRC circuit. The CRC-16 bits actually transmitted by the interrogator are inverted from those shown in step 8 of Table A.4, namely $8F26_h$.

Table A.3 — Example Type B command packet

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

Table A.4 — Values of Q[15:0] for the command in Table A.3

Step	Bit input (SUCCESS command)	CRC register values Q[15:0]
1	0	$EFDF_h$
2	0	$CF9F_h$
3	0	$8F1F_h$
4	0	$0E1F_h$
5	1	$0C1F_h$
6	0	$183E_h$
7	0	$307C_h$
8	1	$70D9_h$

Checking the CRC-16: As described in A.2, a tag may use choose one of two methods to verify the CRC-16 in a received command. Table A.5 shows the first method, in which the tag clocks the received data and CRC-16 {data, CRC-16} bits into the input labelled DATA, MSB first, and verifies that the value in Q[15:0]=$1D0F_h$. Table A.5 assumes, at step 0, that the CRC-16 circuit was already preloaded with $FFFF_h$ and the data bits clocked in, leaving the value in Q[15:0]=$70D9_h$ prior to inputting the CRC-16. Table A.5 shows the bit values in the 16 CRC registers of Figure A.2 as the CRC-16 bits are shifted, one-by-one, into the CRC circuit.

Table A.5 — First method for checking a CRC-16 for the {command, CRC-16} from Table A.4

Step	Input (received CRC-16)	CRC register values Q[15:0]
0		$70D9_h$
1	1	$F193_h$
2	0	$F307_h$
3	0	$F62F_h$
4	0	$FC7F_h$
5	1	$F8FE_h$
6	1	$F1FC_h$
7	1	$E3F8_h$
8	1	$C7F0_h$
9	0	$9FC1_h$
10	0	$2FA3_h$
11	1	$4F67_h$
12	0	$9ECE_h$
13	0	$2DBD_h$
14	1	$4B5B_h$
15	1	8697_h
16	0	$1D0F_h$

Table A.6 shows the second method of checking the CRC, in which the tag clocks the received data bits into the input labelled DATA, MSB first, then inverts and clocks all bits of the received CRC-16 into the input labelled DATA, MSB first, and verifies that the value in Q[15:0]=0000_h. Table A.5 assumes, at step 0, that the CRC-16 circuit was already preloaded with $FFFF_h$ and the data bits clocked in, leaving the value in Q[15:0]=$70D9_h$ prior to inputting the inverted CRC-16. Table A.5 shows the bit values in the 16 CRC registers of Figure A.2 as the inverted CRC-16 bits are shifted, one-by-one, into the CRC circuit.

Table A.6 — Second method for checking a CRC-16 for the {command, CRC-16} from Table A.4

Step	Input (inverted received CRC-16)	CRC register values Q[15:0]
0		$70D9_h$
1	0	$E1B2_h$
2	1	$C364_h$
3	1	$86C8_h$
4	1	$0D90_h$
5	0	$1B20_h$
6	0	3640_h
7	0	$6C80_h$
8	0	$D900_h$
9	1	$B200_h$
10	1	6400_h
11	0	$C800_h$
12	1	9000_h
13	1	2000_h
14	0	4000_h
15	0	8000_h
16	1	0000_h

Page 133, Annex C

Insert the following new annexes after Annex C.

Annex D
(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 International Standard 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 D.1.

Table D.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 International Standard uses EBV-8 values to represent memory addresses and mask lengths.

Annex E
(normative)

State-transition tables

State-transition Tables E.1 to E.7 shall define a tag's response to interrogator commands. The term "handle" used in the state-transition tables is defined in 9.3.2.4.5; error codes are defined in Table K.2; "slot" is the slot-counter output shown in Figure 19 and detailed in Annex L; "–" in the "Action" column means that a tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

E.1 Present state: Ready

Table E.1 — Ready state-transition table

Command	Condition	Action	Next State
Query [a]	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 de-assert **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 [b]	all	–	**ready**

[a] *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.

[b] "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.

E.2 Present state: Arbitrate

Table E.2 — Arbitrate state-transition table

Command	Condition	Action	Next State
Query [a,b]	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 [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	all	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or de-assert **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 [c]	all	–	**arbitrate**

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

b *Query* and *QueryAdjust* instruct a tag to load a new random value into its slot counter.

c "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.

E.3 Present state: Reply

Table E.3 — Reply state-transition table

Command	Condition	Action	Next State
Query [a, b]	slot=0; matching **inventoried** & **SL** flags	backscatter new RN16	**reply**
	slot<>0; matching **inventoried** & **SL** flags	–	**arbitrate**
	otherwise	–	**ready**
QueryRep	all	–	**arbitrate**
QueryAdjust [b]	slot=0	backscatter new RN16	**reply**
	slot<>0	–	**arbitrate**
ACK	valid RN16	backscatter {PC, UII, CRC-16} or {00000_2, truncated UII, CRC-16}	**acknowledged**
	invalid RN16	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	all	–	**arbitrate**
Select	all	assert or de-assert **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	T2 > 20.0Tpri (see Figure Amd.1-19)	–	**arbitrate**
Invalid [c]	all	–	**reply**

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

b *Query* and *QueryAdjust* instruct a tag to load a new random value into its slot counter.

c "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.

E.4 Present state: Acknowledged

Table E.4 — Acknowledged state-transition table

Command	Condition	Action	Next State
Query [a]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] 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, UII, CRC-16} or {00000_2, truncated UII, 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 de-assert **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 > 20.0Tpri (see Figure Amd.1-19)	–	**arbitrate**
Invalid [c]	all	–	**acknowledged**

[a] *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.

[b] As described in 9.3.2.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

[c] "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.

E.5 Present state: Open

Table E.5 — Open state-transition table

Command	Condition	Action	Next State
Query [a]	slot=0; matching **inventoried** [b] & **SL** flags	backscatter new RN16; transition **inventoried** [b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried** [b] & **SL** flags	transition **inventoried** [b] 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, UII, CRC-16} or {00000_2, truncated UII, CRC-16}	**open**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**open**
	invalid handle	–	**open**
Select	all	assert or de-assert **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 Amd.1-26)	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 Amd.1-28)	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 [c]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**open**
	valid commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**arbitrate**

[a] *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.

[b] As described in 9.3.2.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

[c] "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.

E.6 Present state: Secured

Table E.6 — Secured state-transition table

Command	Condition	Action	Next State
Query [a]	slot=0; matching **inventoried**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] 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, UII, CRC-16} or {00000_2, truncated UII, CRC-16}	**secured**
	invalid handle	–	**arbitrate**
NAK	all	–	**arbitrate**
Req_RN	valid handle	backscatter new RN16	**secured**
	invalid handle	–	**secured**
Select	all	assert or de-assert **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 Amd.1-26)	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 Amd.1-28)	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 [c]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**secured**
	valid commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**arbitrate**

a *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.

b As described in 9.3.2.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

c "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.

E.7 Present state: Killed

Table E.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 [a]	all	–	**killed**

[a] "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 F
(normative)
Command-Response Tables

Command-response tables F.1 to F.17 shall define a tag's response to interrogator commands. The term "handle" used in the state-transition tables is defined in 9.3.2.4.5; error codes are defined in Table K.2; "slot" is the slot-counter output shown in Figure Amd.1-22 and detailed in Annex L; "–" in the "Response" column means that a tag neither modifies its **SL** or **inventoried** flags nor backscatters a reply.

F.1 Command response: Power-up

Table F.1 — Power-up command-response table

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

F.2 Command response: *Query*

Table F.2 — Query[a] 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**[b] & **SL** flags	backscatter new RN16; transition **inventoried**[b] from *A*→*B* or *B*→*A* if and only if new session matches prior session	**reply**
	slot<>0; matching **inventoried**[b] & **SL** flags	transition **inventoried**[b] 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**

[a] *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.

[b] As described in 9.3.2.8, a tag transitions its **inventoried** flag prior to evaluating the condition.

F.3 Command response: *QueryRep*

Table F.3 — *QueryRep* command-response table [a]

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**

[a] See Table F.17 for the tag response to a *QueryRep* whose session parameter does not match that of the current inventory round.

F.4 Command response: *QueryAdjust*

Table F.4 — *QueryAdjust* [a] command-response table[b]

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**

[a] *QueryAdjust*, in the arbitrate or reply states, instructs a tag to load a new random value into its slot counter.
[b] See Table F.17 for the tag response to a *QueryRep* whose session parameter does not match that of the current inventory round.

F.5 Command response: *ACK*

Table F.5 — *ACK* command-response table

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

F.6 Command response: *NAK*

Table F.6 — *NAK* command-response table

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

F.7 Command response: *Req_RN*

Table F.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**

F.8 Command response: *Select*

Table F.8 — *Select* command-response table

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

F.9 Command response: *Read*

Table F.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**

F.10 Command response: *Write*

Table F.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**

F.11 Command response: *Kill*

Table F.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**

NOTE 1: See also Figure Amd.1-26

F.12 Command response: *Lock*

Table F.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**

F.13 Command response: *Access*

Table F.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**

NOTE 1: See also Figure Amd.1-28

F.14 Command response: *BlockWrite*

Table F.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**

F.15 Command response: *BlockErase*

Table F.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**

F.16 Command response: T_2 timeout

Table F.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 Amd.1-19)	–	**arbitrate**
open	all	–	**open**
secured	all	–	**secured**
killed	all	–	**killed**

F.17 Command response: Invalid command

Table F.17 — Invalid command table

Starting State	Condition	Response	Next State
ready[a]	all	–	**ready**
arbitrate[b]	all	–	**arbitrate**
reply[b]	all	–	**reply**
acknowledged[b]	all	–	**acknowledged**
open[b]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**open**
	valid commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**arbitrate**
secured[b]	all, excluding commands interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (see Figure Amd.1-26 and Figure Amd.1-28).	–	**secured**
	valid commands, except *Req_RN*, interspersed between successive *Kill* or *Access* commands in a kill or access sequence, respectively (Figure Amd.1-26 and Figure Amd.1-28).	–	**arbitrate**
killed[a]	all	–	**killed**

[a] "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] "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 G
(informative)

Example slot-count *(Q)* selection algorithm

G.1 Example algorithm an interrogator might use to choose *Q*

Figure G.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.

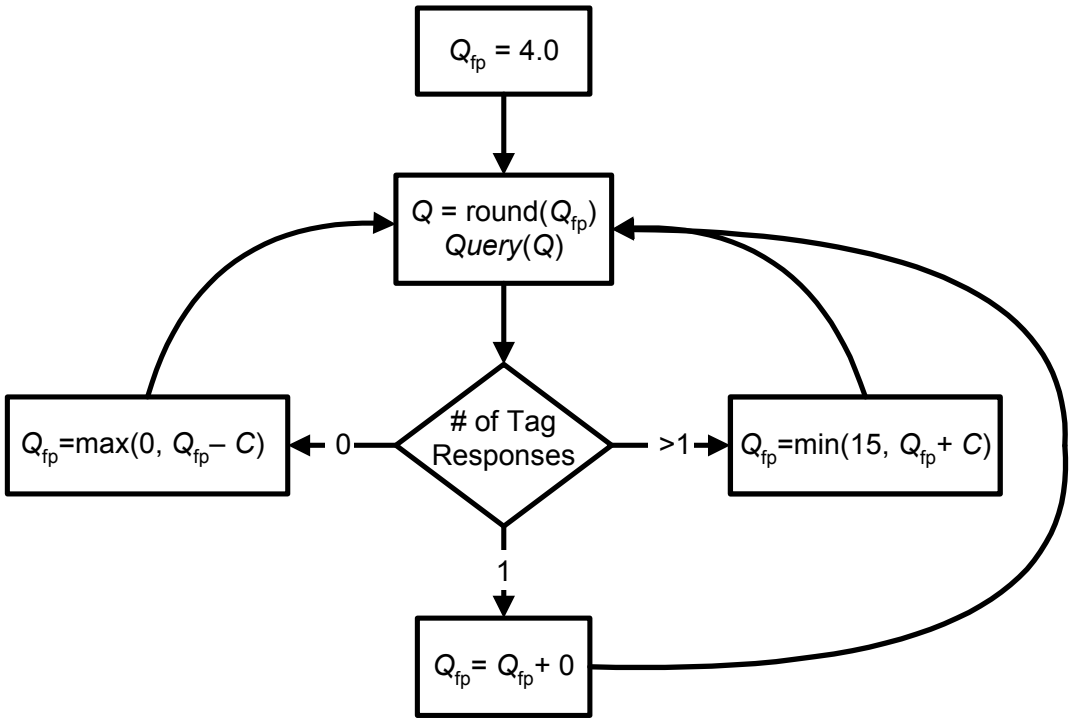


Figure G.1 — Example algorithm for choosing the slot-count parameter *Q*

Annex H
(informative)

Example of tag inventory and access

H.1 Example inventory and access of a single tag

Figure H.1 shows the steps by which an interrogator inventories and accesses a single tag.

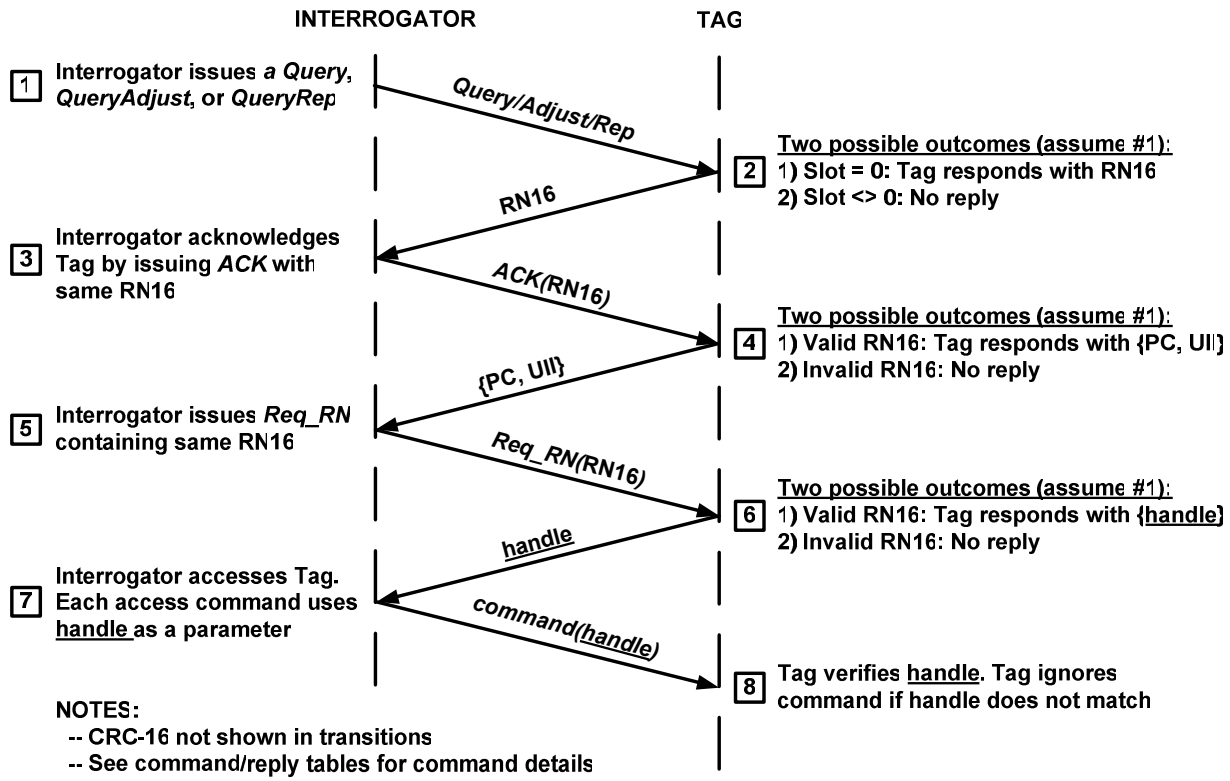


Figure H.1 — Example of tag inventory and access

Annex I
(normative)

Dense- and multiple-interrogator channelised signalling

This Annex describes channelised signalling in the optional dense- and multiple-interrogator modes. It provides several alternative methods that interrogators may use, where permitted by local regulatory authorities, to manage frequency-band usage.

I.1 Dense-interrogator modes

In environments containing two and more interrogators, the range and rate at which interrogators singulate tags can be improved by preventing interrogator transmissions from colliding with tag responses, either temporally or spectrally. This Annex describes time-division multiplexing (TDM) and frequency-division multiplexing (FDM) methods that can minimize interrogator-on-tag collisions. If permitted by local regulations, interrogators that are claimed to operate in dense-interrogator environments shall support one of the TDM or FDM methods described below, determined using the algorithm in Figure I.1. Regardless of the choice, Interrogator signalling (both modulated and CW) shall be centred in a channel with the frequency accuracy specified in 9.3.1.2.1, and interrogator transmissions shall satisfy the dense-interrogator transmit mask in Figure Amd.1-10. If an interrogator uses SSB-ASK modulation, the transmit spectrum shall be centred in the channel during R=>T signalling, and the CW shall be centred in the channel during tag backscatter.

TDM: 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.

FDM: Interrogator transmissions and tag responses shall be separated spectrally, using one of the three frequency plans described below.

Channel-boundary backscatter: Interrogator transmissions shall be centred in channels, and tag backscatter shall be situated at channel boundaries.

Adjacent-channel backscatter: Interrogator transmissions shall be centred in odd-numbered channels, and tag backscatter shall be situated in even-numbered channels.

In-channel backscatter: Interrogator transmissions shall be centred in channels, and tag backscatter shall be situated near but within the channel boundaries.

I.1.1 Examples of dense-interrogator mode operation

Figure I.2, shows examples of the single TDM and three FDM dense-Interrogator modes defined in I.1. For optimum performance, this specification recommends that interrogators choose BLF and M to allow a guardband between interrogator signalling and tag responses.

Example 1: TDM

ERC REC 70-03E Annex 1 allows the band from 869.4–869.65 MHz to be used as a single 250kHz channel. By the algorithm in Figure I.1, the dense-interrogator mode will be TDM. Example 1 of Figure I.2 shows one possible operating mode, in which interrogator transmissions use DSB-ASK modulation with Tari=25μs, and tag backscatter is 20 kbit/s data on an 80 kHz subcarrier (BLF=80kHz, M=4).

Example 2: FDM Channel-boundary backscatter

FCC 15.247, dated October 2000, authorizes frequency-hopping operation in the ISM band from 902–928 MHz with 500kHz maximum channel width, and does not prohibit channel-boundary backscatter. By the algorithm in Figure I.1, interrogators will use 500kHz channels with channel-boundary backscatter. Table I.1 shows the channelisation and channel numbering; example 2 of Figure I.2 shows interrogator transmissions using PR-ASK modulation with Tari=25μs, and 62.5 kbit/s tag data backscatter on a 250kHz subcarrier (BLF=250kHz; M=4). Interrogators centre their R=>T signalling in the channels shown in Table I.1, with transmissions unsynchronized in time, hopping among channels.

Example 3: FDM Adjacent-channel backscatter

ERC REC 70-03E Annex 11 specifies fifteen 200kHz channels in the 865 – 868 MHz frequency range, and does not prohibit adjacent-channel backscatter. By the algorithm in Figure I.1, interrogators will use 200kHz channels with adjacent-channel backscatter. Figure I.3 shows the channel numbering; example 3 of Figure I.2 shows interrogator transmissions using SSB-ASK modulation with Tari=25μs, and 50 kbit/s tag data backscatter on a 200kHz subcarrier (BLF=200kHz, M=4).

Example 4: FDM In-channel backscatter

A hypothetical regulatory environment allocates four 500kHz channels and disallows adjacent-channel and channel-boundary backscatter. By the algorithm in Figure I.1, interrogators will use 500kHz channels with in-channel backscatter. Example 4 of Figure I.2 shows interrogator transmissions using PR-ASK modulation with Tari=25μs, and 25 kbit/s tag data backscatter on a 200kHz subcarrier (BLF=200kHz, M=8).

I.2 Channelisation in multiple- and dense-interrogator environments

When Interrogators in multiple- and dense-Interrogator environments instruct tags to use subcarrier backscatter, the Interrogators shall adopt the channelisation determined by the algorithm in I.1. When interrogators in multiple- and dense-interrogator environments instruct tags to use FM0 backscatter, the interrogators shall adopt a channelisation that is in accordance with local regulations. Regardless of the backscatter data encoding, interrogator transmissions shall satisfy the multiple- or dense- interrogator transmit mask in clause 9.3.1.2.11.

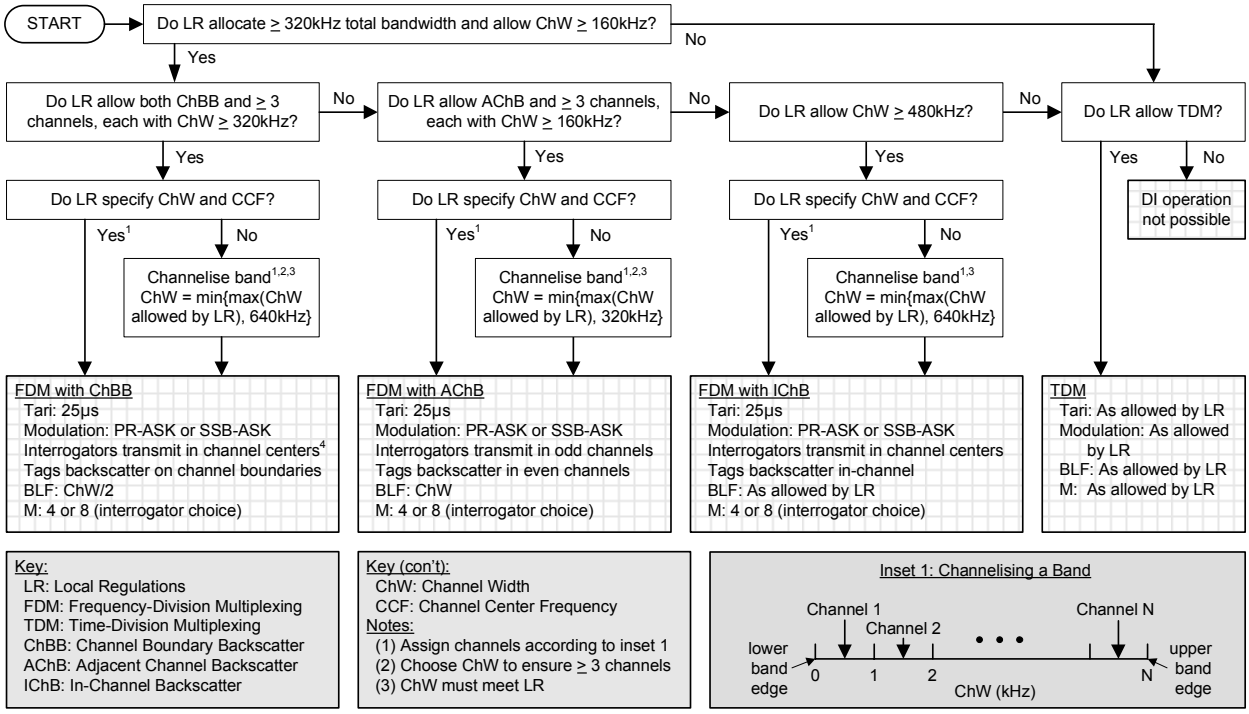


Figure I.1 — Algorithm for determining channelisation and dense-interrogator-mode parameters

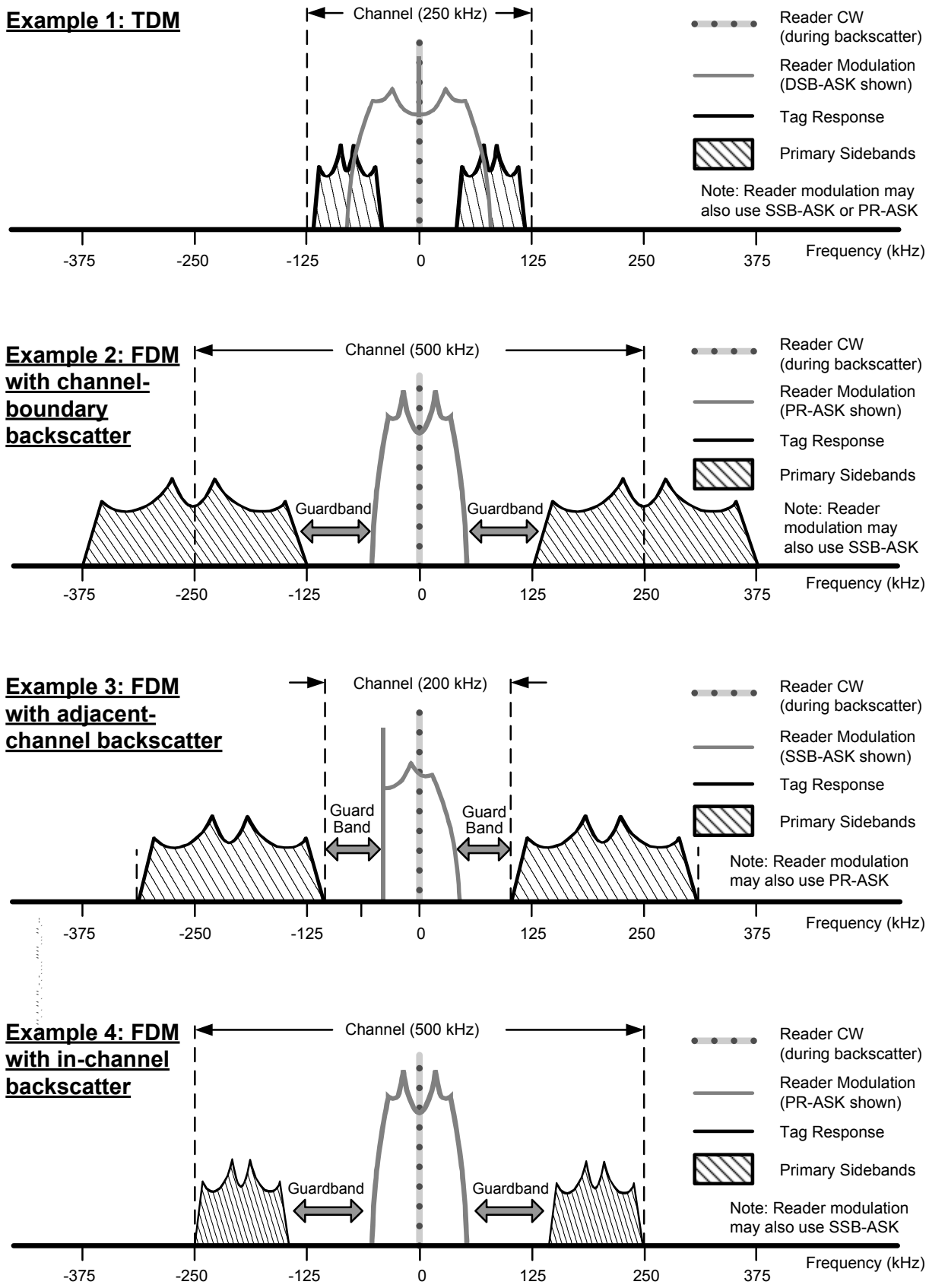


Figure I.2 — Examples of dense-interrogator mode operation

I.2.1 Example channelisation

In the FCC 15.247 environment from example 2 above, Interrogators will use the channelisation in Table I.1.

Table I.1 — Channelisation for Example 2

Commanded tag backscatter format	Channel width	Channel centre 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

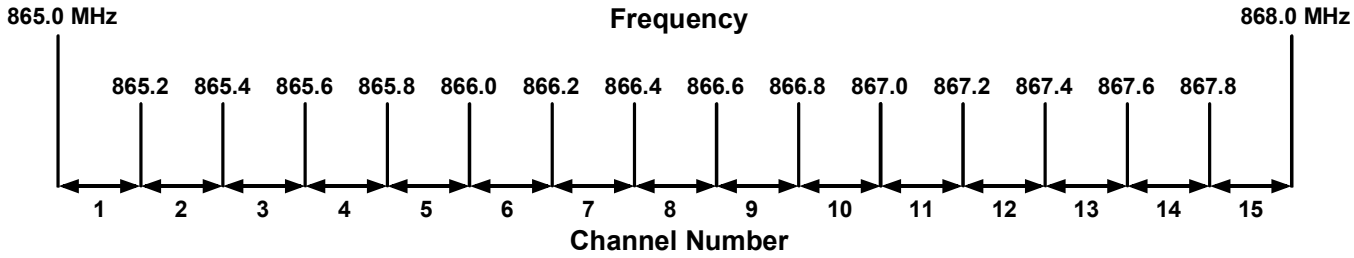


Figure I.3 — Channel numbering for Example 3

Annex J
(informative)

Interrogator-to-tag link modulation

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

Figure J.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.

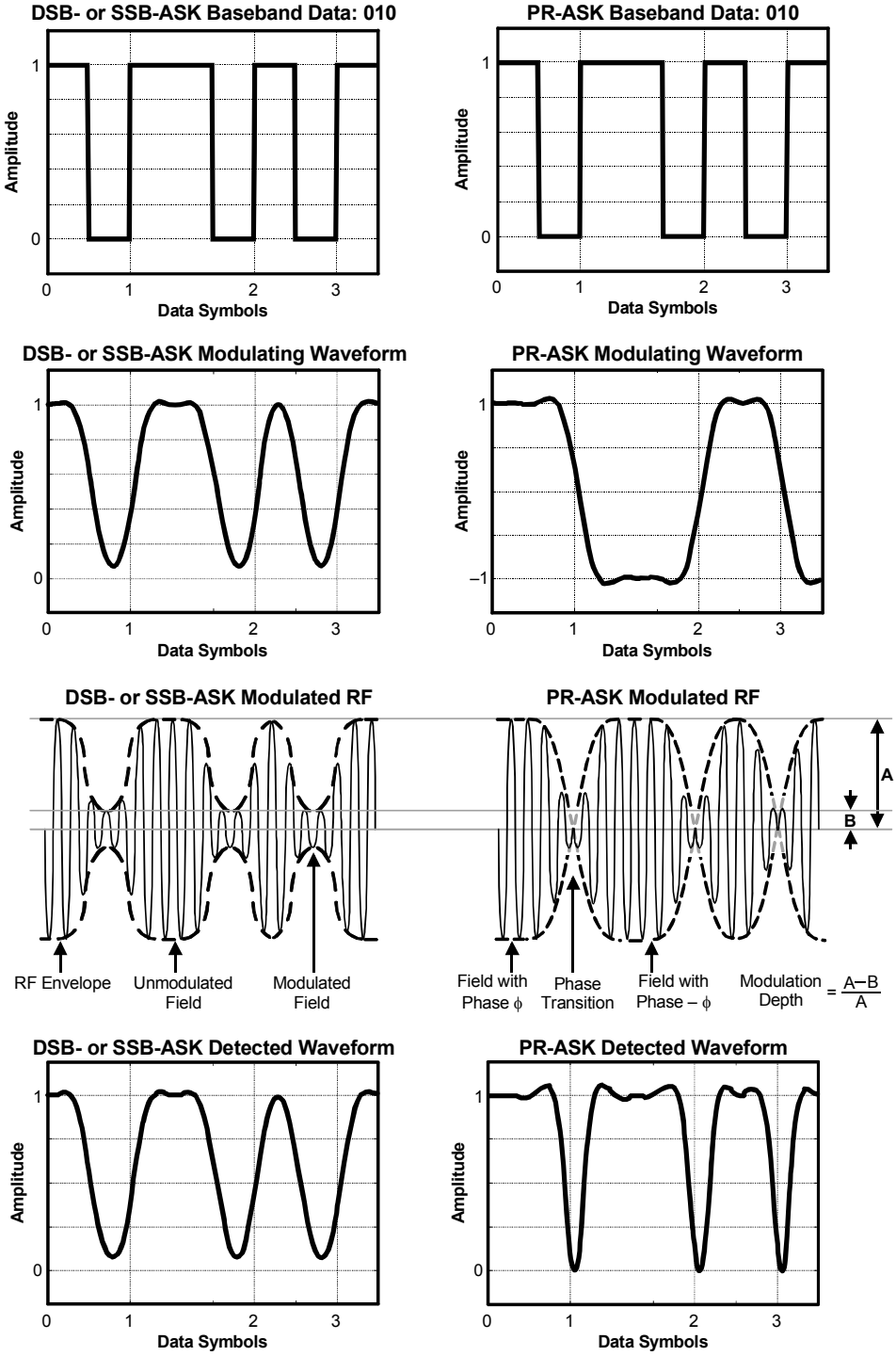


Figure J.1 — Interrogator-to-tag modulation

Annex K
(normative)

Error codes

K.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 K.1 instead of its normal reply.

If the tag supports error-specific codes, it shall use the error-specific codes shown in Table K.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 K.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 K.1 — Tag-error reply format

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

Table K.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 L
(normative)

Slot counter

L.1 Slot-counter operation

As described in 9.3.2.4.8, tags implement a 15-bit slot counter. As described in 9.3.2.8, 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 9.3.2.5), 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*.

A tag in the **arbitrate** state shall decrement its slot counter every time it receives a *QueryRep* command, transitioning to the **reply** state and backscattering an RN16 when its slot-counter value reaches 0000_h. A tag whose slot-counter value reached 0000_h, who replied, and who was not acknowledged (including a tag that responded to the original *Query* and was not acknowledged) returns to **arbitrate** with a slot-counter value of 0000_h.

A tag that returns to **arbitrate** with a slot-counter value of 0000_h shall decrement its slot-counter from 0000_h to $7FFF_h$ (i.e. the slot counter rolls over) at the next *QueryRep* with matching session. Because the slot-counter value is now nonzero, the tag remains in **arbitrate**. Slot counters implements continuous counting, meaning that, after a slot counter rolls over it begins counting down again from $7FFF_h$, effectively preventing subsequent tag replies until the tag receives either a *Query* or a *QueryAdjust* and loads a new random value into its slot counter.

Annex E and Annex F contain tables describing a tag's response to interrogator commands; "slot" is a parameter in these tables.

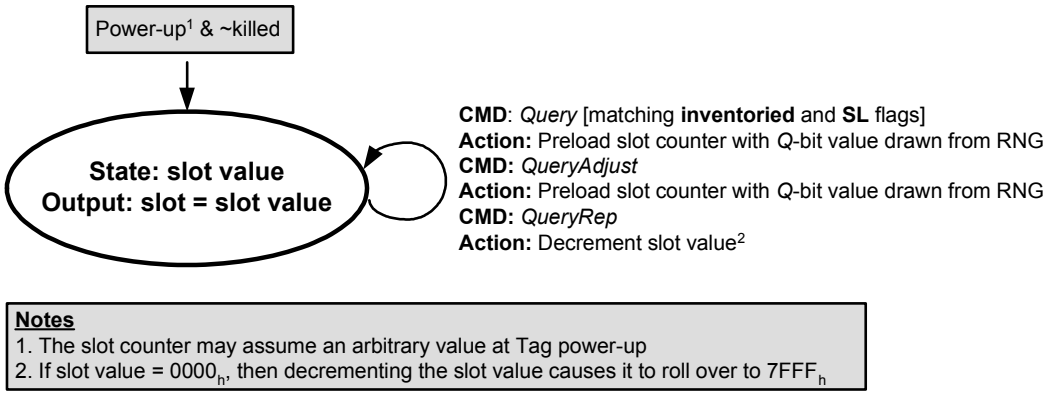


Figure L.1 — Slot-counter state diagram

Annex M
(informative)

Example data-flow exchange

M.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 UII is 64 bits in length.

The tag's access password is $ACCEC0DE_h$.

The tag's kill password is $DEADC0DE_h$.

The 1st half of the access password EXORed with RN16_2 = $ACCE_h \otimes 1602_h = BACC_h$.

The 2nd half of the access password EXORed with RN16_3 = $C0DE_h \otimes 1603_h = D6DD_h$.

M.2 Tag memory contents and lock-field values

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

Table M.1 — Tag memory contents

Memory Bank	Memory Contents	Memory Addresses	Memory Values
TID	TID[15:0]	10_h–$1F_h$	$54E2_h$
	TID[31:16]	00_h–$0F_h$	$A986_h$
UII	UII[15:0]	50_h–$5F_h$	3210_h
	UII[31:16]	40_h–$4F_h$	7654_h
	UII[47:32]	30_h–$3F_h$	$BA98_h$
	UII[63:48]	20_h–$2F_h$	$FEDC_h$
	PC[15:0]	10_h–$1F_h$	2000_h
	CRC-16[15:0]	00_h–$0F_h$	as calculated (see Annex A)
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 M.2 — Lock-field values

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

M.3 Data-flow exchange and command sequence

The data-flow exchange follows the *Access* procedure outlined in Figure Amd.1-28 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 M.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 M.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 N
(informative)

Optional tag features

N.1 General

The following options are available to tags according this International Standard.

N.2 Optional tag passwords

N.2.1 Kill password

A tag may optionally implement a kill password. A tag that does not implement a kill password operates as if it has a zero-valued kill password that is permanently read/write locked. See 9.3.2.1.1.

N.2.2 Access password

A tag may optionally implement an access password. A tag that does not implement an access password operates as if it has a zero-valued access password that is permanently read/write locked. See 9.3.2.1.2

N.3 Optional Tag memory banks and memory-bank sizes

N.3.1 Reserved memory

Reserved memory is optional. If a tag does not implement either a kill password or an access password then the tag need not physically implement Reserved memory. Because a tag with non-implemented passwords operates as if it has zero-valued password(s) that are permanently read/write locked, these passwords must still be logically addressable in Reserved memory at the memory locations specified in 9.3.2.1.1.1 and 9.3.2.1.1.2.

N.3.2 UII memory

UII memory is required, but its size is vendor-defined. The minimum size is 32 bits, to contain a 16-bit CRC and 16-bit PC. UII memory may be larger than 32 bits, to contain an UII whose vendor-specified length may be 16 bits to 496 bits in 16-bit increments. See 9.3.2.1.2.

N.3.3 TID memory

TID memory is required, but its size is vendor-defined. The minimum-size TID memory contains an 8-bit ISO/IEC 15963 allocation class identifier, as well as sufficient identifying information for an interrogator to uniquely identify the custom commands and/or optional features that a tag supports. TID memory may optionally contain vendor-specific data. See 9.3.2.1.3.

N.3.4 User memory:

User memory is optional. See 9.3.2.1.4, 9.3.2.1.4.1 and 9.3.2.1.4.2.

N.4 Optional tag commands

***Proprietary*:** A tag may support proprietary commands. See 9.2.5.

***Custom*:** A tag may support custom commands. See 9.2.4.

***Access*:** A tag may support the *Access* command. See 9.3.2.10.3.6.

***BlockWrite*:** A tag may support the *BlockWrite* command. See 9.3.2.10.3.7.

***BlockErase*:** A tag may support the *BlockErase* command. See 9.3.2.10.3.8.

N.5 Optional tag error-code reporting format

A tag may support error-specific or non-error-specific error-code reporting. See Annex K.

N.6 Optional tag backscatter modulation format

A tag may support ASK and/or PSK backscatter modulation. See 9.3.1.3.1.

Page 134, Bibliography

Delete the following references, which have been moved to Clause 3:

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

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

[4] 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*

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

Page 134, Bibliography

Insert the following references and renumber accordingly:

[1] ARIB STD-T81, *RFID Equipment Using Frequency Hopping System For Specified Low Power Radio Station*

[2] CEPT/ERC Recommendation 70-03, *Relating to the use of Short Range Devices (SRD), Annex 11*

[3] ETSI EN 300 220 (all parts), *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW*

[4] ETSI EN 302 208-1, *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*

[5] ETSI EN 302 208-2, *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*

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

[7] ISO/IEC 18000-2, *Information technology — Radio frequency identification for item management — Part 2: Parameters for air interface communications below 135 kHz*

[8] ISO/IEC 18000-3, *Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz*

[9] ISO/IEC 18000-4, *Information technology — Radio frequency identification for item management — Part 4: Parameters for air interface communications at 2,45 GHz*

[10] ISO/IEC 18000-7, *Information technology — Radio frequency identification for item management — Part 7: Parameters for active air interface communications at 433 MHz*

[11] RCR STD-1, *RFID Equipment For Premises Radio Station*

[12] RCR STD-29, *RFID Equipment For Specified Low Power Radio Station*

[13] US Code of Federal Regulations (CFR) Title 47, Chapter I, Part 15. “Radio Frequency Devices”; U.S. Federal Communications Commission

[14] EPCglobal™: EPC™ Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID, Protocol for Communications at 860 MHz – 960 MHz, Version 1.0.9

ICS 35.040

Price based on 124 pages