
Identification cards — Test methods —

Part 6:
Proximity cards

*Cartes d'identification — Méthodes d'essai —
Partie 6: Cartes de proximité*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011



COPYRIGHT PROTECTED DOCUMENT

© ISO 2011

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Foreword	vi
1 Scope	1
2 Normative references	1
3 Terms, definitions, symbols and abbreviated terms	2
3.1 Terms and definitions	2
3.2 Symbols and abbreviated terms	3
4 Default items applicable to the test methods	5
4.1 Test environment	5
4.2 Pre-conditioning	5
4.3 Default tolerance	6
4.4 Spurious Inductance	6
4.5 Total measurement uncertainty	6
5 Apparatus and circuits for test of ISO/IEC 14443-1 and ISO/IEC 14443-2 parameters	6
5.1 Minimum requirements for measurement instruments	6
5.1.1 Oscilloscope	6
5.2 Calibration coil	6
5.2.1 Size of the calibration coil card	6
5.2.2 Thickness and material of the calibration coil card	7
5.2.3 Coil characteristics	7
5.3 Test PCD assembly	7
5.3.1 Test PCD antenna	8
5.3.2 Sense coils	8
5.3.3 Assembly of Test PCD	8
5.4 Reference PICC	9
5.4.1 Dimensions of the Reference PICC	9
5.4.2 Reference PICC construction	10
5.4.3 Reference PICC resonance frequency tuning	11
6 Test of ISO/IEC 14443-1 parameters	12
6.1 PCD tests	12
6.1.1 Alternating magnetic field	12
6.2 PICC tests	12
6.2.1 Alternating magnetic field	12
6.2.2 Static electricity test	13
7 Test of ISO/IEC 14443-2 parameters	14
7.1 PCD tests	14
7.1.1 PCD field strength	14
7.1.2 PCD field strength supporting operation with "Class 1" PICCs	15
7.1.3 Power transfer PCD to PICC	16
7.1.4 Modulation index and waveform	16
7.1.5 Load modulation reception	17
7.2 PICC tests	18
7.2.1 PICC transmission	18
7.2.2 PICC reception	19
7.2.3 PICC resonance frequency (informative)	21
7.2.4 "Class 1" PICC maximum loading effect	21
8 Test of ISO/IEC 14443-3 and ISO/IEC 14443-4 parameters	22
8.1 PCD tests	22
8.2 PICC tests	22

Annex A (normative) Test PCD Antenna	23
A.1 Test PCD Antenna layout including impedance matching network	23
A.2 Impedance matching network	27
A.2.1 Impedance matching network for a bit rate of $fc/128$	27
A.2.2 Impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$	28
Annex B (informative) Test PCD Antenna tuning	29
Annex C (normative) Sense coil	31
C.1 Sense coil layout	31
C.2 Sense coil assembly	32
Annex D (normative) Reference PICC antenna layout	33
Annex E (normative) Modulation index and waveform analysis tool	34
E.1 Overview	34
E.2 Sampling	34
E.3 Filtering	35
E.4 Envelope generation	36
E.5 Envelope smoothing	36
E.6 Modulation index determination	36
E.7 Timing determination	36
E.8 Overshoot and undershoot determination	37
E.9 Program of the modulation index and waveform analysis tool (informative)	37
E.9.1 structures.h	38
E.9.2 ffrm.h	39
E.9.3 ffrm.c	40
E.9.4 hilbert.h	44
E.9.5 hilbert.c	45
E.9.6 functs.c	52
Annex F (informative) Program for the evaluation of the spectrum	80
Annex G (normative) Additional PICC test methods	85
G.1 PICC-test-apparatus and accessories	85
G.1.1 Emulating the I/O protocol	85
G.1.2 Generating the I/O character timing in reception mode	85
G.1.3 Measuring and monitoring the RF I/O protocol	85
G.1.4 Protocol Analysis	85
G.1.5 RFU fields	85
G.2 Relationship of test methods versus base standard requirement	86
G.3 Test method for initialization of the PICC Type A	87
G.3.1 Introduction	87
G.3.2 Scenario G.1: Polling	87
G.3.3 Testing of the PICC Type A state transitions	88
G.3.4 Scenario G.13: Handling of Type A anticollision	109
G.3.5 Handling of RATS	110
G.3.6 Handling of PPS request	110
G.3.7 Scenario G.20: Handling of FSD	111
G.4 Test method for initialization of the PICC Type B	112
G.4.1 Introduction	112
G.4.2 Scenario G.21: Polling	112
G.4.3 Scenario G.22: PICC Reception	113
G.4.4 Testing of the PICC Type B state transitions	114
G.4.5 Scenario G.28: Handling of Type B anticollision	124
G.4.6 Handling of ATTRIB	126
G.4.7 Scenario G.31: Handling of Maximum Frame Size	127
G.5 Test methods for logical operation of the PICC Type A or Type B	127
G.5.1 Introduction	127
G.5.2 PICC reaction to ISO/IEC 14443-4 Scenarios	128
G.5.3 Handling of PICC error detection	138
G.5.4 PICC reaction on CID	140
G.5.5 PICC reaction on NAD	142

G.6	Reported results	143
Annex H	(normative) Additional PCD test methods	147
H.1	PCD-test-apparatus and accessories	147
H.1.1	Test method	147
H.1.2	PCD-test-apparatus structure	147
H.1.3	PCD-test-apparatus interface	148
H.1.4	Emulating the I/O protocol	148
H.1.5	Generating the I/O character timing in transmission mode	148
H.1.6	Measuring and monitoring the RF I/O protocol	149
H.1.7	Protocol Analysis	149
H.1.8	Protocol activation procedure	149
H.1.9	Scenario	149
H.1.10	UT, LT and PCD behavior	150
H.1.11	Relationship of test methods versus base standard requirement	151
H.2	Type A specific test methods	151
H.2.1	Frame delay time PICC to PCD	151
H.2.2	Request Guard Time	152
H.2.3	Handling of bit collision during ATQA	153
H.2.4	Handling of anticollision loop	153
H.2.5	Handling of RATS and ATS	157
H.2.6	Handling of PPS response	160
H.2.7	Frame size selection mechanism	161
H.2.8	Handling of Start-up Frame Guard Time	163
H.2.9	Handling of the CID during activation by the PCD	164
H.3	Type B specific test methods	165
H.3.1	I/O transmission timing	165
H.3.2	Frame size selection mechanism	166
H.3.3	Handling of the CID during activation by the PCD	167
H.4	Test method for logical operations of the PCD	169
H.4.1	Handling of the polling loop	169
H.4.2	Reaction of the PCD to request for waiting time extension	170
H.4.3	Error detection and recovery	173
H.4.4	Handling of NAD during chaining	182
H.5	Continuous monitoring of packets sent by the PCD	183
H.5.1	RFU fields	183
H.5.2	RFU values	183
H.5.3	R-block	183
H.5.4	S-block	183
H.5.5	PCB	183
H.5.6	Type A initialization frames	183
H.5.7	Apparatus	183
H.5.8	Procedure	183
H.5.9	Test report	184
H.6	Reported results	184
Annex I	(normative) High bit rate selection test methods for PCD	188
I.1	Apparatus	188
I.2	Procedure	188
I.2.1	Procedure for Type A	188
I.2.2	Procedure for Type B	193
	Bibliography	199

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.

ISO/IEC 10373-6 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

This second edition cancels and replaces the first edition (ISO/IEC 10373-6:2001), which has been technically revised. It also includes the Amendments ISO/IEC 10373-6:2001/Amd.1:2007, ISO/IEC 10373-6:2001/Amd.2:2003, ISO/IEC 10373-6:2001/Amd.3:2006, ISO/IEC 10373-6:2001/Amd.4:2006 and ISO/IEC 10373-6:2001/Amd.5:2007.

ISO/IEC 10373 consists of the following parts, under the general title *Identification cards — Test methods*:

- *Part 1: General characteristics*
- *Part 2: Cards with magnetic stripes*
- *Part 3: Integrated circuit cards with contacts and related interface devices*
- *Part 5: Optical memory cards*
- *Part 6: Proximity cards*
- *Part 7: Vicinity cards*
- *Part 8: USB-ICC*
- *Part 9: Optical memory cards — Holographic recording method*

Identification cards — Test methods —

Part 6: Proximity cards

1 Scope

ISO/IEC 10373 defines test methods for characteristics of identification cards according to the definition given in ISO/IEC 7810. Each test method is cross-referenced to one or more base standards, which can be ISO/IEC 7810 or one or more of the supplementary standards that define the information storage technologies employed in identification card applications.

NOTE 1 Criteria for acceptability do not form part of ISO/IEC 10373, but will be found in the International Standards mentioned above.

NOTE 2 Test methods defined in this part of ISO/IEC 10373 are intended to be performed separately. A given proximity card or object, or proximity coupling device, is not required to pass through all the tests sequentially.

This part of ISO/IEC 10373 defines test methods which are specific to proximity cards and objects, and proximity coupling devices, defined in ISO/IEC 14443-1:2008, ISO/IEC 14443-2:2010, ISO/IEC 14443-3:— and ISO/IEC 14443-4:2008. ISO/IEC 10373-1 defines test methods which are common to one or more integrated circuit card technologies and other parts deal with other technology-specific tests.

2 Normative references

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

ISO/IEC 7810:2003, *Identification cards — Physical characteristics*

ISO/IEC 14443-1:2008, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 1: Physical characteristics*

ISO/IEC 14443-2:2010, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 2: Radio frequency power and signal interface*

ISO/IEC 14443-3:—¹⁾, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision*

ISO/IEC 14443-4:2008, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 4: Transmission protocol*

IEC 61000-4-2:2008, *Electromagnetic compatibility (EMC) — Part 4-2: Testing and measurement techniques — Electrostatic discharge immunity test*

1) To be published.

3 Terms, definitions, symbols and abbreviated terms

For the purposes of this document, the terms, definitions, abbreviations and symbols given in ISO/IEC 14443-1:2008, ISO/IEC 14443-2:2010, ISO/IEC 14443-3:—, ISO/IEC 14443-4:2008 and the following apply.

NOTE Elements in bold square brackets [] are optional definitions.

3.1 Terms and definitions

3.1.1

base standard

standard which the test method is used to verify conformance to

3.1.2

Cascade Levels

number of cascade levels of the PICC

3.1.3

"Class 1" PICC

PICC whose antenna is located as defined in ISO/IEC 14443-1:2008 and which passes the "Class 1" PICC maximum loading effect test defined in 7.2.4

3.1.4

Command Set

set describing the PICC commands during initialization and anticollision

NOTE See ISO/IEC 14443-3:—, 6.4 for PICC Type A and ISO/IEC 14443-3:—, 7.5 for PICC Type B.

3.1.5

Mute

no response within a specified timeout, e.g. expiration of FWT

3.1.6

PICC States

different PICC states during initialization and anticollision

NOTE See ISO/IEC 14443-3:—, 6.3 for PICC Type A and ISO/IEC 14443-3:—, 7.4 for PICC Type B.

3.1.7

Scenario

defined typical protocol and application specific communication to be used with the test methods defined in this part of ISO/IEC 10373

3.1.8

Test Initial State

TIS

element from PICC States that is the PICC state before performing a specific PICC command from Command Set

3.1.9

test method

method for testing characteristics of identification cards for the purpose of confirming their compliance with International Standards

3.1.10

Test Target State

TTS

element from PICC States that is the PICC state after performing a specific PICC command from Command Set

3.2 Symbols and abbreviated terms

(xxxxx)b	Data bit representations
'XY'	Hexadecimal notation, equal to XY in base 16
ATA(cid)	Answer to ATTRIB, i.e. (mbli+cid CRC_B), with mbli an arbitrary hex value (see ISO/IEC 14443-3:—, 7.11)
ATTRIB(cid, fsdi)	Default ATTRIB command with PUPI from ATQB, CID = cid and Maximum Frame Size Code value = fsdi i.e. ('1D' PUPI cid fsdi '01 00' CRC_B)
DUT	Device under test
ESD	Electrostatic Discharge
$I(c)_n(\text{inf } [,CID = cid] [,NAD = nad] [,~CRC])$	ISO/IEC 14443-4 I-block with chaining bit $c \in \{1,0\}$, block number $n \in \{1,0\}$ and information field inf. By default no CID and no NAD will be transmitted. If $CID = cid \in \{0 \dots 15\}$ is specified, it will be transmitted as second parameter. If $NAD = nad \in \{0 \dots 'FF'\}$ is specified it will be transmitted as third parameter (or second parameter if no CID is transmitted). If the literal '~CRC' is not specified, a valid CRC corresponding to the type of the PICC will be transmitted by default (i.e. CRC_A or CRC_B)
IUT	Implementation Under Test (ISO/IEC 9646); within the scope of this part of ISO/IEC 10373, IUT represents the PCD under the test
LT	Lower Tester (ISO/IEC 9646), the PICC-emulation part of the PCD-test-apparatus
m	Modulation index
Mute	No response within a specified timeout
N/A	Not applicable
PPS(cid, dri, dsi)	Default PPS request with CID = cid, DRI = dri and DSI = dsi, i.e. ('D' + cid '11' dsi × 4 + dri CRC_A)
$R(\text{ACK } [,CID = cid] [,~CRC])_n$	ISO/IEC 14443-4 R(ACK) Block with block number n. The definition of the optional CID and ~CRC symbols is as described in the $I(c)_n$ block above
$R(\text{NAK } [,CID = cid] [,~CRC])_n$	ISO/IEC 14443-4 R(NAK) Block with block number n. The definition of the optional CID and ~CRC symbols is as described in the $I(c)_n$ block above
RATS(cid, fsdi)	Default RATS command with CID = cid and FSDI value = fsdi i.e. ('E0' fsdi × 16 + cid CRC_A)
READY(I)	READY state in cascade level I, $I \in \{1, 2, 3\}$; e.g. READY(2) is a PICC cascade level 2

READY*(I)	READY* state in cascade level I, $I \in \{1, 2, 3\}$; e.g. READY*(2) is a PICC cascade level 2
REQB(N)	REQB command with N as defined in ISO/IEC 14443-3:—, 7.7.4
S(WTX)(WTXM [,CID = cid][,~CRC])	ISO/IEC 14443-4 S(WTX) block with parameter WTXM. The definition of the optional CID and ~CRC symbols is as described in the I(c) _n block above
S(DESELECT [,CID = cid] [,~CRC])	ISO/IEC 14443-4 S(DESELECT) block. The definition of the optional CID and ~CRC symbols is as described in the I(c) _n block above
SAK(cascade)	the SELECT(I) answer with the cascade bit (bit 3) set to (1)b
SAK(complete)	the SELECT(I) answer with the cascade bit (bit 3) set to (0)b
SEL(c)	Select code of level c (i.e. SEL(1) = '93', SEL(2) = '95', SEL(3) = '97')
SELECT(I)	SELECT command of cascade level I, i.e. SELECT(1) = ('93 70' UIDTX ₁ BCC CRC_A) SELECT(2) = ('95 70' UIDTX ₂ BCC CRC_A) SELECT(3) = ('97 70' UIDTX ₃ BCC CRC_A)
SLOTMARKER(n)	Slot-MARKER command with slot number n, i.e. (16 × (n - 1) + 5 CRC_B)
TB-PDU	Transmission Block Protocol Data Unit, which consists of either I-block, R-block or S-block
TEST_COMMAND1(1)	Default test command consisting of one unchained I-block NOTE This command depends on the negotiated maximum frame size value of the PICC.
TEST_COMMAND1(n), n > 1	Default test command consisting of n chained I-blocks (PCD chaining) NOTE This command depends on the negotiated maximum frame size value of the PICC.
TEST_COMMAND1(n) _k	INF field of k'th I-block chain of TEST_COMMAND1(n) NOTE This command depends on the negotiated maximum frame size value of the PICC.
TEST_COMMAND2(n), n > 1	Default test command which expects a response consisting of n chained I-blocks NOTE This command depends on the negotiated maximum frame size value of the PCD.
TEST_COMMAND3	Default test command consisting of one I-block which needs more than FWT time for execution
TEST_RESPONSE1(n)	INF field of the response to TEST_COMMAND1(n) NOTE This response is assumed to be always unchained.
TEST_RESPONSE2(n)	Response to TEST_COMMAND2(n)

	NOTE	This response depends on the negotiated maximum frame size value of the PCD.
TEST_RESPONSE2(n) _k		INF field of k'th I-block chain of TEST_RESPONSE2(n)
	NOTE	This response depends on the negotiated maximum frame size value of the PCD.
TEST_RESPONSE3		Response I-block to TEST_COMMAND3
	NOTE	This response is always assumed to be unchained.
TM- PDU		Test Management Protocol Data Unit (ISO/IEC 9646-1, PDU)
UIDTX _l		Transmitted UID 32-bit data at cascade level l (see Table 1)
UT		Upper Tester (ISO/IEC 9646), the master part of the PCD-test-apparatus
UT_APDU		Upper Tester Application Protocol Data Unit: a packet of data to be sent by the PCD to the LT through the RF interface
WUPB(N)		WUPB command with N as defined in ISO/IEC 14443-3:—, 7.7.4
~X		Bit sequence consisting of the inverted bits of bit sequence X or any other bit sequence different from X
X[[a..b]]		Bit subsequence of bit sequence X consisting of the bits between position a and b included. If a > b then the sequence is empty
X[[n]]		Bit at position n of bit sequence X. First bit is at position 1
X[n]		Byte at position n of bit sequence X. First byte is at position 1 (i.e. X[n] = X[[(n - 1) × 8 + 1..n × 8]])

Table 1 — Mapping from UID to UIDTX

Cascade level	Single UID PICC	Double UID PICC	Triple UID PICC
UIDTX ₁	UID0 UID1 UID2 UID3	'88' UID0 UID1 UID2	'88' UID0 UID1 UID2
UIDTX ₂	---	UID3 UID4 UID5 UID6	'88' UID3 UID4 UID5
UIDTX ₃	---	---	UID6 UID7 UID8 UID9

4 Default items applicable to the test methods

4.1 Test environment

Unless otherwise specified, testing shall take place in an environment of temperature 23 °C ± 3 °C (73 °F ± 5 °F) and of relative humidity 40 % to 60 %.

4.2 Pre-conditioning

No environmental pre-conditioning of PICCs or PCDs is required by the test methods in this document.

4.3 Default tolerance

Unless otherwise specified, a default tolerance of $\pm 5\%$ shall be applied to the quantity values given to specify the characteristics of the test equipment (e.g. linear dimensions) and the test method procedures (e.g. test equipment adjustments).

4.4 Spurious Inductance

Resistors and capacitors should have negligible inductance.

4.5 Total measurement uncertainty

The total measurement uncertainty for each quantity determined by these test methods shall be stated in the test report.

Basic information is given in ISO/IEC Guide 98-3:2008.

5 Apparatus and circuits for test of ISO/IEC 14443-1 and ISO/IEC 14443-2 parameters

This clause defines the test apparatus and test circuits for verifying the operation of a PICC or a PCD according to ISO/IEC 14443-1:2008 and ISO/IEC 14443-2:2010. The test apparatus includes:

- Measurement instruments (see 5.1);
- Calibration coil (see 5.2);
- Test PCD assembly (see 5.3);
- Reference PICC (see 5.4).

These are described in the following clauses.

5.1 Minimum requirements for measurement instruments

5.1.1 Oscilloscope

The digital sampling oscilloscope shall be capable of sampling at a rate of at least 500 million samples per second with a resolution of at least 8 bits at optimum scaling and shall have an overall minimum bandwidth of 250 MHz. The oscilloscope should have the capability to output the sampled data as a text file to facilitate mathematical and other operations such as windowing on the sampled data using external software programs (see Annexes E and F).

NOTE The overall bandwidth is the combination of oscilloscope and probing system bandwidth.

5.2 Calibration coil

This clause defines the size, thickness and characteristics of the calibration coil.

5.2.1 Size of the calibration coil card

The calibration coil card shall consist of an area which has the height and width of an ID-1 type defined in ISO/IEC 7810:2003 containing a single turn coil concentric with the card outline (see Figure 1).

ISO/IEC 7810 ID-1 outline

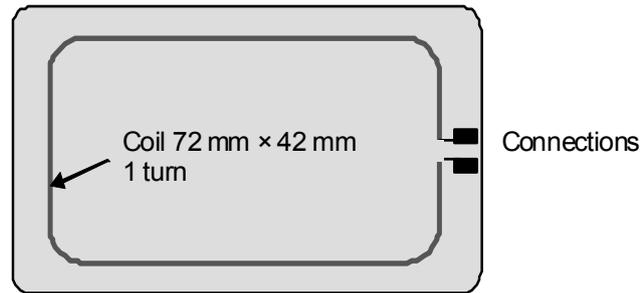


Figure 1 — Calibration coil

5.2.2 Thickness and material of the calibration coil card

The thickness of the calibration coil card shall be less than that of an ID-1 card. It shall be constructed of a suitable insulating material.

5.2.3 Coil characteristics

The coil on the calibration coil card shall have one turn. The outer size of the coil shall be 72 mm × 42 mm with corner radius 5 mm. Relative dimensional tolerance shall be ± 2 %.

NOTE 1 The area over which the field is integrated is approximately 3000 mm².

The coil shall be made as a printed coil on printed circuit board (PCB) plated with 35 µm copper. Track width shall be 500 µm with a relative tolerance of ± 20 %. The size of the connection pads shall be 1,5 mm × 1,5 mm.

NOTE 2 At 13,56 MHz the approximate inductance is 250 nH and the approximate resistance is 0,4 Ω.

A high impedance oscilloscope probe with an input admittance equivalent to a parallel capacitance $C_p < 14$ pF and a parallel resistance $R_p > 9$ kΩ at 13,56 MHz shall be used to measure the (open circuit) voltage induced in the coil. The resonance frequency of the calibration coil and connecting leads shall be above 60 MHz.

NOTE 3 A parasitic capacitance of the probe assembly of less than 35 pF normally ensures for the whole set a resonant frequency greater than 60 MHz.

The open circuit calibration factor for this coil is 0,32 V (rms) per A/m (rms) [Equivalent to 900 mV (peak-to-peak) per A/m (rms)].

NOTE 4 The high impedance oscilloscope probe ground connection should be as short as possible, less than 20 mm or coaxial connection.

5.3 Test PCD assembly

The test PCD assembly shall consist of a 150 mm diameter test PCD antenna and two parallel sense coils: sense coil a and sense coil b. The test set-up is shown in Figure 2. The sense coils shall be connected such that the signal from one coil is in opposite phase to the other. The 10 Ω potentiometer P1 serves to fine adjust the balance point when the sense coils are not loaded by a PICC or any magnetically coupled circuit. The capacitive load of the probe including its parasitic capacitance shall be less than 14 pF.

NOTE 1 The capacitance of the connections and of the oscilloscope probe should be kept to a minimum for reproducibility.

NOTE 2 In order to avoid any unintended misalignment in case of an unsymmetrical set-up the tuning range of the potentiometer P1 is only 10 Ω. If the set-up cannot be compensated by the 10 Ω potentiometer P1 the overall symmetry of the set-up should be checked.

NOTE 3 The high impedance oscilloscope probe ground connection should be as short as possible, less than 20 mm or coaxial connection.

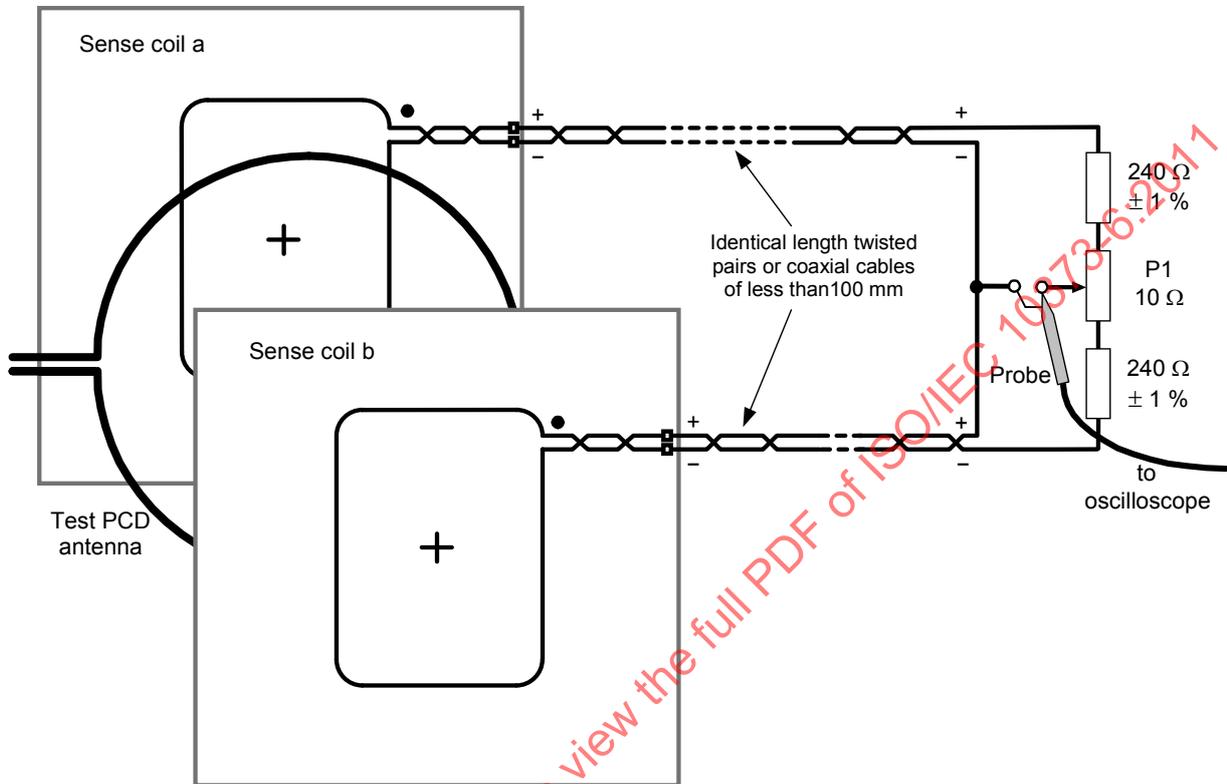


Figure 2 — Test set-up (principle)

5.3.1 Test PCD antenna

The test PCD antenna shall have a diameter of 150 mm and its construction shall conform to the drawings in Annex A.

The matching of the test PCD antenna should be accomplished by using an impedance analyzer or a network analyzer or a LCR meter. If either an impedance analyzer or a network analyzer or a LCR meter is not available, then the matching may be accomplished with the procedure given in Annex B.

5.3.2 Sense coils

The size of the sense coils shall be 100 mm × 70 mm with corner radius 10 mm. The sense coil construction shall conform to the drawings in Annex C.

5.3.3 Assembly of Test PCD

The sense coils and test PCD antenna shall be assembled parallel and with the sense and antenna coils coaxial and such that the distance between the active conductors is 37,5 mm as shown in Figure 3. The dimensional tolerance shall be better than ± 0,5 mm. The distance between the coil in the DUT and the calibration coil shall be equal with respect to the coil of the test PCD antenna.

NOTE These distances are chosen to represent the typical operating distance of the PICC.

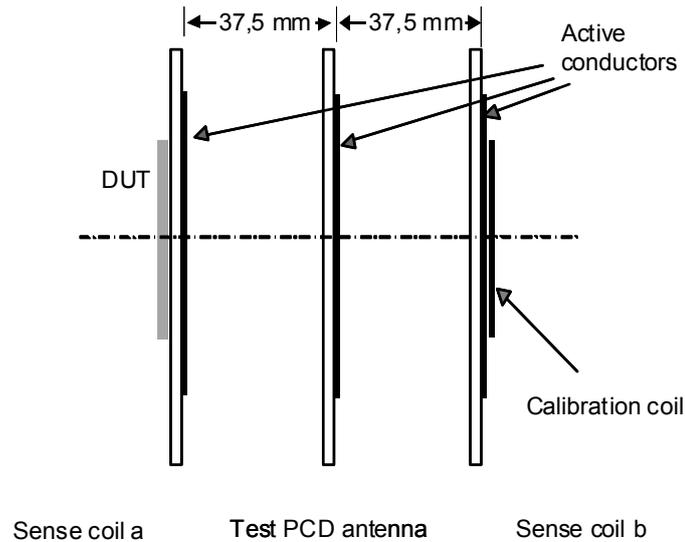


Figure 3 — Test PCD assembly

5.4 Reference PICC

A Reference PICC is defined to test the ability of a PCD to:

- generate a field strength of at least H_{min} and not exceeding H_{max} ;
- transfer power to a PICC;
- transmit a modulated signal to a PICC;
- receive a load modulation signal from the PICC;

in its operating volume.

5.4.1 Dimensions of the Reference PICC

The Reference PICC shall consist of an area containing the coils which has the height and width defined in ISO/IEC 7810:2003 for ID-1 type. An area external to this, containing the circuitry which emulates the required PICC functions, shall be appended in such a way as to allow insertion into the test set-ups and so as to cause no interference to the tests. The dimensions shall be as in Figure 4.

Outline ISO/IEC 7810
ID-1 type

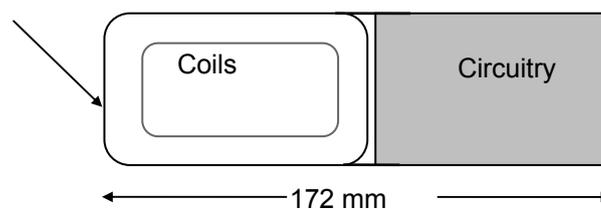


Figure 4 — Reference PICC dimensions

5.4.2 Reference PICC construction

The Reference PICC construction is defined in Annex D.

The Reference PICC shall have a circuit diagram as defined in Figure 5 and component values as defined in Table 2.

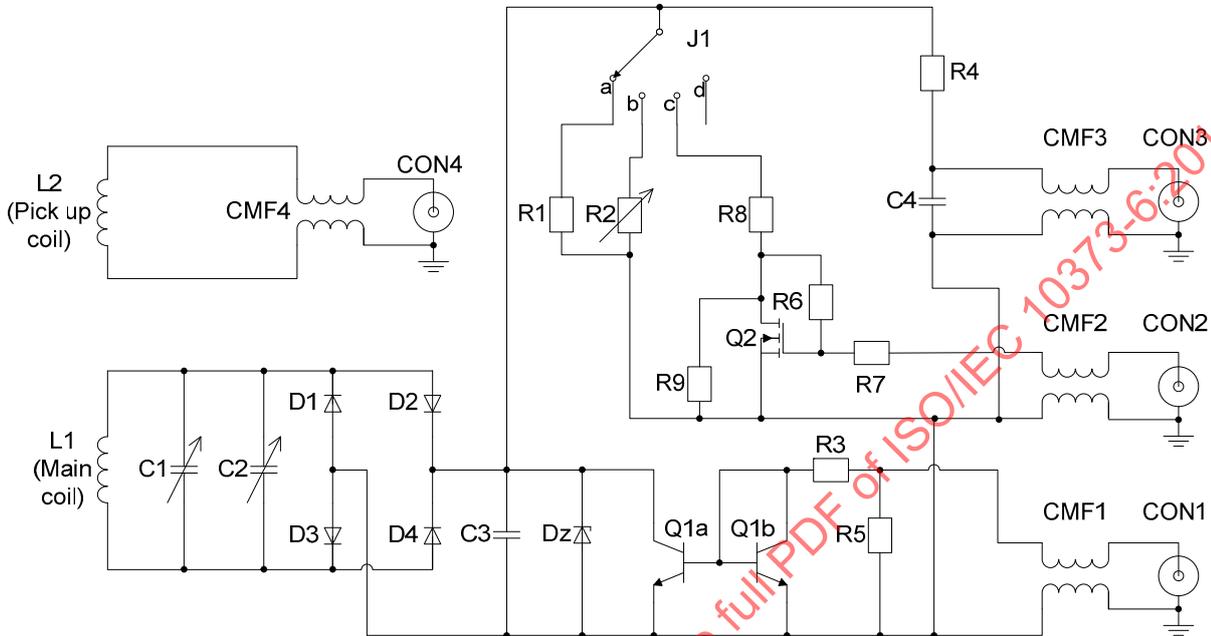


Figure 5 — Reference PICC circuit diagram

NOTE Position 'd' of jumper J1 is RFU.

Table 2 — Reference PICC components list

Component	Value	Component	Value
L1	See Annex D	C1	7 pF – 50 pF ^b
L2	See Annex D	C2	3 pF – 10 pF ^b
R1	1,8 kΩ	C3	27 pF
R2	0 kΩ – 2 kΩ ^a	C4	1 nF
R3	220 Ω	D1, D2, D3, D4	BAR43S or equivalent ^c
R4	51 kΩ	Dz	BZX84, 15 V or equivalent ^c
R5	51 Ω	Q1a, Q1b	BCV61A or equivalent
R6	500 kΩ	Q2	BSS83 or equivalent
R7	110 kΩ	CMF1, CMF2, CMF3, CMF4	ACM3225 -102-2P or equivalent
R8	51 Ω	CON1, CON2, CON3, CON4	RF connector
R9	1,5 kΩ		

^a A multi-turn potentiometer (turns ≥ 10) should be used.

^b Q - factor shall be higher than 100 at 13,56 MHz.

^c Care should be taken on parameters C_j (Junction capacitance), C_p (Package capacitance), L_s (Series inductance) and R_s (Series resistance) of equivalent diodes. Note that these values may not be available in the datasheet.

At CON1 the load modulation signal shall be applied. The load modulation can be determined in test PCD assembly. When not used, the load modulation signal generator shall be disconnected or set to 0 V.

With the voltage at CON2 the Reference PICC load can be adjusted until the required DC voltage shows at CON3.

The Reference PICC DC voltage shall be measured at CON3 using a high impedance voltmeter and the connection wires should be twisted or coaxial.

The PCD waveform parameters are picked up at CON4 using a high impedance oscilloscope probe. The high impedance oscilloscope probe ground connection should be as short as possible, less than 20 mm or coaxial connection.

5.4.3 Reference PICC resonance frequency tuning

The Reference PICC resonance frequency shall be calibrated with the following procedure:

- a) Set jumper J1 to position 'a'.
- b) Connect the calibration coil directly to a signal generator and the Reference PICC connector CON3 to a high impedance voltmeter. Connect all the other connectors to the same equipment as used for the tests.
- c) Locate the Reference PICC at a distance $d = 10$ mm above the calibration coil with the axes of the two coils (calibration coil and Reference PICC main coil) being congruent (see Figure 6).
- d) Drive the calibration coil with a sine wave set to the desired resonance frequency.
- e) Adjust the Reference PICC capacitors C1 and C2 to get maximum DC voltage at CON3.
- f) Adjust the signal generator drive level to read a DC voltage of 6 V at CON3.
- g) Repeat steps e) and f) until the maximum voltage after step e) is 6 V.
- h) Calibrate the Test PCD assembly to produce the H_{\min} operating condition on the calibration coil.
- i) Place the Reference PICC into the DUT position on the Test PCD assembly. Switch the jumper J1 to position 'b' and adjust R2 to obtain a DC voltage of 6 V measured at connector CON3. The operating field condition shall be verified by monitoring the voltage on the calibration coil and adjusted if necessary.
- j) Repeat steps b) to g) with the obtained value of R2.

NOTE Instead of a signal generator a vector network analyzer may be used if sufficient power is provided to produce 6 V at CON3 while reaching the maximum resistive part of the measured complex impedance of the calibration coil.

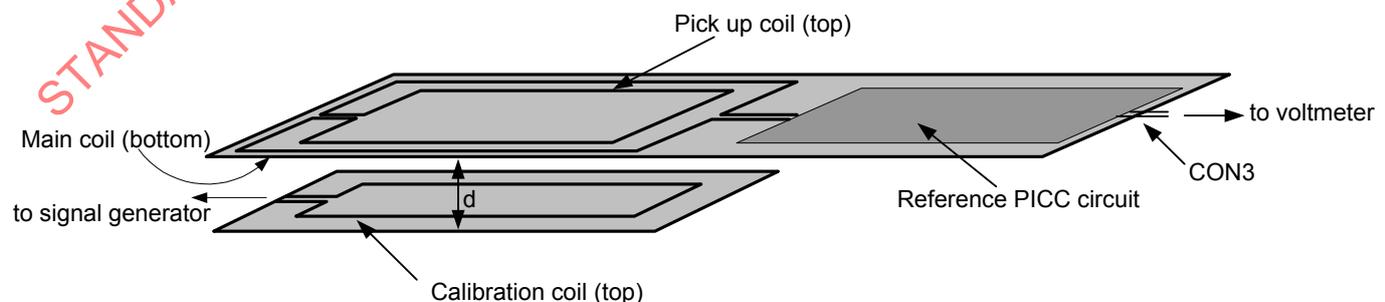


Figure 6 — Reference PICC frequency tuning set-up (principle)

6 Test of ISO/IEC 14443-1 parameters

6.1 PCD tests

6.1.1 Alternating magnetic field

6.1.1.1 Purpose

This test determines that the PCD generates a field not higher than the average value specified in ISO/IEC 14443-1:2008, in any possible PICC position.

6.1.1.2 Test procedure

- a) Tune the Reference PICC to 19 MHz as described in 5.4.3 steps a) to g).
- b) Calibrate the Test PCD assembly to produce the average field value specified in ISO/IEC 14443-1:2008, 4.4 on the calibration coil.
- c) Place the Reference PICC into the DUT position on the Test PCD assembly. Switch the jumper J1 to position 'b' and adjust R2 to obtain a DC voltage of 3 V measured at connector CON3. The operating field condition shall be verified by monitoring the voltage on the calibration coil and adjusted if necessary.

WARNING - R2 value should be between 55 and 65 Ω .

- d) Position the Reference PICC in any possible PICC position. The DC voltage at CON3 shall not exceed 3 V.
- e) If exceeded, use the same conversion factor to measure the maximum and average DC voltage and convert in field strength to check the maximum and the average field values specified in ISO/IEC 14443-1:2008, 4.4 in 30 s period.

6.1.1.3 Test report

The test report shall give the DC voltage measured at CON3.

6.2 PICC tests

6.2.1 Alternating magnetic field

The purpose of this test is to check the behavior of the PICC in relation to alternating magnetic field exposure at 13,56 MHz.

6.2.1.1 Apparatus

The test PCD assembly shall be used to produce the alternating magnetic field.

6.2.1.2 Test procedure

The procedure is as follows.

- a) Adjust the RF power delivered by the signal generator to the test PCD antenna to a field strength of 10 A/m (rms) as measured by the calibration coil.
- b) Place the PICC under test in the DUT position and readjust immediately the RF drive into the test PCD antenna to the required field strength if necessary.

- c) After 5 min, remove the PICC from the DUT position for at least 5 s.
- d) Adjust the RF power delivered by the signal generator to the test PCD antenna to a field strength of 12 A/m (rms) as measured by the calibration coil.
- e) Place the PICC under test in the DUT position and readjust immediately the RF drive into the test PCD antenna to the required field strength if necessary.
- f) Apply for 5 min an ASK 100 % modulation to this field with the following duty cycle:
 - 5 s at 0 A/m (rms);
 - 25 s at 12 A/m (rms).
- g) Check that the PICC operates as intended.

6.2.1.3 Test report

The test report shall state whether or not the PICC operates as intended.

6.2.2 Static electricity test

The purpose of this test is to check the behavior of the card IC in relation to electrostatic discharge (ESD) exposure in the test sample. The PICC under test is exposed to a simulated electrostatic discharge (ESD, human body model) and its basic operation checked following the exposure (see Figure 7).

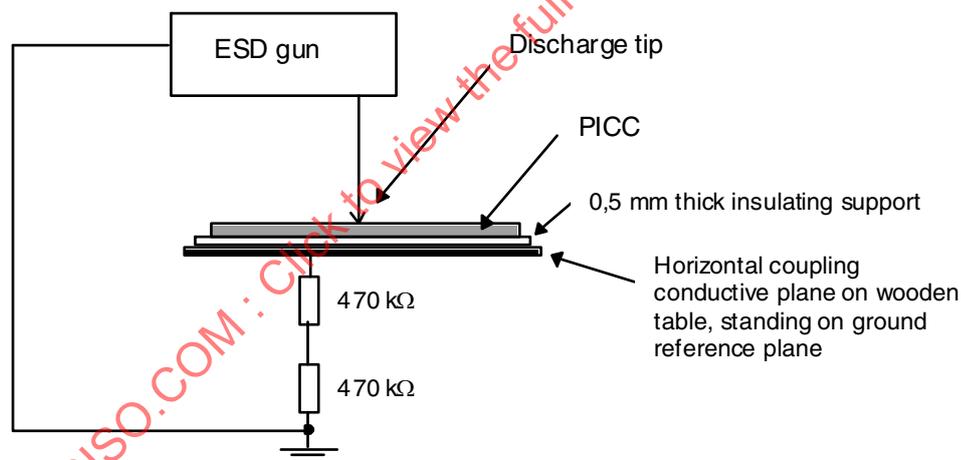


Figure 7 — ESD test circuit

6.2.2.1 Apparatus

Refer to IEC 61000-4-2:2008.

- a) Main specifications of the ESD generator:
 - energy storage capacitance: 150 pF \pm 10 %;
 - discharge resistance: 330 Ω \pm 10 %;
 - charging resistance: between 50 M Ω and 100 M Ω ;
 - rise time: 0,7 ns to 1 ns.

b) Selected specifications from the optional items:

- type of equipment: table top equipment;
- discharge method: direct application of air discharge to the equipment under test;
- discharge electrodes of the ESD generator: Round tip probe of 8 mm diameter.

6.2.2.2 Test procedure

Connect the ground pin of the apparatus to the conductive plate upon which the PICC is placed.

Apply the discharge successively in normal polarity to each of the 20 test zones shown in Figure 8. Then repeat the same procedure with reversed polarity. Allow a cool-down period between successive pulses of at least 10 s.

WARNING - If the PICC includes contacts, the contacts should face up and the zone which includes contacts should not be exposed to this discharge.

Check that the PICC operates as intended at the end of the test.

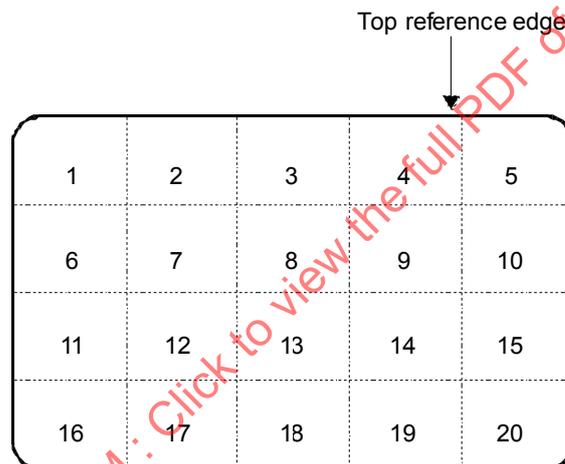


Figure 8 — Test zones on PICC for ESD test

6.2.2.3 Test report

The test report shall state whether or not the PICC operates as intended.

7 Test of ISO/IEC 14443-2 parameters

7.1 PCD tests

All the tests described below will be done in the operating volume as defined by the PCD manufacturer.

7.1.1 PCD field strength

7.1.1.1 Purpose

This test measures the field strength produced by a PCD in its operating volume.

NOTE The test takes account of PICC loading of the PCD.

7.1.1.2 Test procedures

Procedure for H_{\max} test:

- a) Tune the Reference PICC to 19 MHz as described in 5.4.3 steps a) to g).
- b) Calibrate the Test PCD assembly to produce the H_{\max} operating condition on the calibration coil.
- c) Place the Reference PICC into the DUT position on the Test PCD assembly. Switch the jumper J1 to position 'b' and adjust R2 to obtain a DC voltage of 3 V measured at connector CON3. Alternatively, jumper J1 may be set to position 'c' and the applied voltage on CON2 is adjusted to obtain a DC voltage of 3 V at connector CON3. In both cases, the operating field condition shall be verified by monitoring the voltage on the calibration coil and adjusted if necessary.

WARNING - R2 value should be between 75 and 85 Ω . Check this range at least once before using the alternative method.

- d) Position the Reference PICC within the defined operating volume of the PCD under test. The DC voltage at CON3 shall not exceed 3 V.

Procedure for H_{\min} test:

- a) Tune the Reference PICC to 13,56 MHz as described in 5.4.3.
- b) Calibrate the Test PCD assembly to produce the H_{\min} operating condition on the calibration coil.
- c) Place the Reference PICC into the DUT position on the Test PCD assembly. Switch the jumper J1 to position 'b' and adjust R2 to obtain a DC voltage of 3 V measured at connector CON3. Alternatively, jumper J1 may be set to position 'c' and the voltage on CON2 is adjusted to obtain a DC voltage of 3 V at connector CON3. In both cases, the operating field condition shall be verified by monitoring the voltage on the calibration coil and adjusted if necessary.

WARNING - R2 value should be between 400 and 550 Ω . Check this range at least once before using the alternative method.

- d) Position the Reference PICC within the defined operating volume of the PCD under test. The DC voltage at CON3 shall exceed 3 V.

7.1.1.3 Test report

The test report shall give the DC voltage measured at CON3 for R2 or variable load resistor adjusted to H_{\min} and H_{\max} field strength under the conditions applied.

7.1.2 PCD field strength supporting operation with "Class 1" PICCs

7.1.2.1 Purpose

The following additional PCD test is necessary for interoperability between PCDs and "Class 1" PICCs. This test is similar to PCD field strength test specified in 7.1.1 but uses a Reference PICC adjusted to apply a higher loading effect.

NOTE The loading effect of "Class 1" PICCs is lower than the loading effect of the Reference PICC used in this test (see "Class 1" PICC maximum loading effect test in 7.2.4). This guarantees that H_{\min} is supplied in the "Class 1" PCD operating volume if additionally the "Class 1" PICCs' antenna size and location are similar to the Reference PICC (Annex D) antenna size and location. For PICCs with different antenna size and/or location other classes may be created with, for each class, a corresponding Reference PICC.

7.1.2.2 Test procedure

The PCD field strength test defined in 7.1.1, procedure for H_{\min} test, shall be repeated with the Reference PICC calibrated to obtain 6 V at CON3 instead of 3 V.

NOTE The Reference PICC calibration used in this test is the same as the calibration defined in 7.1.1.

7.1.2.3 Test report

The test report shall give the DC voltage measured at CON3 for R2 or variable load resistor adjusted to H_{\min} field strength under the conditions applied.

NOTE The volume in which the DC voltage exceeds 6 V defines the "Class 1" operating volume.

7.1.3 Power transfer PCD to PICC

7.1.3.1 Purpose

This test is used to determine that the PCD is able to supply a certain power to a PICC placed anywhere within the defined operating volume.

7.1.3.2 Test procedure

- a) Tune the Reference PICC to 19 MHz as described in 5.4.3 steps a) to g) and leave jumper J1 to position 'a'.
- b) Position the Reference PICC within the defined operating volume of the PCD under test. The DC voltage at CON3 shall exceed 3 V.

7.1.3.3 Test report

The test report shall give the DC voltage at CON3 measured within the defined operating volume under the defined conditions.

7.1.4 Modulation index and waveform

7.1.4.1 Purpose

This test is used to determine the index of modulation of the PCD field as well as the rise and fall times and the overshoot values as defined in ISO/IEC 14443-2:2010.

7.1.4.2 Test procedure

- a) Position the calibration coil at an arbitrary position in the defined operating volume and display the induced coil voltage on a suitable oscilloscope. Determine the modulation index and waveform characteristics using the analysis tool defined in Annex E.
- b) Tune the Reference PICC to 16,5 MHz as described in 5.4.3 steps a) to g) and switch the jumper J1 to position 'c'.
- c) Place the Reference PICC at a particular position in the PCD operating volume.
- d) Apply and adjust a DC voltage at CON2 to obtain a DC voltage at connector CON3 of 3 V or optionally 6 V when supporting "Class 1" at that position.

NOTE 1 If a DC voltage of 6 V cannot be reached at the selected position, the maximum achievable voltage should be used for the test.

- e) If the unmodulated voltage on CON4, measured with a suitable oscilloscope (requirements see 5.1.1) is below 1 V (peak-to-peak), use an alternative pick up coil to determine the waveform characteristic.

NOTE 2 This alternative pick up coil should have a "figure of 8" shape with 15 mm radius positioned farthest away from the Reference PICC to minimize coupling and as close as possible to the PCD antenna to maximize induced voltage.

- f) Determine the modulation index and waveform characteristic from the voltage at CON4 or at the alternative pick up coil using the analysis tool defined in Annex E.

- g) Repeat steps c) to f) for various positions within the operating volume.

NOTE 3 The selected position of the calibration coil within the operating volume is not expected to affect the results.

NOTE 4 The Reference PICC load does not represent the worst case loading effect of a PICC. Higher loading effects may be achieved with resonance frequencies closer to carrier frequency (e.g. 15 MHz or 13,56 MHz).

7.1.4.3 Test report

The test report shall give the measured modulation index of the PCD field, the rise and fall times and overshoot values, within the defined operating volume in unloaded and loaded conditions.

7.1.5 Load modulation reception

7.1.5.1 Purpose

This test is used to verify that a PCD correctly detects the load modulation of a PICC which conforms to ISO/IEC 14443-2:2010.

7.1.5.2 Test procedure

The Reference PICC and its calibration procedure allow the sensitivity of a PCD to load modulation to be assessed. The Reference PICC does not emulate the loading effect of all types of PICC.

- a) Tune the Reference PICC to 13,56 MHz as described in 5.4.3 and switch the jumper J1 to position 'c'.
- b) Place the Reference PICC at a particular position in the PCD operating volume.
- c) Apply and adjust a DC voltage at CON2 to obtain a DC voltage at connector CON3 of 3 V or optionally 6 V when supporting "Class 1" at that position.
- d) Increase the modulation signal amplitude at CON1 to produce responses until the PCD detects at least 10 of them consecutively.
- e) Place the Reference PICC in the DUT position on the Test PCD assembly.
- f) Adjust the Test PCD assembly to produce a field strength H which gives the same voltage at CON3 and note the corresponding field strength by reading the calibration coil voltage.
- g) Measure the Reference PICC load modulation amplitude V_{LMA} as described in 7.2.1 and compare it with the standard limit associated with the noted field strength. This measured V_{LMA} level defines the PCD sensitivity criterion in order to compare with the standard limit to perform these test measurements.
- h) Repeat steps b) to g) for various positions within the operating volume.
- i) Repeat steps a) to h) with Reference PICC resonance frequency 15 MHz.

Any position in which the PCD sensitivity is above the standard limit shall be considered out of the operating volume.

NOTE 1 The test coverage may be expanded by using additional resonance frequencies below 13,56 MHz such as 12 MHz and 10 MHz.

NOTE 2 The PCD sensitivity should be below the standard limit to ensure good reception of the PICC load modulation.

NOTE 3 This test does not check that the PCD reception is independent of the phase of the PICC load modulation. Consequently, it cannot guarantee the correct reception of any PICC compliant with ISO/IEC 14443-2:2010.

7.1.5.3 Test report

The test report shall give the PCD load modulation sensitivity for the tested positions.

7.2 PICC tests

7.2.1 PICC transmission

7.2.1.1 Purpose

The purpose of this test is to determine the load modulation amplitude V_{LMA} of the PICC within the operating field range $[H_{min}, H_{max}]$ as specified in ISO/IEC 14443-2:2010. Also the functionality of the PICC for Type A and Type B within their corresponding modulation ranges as defined in ISO/IEC 14443-2:2010 shall be determined.

NOTE No load modulation test is required for bit rates of $fc/64$, $fc/32$ and $fc/16$.

7.2.1.2 Test procedure

Step 1: The load modulation test circuit of Figure 2 and the Test PCD assembly of Figure 3 are used.

Adjust the RF power delivered by the signal generator to the test PCD antenna to the required field strength as measured by the calibration coil. Connect the output of the load modulation test circuit of Figure 2 to a digital sampling oscilloscope. The 10 Ω potentiometer P1 shall be trimmed to minimize the residual carrier. This signal shall be at least 40 dB lower than the signal obtained by shorting one sense coil.

WARNING - The PICC load modulation amplitude test should be done by increasing the field strength from 0 A/m (rms), thus checking correct PICC operation starting from H_{min} .

Step 2: The PICC under test shall be placed in the DUT position, concentric with sense coil a. The RF drive into the test PCD antenna shall be re-adjusted to the required field strength.

A REQA or a REQB command sequence as defined in ISO/IEC 14443-3:— shall be sent by the Test PCD to obtain a signal or load modulation response from the PICC.

Display a segment of at least six cycles of the waveform of the subcarrier load modulation on the digital sampling oscilloscope and store the sampled data in a file for analysis by a computer software program (see Annex F).

NOTE 1 Care should be taken to apply a proper synchronization method for low amplitude load modulation.

Fourier transform with a Bartlett window exactly six subcarrier cycles of the sampled modulation waveform using suitable computer software (as the one given in Annex F). Use a discrete Fourier transformation with a scaling such that a pure sinusoidal signal results in its peak magnitude. In order to minimize transient effects, avoid analyzing a subcarrier cycle immediately following a non-modulating period or a phase shift of the subcarrier. The discrete Fourier transformation shall be done at the sidebands frequencies generated by the PICC under test, i.e. $fc + fs$ and $fc - fs$.

The resulting peak amplitudes of the upper and lower sidebands at $fc + fs$ and $fc - fs$ shall be above the value defined in ISO/IEC 14443-2:2010, 8.2.2.

NOTE 2 For Type B PICC load modulation test, the oscilloscope FFT (Fast Fourier Transform) option may also be used on a large number of subcarrier cycles with neither transient effect nor phase shift (i.e. on a stable part of synchronization time TR1 as defined in ISO/IEC 14443-2:2010, 9.2.5 or on a stable part of SOF as defined in ISO/IEC 14443-3:—, 7.1.4).

7.2.1.3 Test report

The test report shall give the measured peak amplitudes of the upper and lower sidebands at $f_c + f_s$ and $f_c - f_s$ and the applied fields and modulations.

7.2.2 PICC reception

7.2.2.1 Purpose

The purpose of this test is to verify the ability of the PICC to receive the PCD commands.

7.2.2.2 PICC Type A

7.2.2.2.1 Test conditions for a bit rate of $fc/128$

Three test conditions are defined with timings at the border of the Type A PICC modulation waveform timing parameters zone defined in ISO/IEC 14443-2:2010, Figure 4:

- Condition 1: maximum $t_1 - t_2$ and maximum t_3 , no overshoot;
- Condition 2: minimum achievable (with Test PCD assembly) $t_1 - t_2$ and maximum associated t_3 , maximum overshoot;
- Condition 3: minimum achievable (with Test PCD assembly) t_3 and maximum associated $t_1 - t_2$, maximum overshoot.

NOTE The carrier amplitude at the end of t_2 should be less than 4 %.

These 3 tests conditions shall be tested at least using H_{\min} and H_{\max} whereas parameter t_1 is the maximum specified value when condition 1 and condition 3 are used and is the minimum specified value when condition 2 is used.

7.2.2.2.2 Test conditions for bit rates of $fc/64$, $fc/32$ and $fc/16$

Three test conditions are defined with timings at the border of the Type A PICC modulation waveform timing parameters zone defined in ISO/IEC 14443-2:2010, Figures 7, 8 and 9:

- Condition 1: maximum $t_1 - t_5$ and maximum t_6 , no overshoot;
- Condition 2: minimum achievable (with Test PCD assembly) $t_1 - t_5$ and maximum associated t_6 , maximum overshoot;
- Condition 3: minimum achievable (with Test PCD assembly) t_6 and maximum associated $t_1 - t_5$, maximum overshoot.

These 3 tests conditions shall be tested at least using H_{\min} and H_{\max} whereas:

- parameter a is the maximum specified value when condition 1 is used and is the minimum achievable value for the Test PCD assembly when condition 2 and condition 3 are used;
- parameter t_1 is the maximum specified value when condition 1 and condition 3 are used and is the minimum specified value when condition 2 is used.

7.2.2.2.3 Test procedure

Under the conditions defined in 7.2.2.2.1 the PICC shall answer to a REQA with ATQA.

A PICC supporting the optional $fc/64$ bit rate shall operate under the conditions defined in 7.2.2.2.2 after selection of a bit rate of $fc/64$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/64$.

A PICC supporting the optional $fc/32$ bit rate shall operate under the conditions defined in 7.2.2.2.2 after selection of a bit rate of $fc/32$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/32$.

A PICC supporting the optional $fc/16$ bit rate shall operate under the conditions defined in 7.2.2.2.2 after selection of a bit rate of $fc/16$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/16$.

7.2.2.3 PICC Type B

7.2.2.3.1 Test conditions

Three test conditions are defined with timings at the border of the Type B PICC modulation waveform timing parameters zone defined in ISO/IEC 14443-2:2010, Figures 14, 15, 16 and 17:

- Condition 1: maximum t_f and maximum t_r , no undershoot and overshoot;
- Condition 2: minimum achievable (with Test PCD assembly) t_f and maximum associated t_r , maximum undershoot and overshoot;
- Condition 3: minimum achievable (with Test PCD assembly) t_r and maximum associated t_f , maximum undershoot and overshoot.

These 3 tests conditions shall be tested at least using:

- H_{\min} and H_{\max} ;
- minimum and maximum modulation index m for the associated field strength applied (see ISO/IEC 14443-2:2010, Figure 13).

7.2.2.3.2 Test procedure

Under the conditions defined in 7.2.2.3.1 the PICC operating at a bit rate of $fc/128$ shall answer to a REQB with ATQB.

A PICC supporting the optional $fc/64$ bit rate shall operate under the conditions defined in 7.2.2.3.1 after selection of a bit rate of $fc/64$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/64$.

A PICC supporting the optional $fc/32$ bit rate shall operate under the conditions defined in 7.2.2.3.1 after selection of a bit rate of $fc/32$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/32$.

A PICC supporting the optional $fc/16$ bit rate shall operate under the conditions defined in 7.2.2.3.1 after selection of a bit rate of $fc/16$. This PICC shall respond correctly to an I-block transmitted at a bit rate of $fc/16$.

7.2.2.4 Test report

The test report shall confirm the intended operation at the mandatory $fc/128$ bit rate. For PICCs supporting one or more of the optional high bit rates the test report shall confirm the intended operation at the supported bit rates. Used test conditions shall be mentioned in the test report.

7.2.3 PICC resonance frequency (informative)

7.2.3.1 Purpose

The test may be used to measure the resonance frequency of a PICC.

When two or more PICCs are placed in the same PCD energizing field, the resonance frequency of each PICC decreases.

Care should be taken in designing each PICC resonance frequency.

WARNING - The resonance frequency may depend on the field strength used during the measurement.

7.2.3.2 Procedure

The resonance frequency of a PICC is measured by using an impedance analyzer or a network analyzer or a LCR-meter connected to a calibration coil. The PICC should be placed on the calibration coil at a distance of 10 mm, with the axes of the two coils being congruent. The resonance frequency is that frequency at which the resistive part of the measured complex impedance is at maximum.

7.2.3.3 Test report

When applied the test report shall give the PICC resonance frequency and the measurement conditions.

7.2.4 "Class 1" PICC maximum loading effect

7.2.4.1 Purpose

The following additional PICC test is necessary for interoperability between PCDs and "Class 1" PICCs defined in ISO/IEC 14443-1:2008.

NOTE This test improves interoperability only if the "Class 1" PICCs antenna size and location are similar to the Reference PICC (Annex D) antenna size and location. For PICCs with different antenna size and/or location other classes may be created with, for each class, a corresponding reference PICC.

7.2.4.2 Test procedure

The PICC loading effect at H_{\min} shall be measured using the test PCD assembly. It shall not exceed the loading effect of the Reference PICC tuned to 13,56 MHz and calibrated to obtain 6 V at CON3 at H_{\min} . The procedure of this substitution method is as follows.

- a) Tune the Reference PICC to 13,56 MHz as described in 5.4.3.
- b) Calibrate the Test PCD assembly to produce the H_{\min} operating condition on the calibration coil.
- c) Place the Reference PICC into the DUT position on the Test PCD assembly. Switch the jumper J1 to position 'b' and adjust R2 to obtain a DC voltage of 6 V measured at connector CON3. Alternatively, jumper J1 may be set to position 'c' and the applied voltage on CON2 is adjusted to obtain a DC voltage of 6 V at connector CON3. In both cases, the operating field condition shall be verified by monitoring the voltage on the calibration coil and adjusted if necessary.

WARNING - R2 value should be between 900 and 1000 Ω . Check this range at least once before using the alternative method.

- d) Remove the Reference PICC.
- e) Place the PICC under test into the DUT position on the Test PCD assembly.

f) Measure the field strength H_C monitored by the calibration coil.

The field strength H_C shall be greater than H_{\min} .

7.2.4.3 Test report

The test report shall give the value H_C .

8 Test of ISO/IEC 14443-3 and ISO/IEC 14443-4 parameters

8.1 PCD tests

See Annex H and Annex I.

8.2 PICC tests

See Annex G.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

Annex A (normative)

Test PCD Antenna

A.1 Test PCD Antenna layout including impedance matching network

Figures A.1, A.2, A.3 and A.4 illustrate Test PCD antenna layout. Drawings are not to scale.

The antenna coil track width is 1,8 mm (except for through-plated holes).

Starting from the impedance matching network there are crossovers every 45 °.

Printed circuit board (PCB): FR4 material, thickness 1,6 mm, double sided with 35 µm copper.

NOTE 1 The layout of the impedance matching network is informative.

NOTE 2 Such printed circuit boards and R_{ext} resistors are available from various commercial sources.

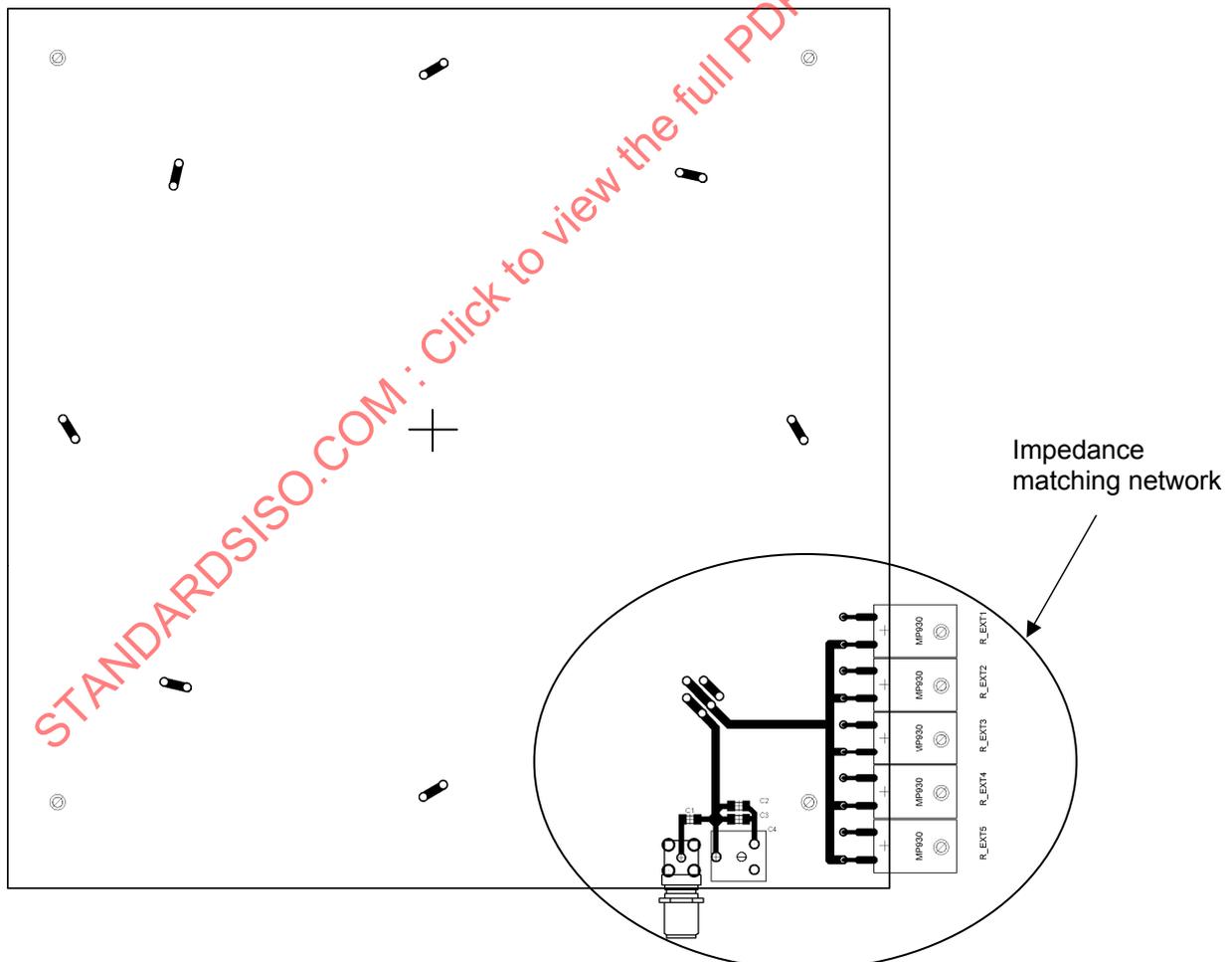


Figure A.1 — Test PCD antenna layout including impedance matching network for a bit rate of $fc/128$
(View from front)

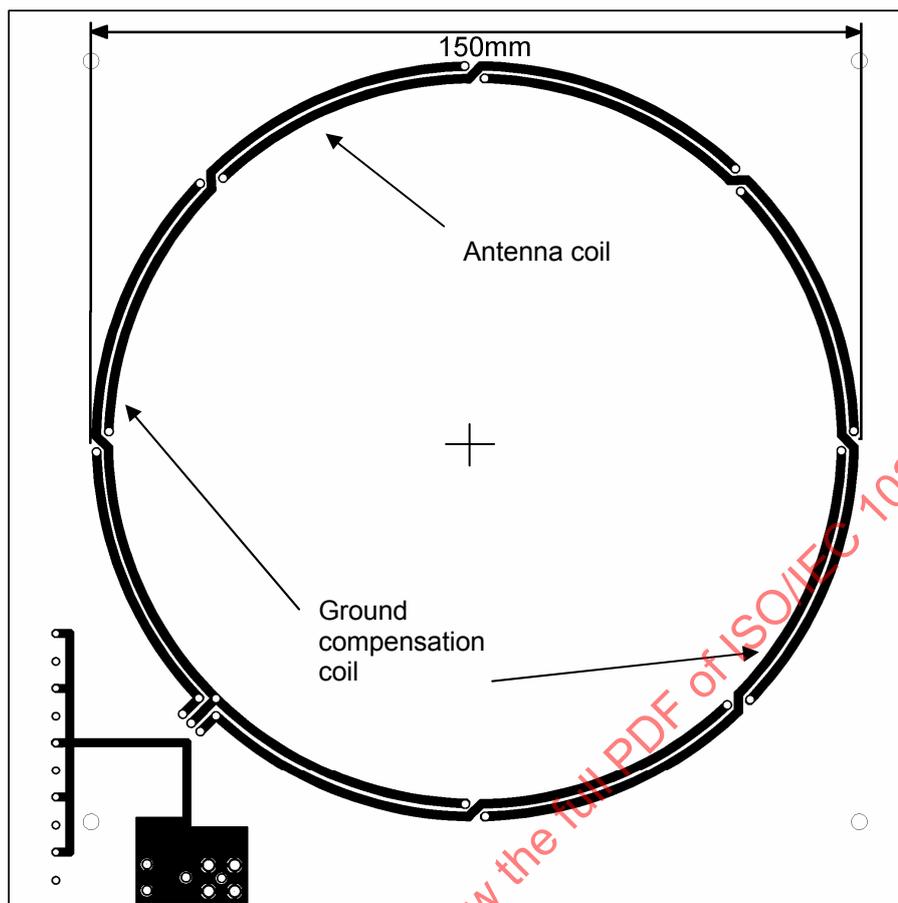


Figure A.2 — Test PCD antenna layout including impedance matching network for a bit rate of $f_c/128$
(View from back)

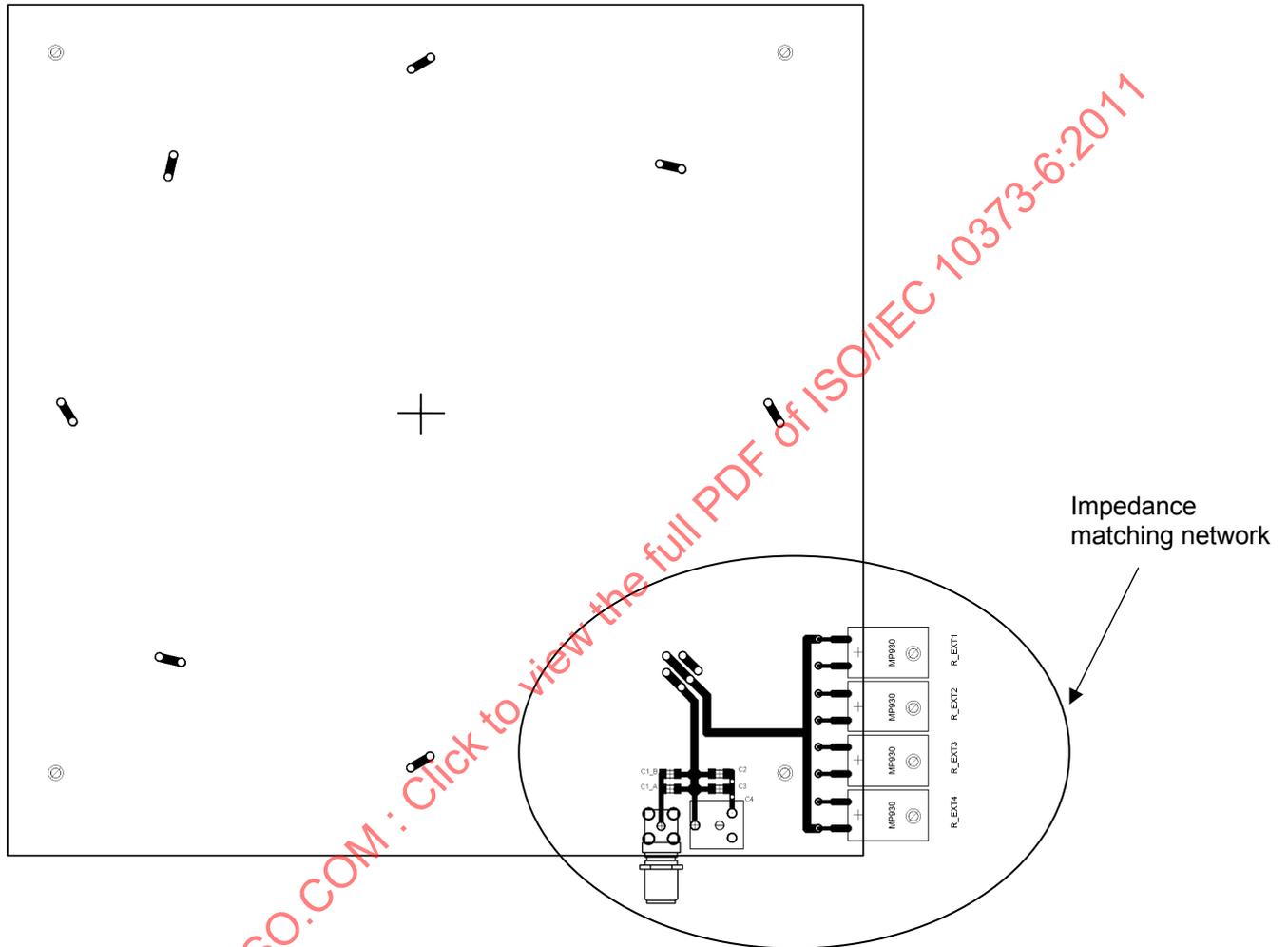


Figure A.3 — Test PCD antenna layout including impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$ (View from front)

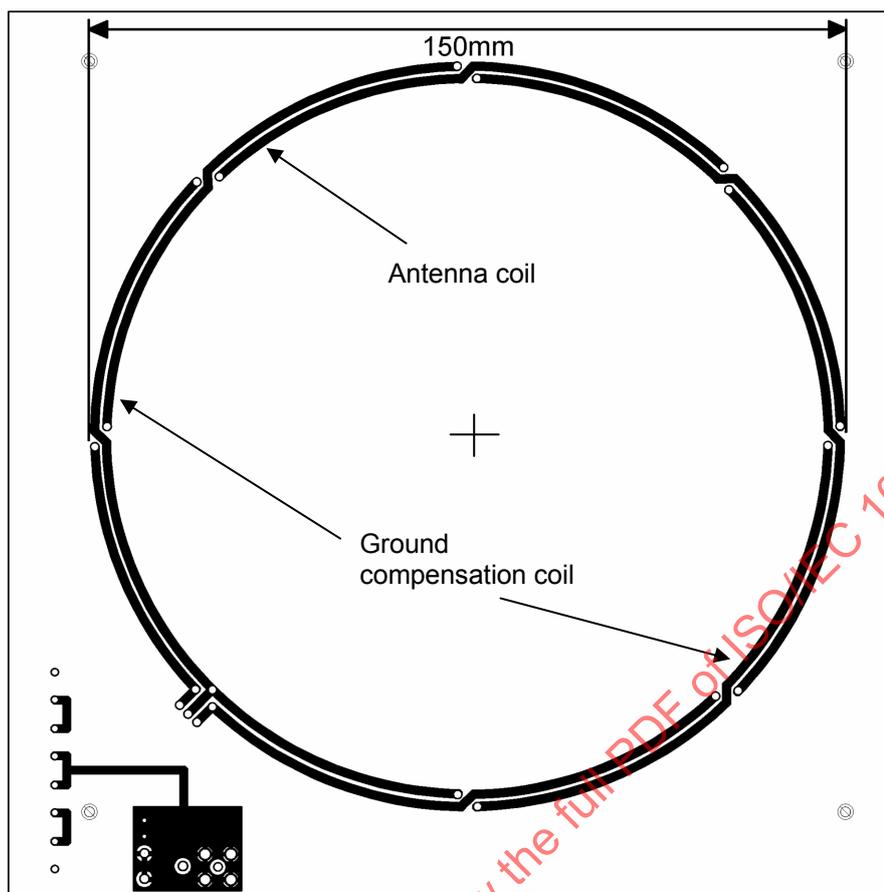


Figure A.4 — Test PCD antenna layout including impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$ (View from back)

A.2 Impedance matching network

The antenna impedance (R_{ant} , L_{ant}) is adapted to the signal generator output impedance ($Z = 50 \Omega$) by a matching circuit (see A.2.1 and A.2.2). The capacitors C1, C1a, C1b, C2 and C3 have fixed values. The input impedance phase can be adjusted with the variable capacitor C4.

The test PCD assembly as defined in 5.3 and in this Annex is intended to be used for time limited measurements, to avoid any overheating of the individual components. If the test is run continuously, heat dissipation shall be improved. Care shall be taken to keep maximum voltages and maximum heat dissipation within the specified limits of the individual components.

Two impedance matching networks are described: the impedance matching network for a bit rate of $fc/128$ and the impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$. Table A.1 contains their use cases.

Table A.1 — Impedance matching network use cases

Impedance matching network	Magnetic field capability	Bit rate capability
For a bit rate of $fc/128$	Up to 12 A/m (rms)	Only $fc/128$
For bit rates of $fc/64$, $fc/32$ and $fc/16$	Up to 7,5 A/m (rms)	From $fc/128$ to $fc/16$

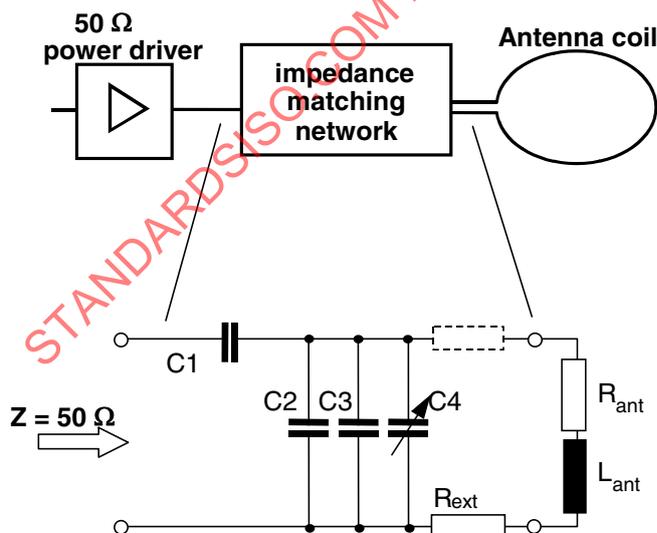
NOTE 1 The tolerance of the matched antenna impedance is $\pm 5 \Omega$ and $\pm 10^\circ$.

NOTE 2 R_{ext} is placed on the ground side of the antenna coil.

NOTE 3 The power and voltage ranges include a safety margin.

NOTE 4 The linear low distortion variable output 50Ω power driver should be capable of emitting Type A and Type B modulations for transmission of REQA and REQB as defined in 7.2.2. The output power should be adjustable to deliver H fields in the range of 1,5 A/m (rms) – 12 A/m (rms). Care should be taken with the duration of fields above the upper operating range of 7,5 A/m (rms).

A.2.1 Impedance matching network for a bit rate of $fc/128$



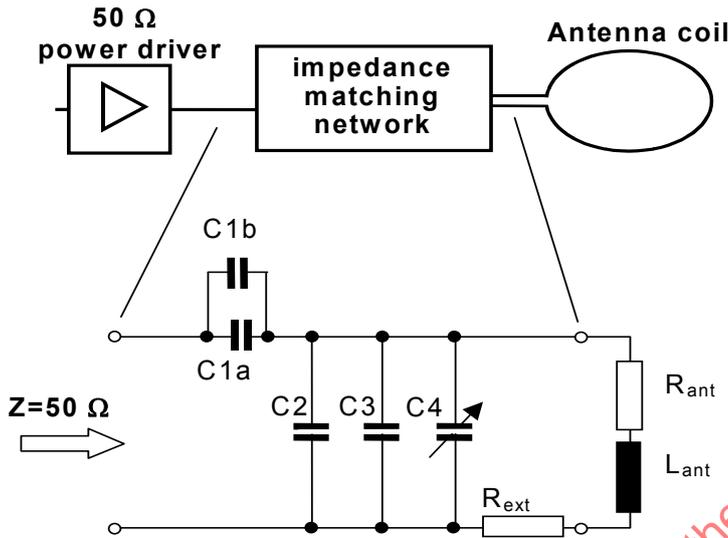
Component Table:

	Value	Unit	Remarks
C1	47	pF	Voltage range 200 V
C2	180	pF	Voltage range 200 V
C3	22	pF	Voltage range 200 V
C4	2-27	pF	Voltage range 200 V
R_{ext}	0,94	Ω	Power range 10 W

Figure A.5 — Impedance matching network for a bit rate of $fc/128$

- NOTE 1 R_{ext} may be built by connecting five resistors of $4,7 \Omega$ 2 W in parallel.
- NOTE 2 R_{ext} may still be placed at the position marked with a dashed outline, as shown in Figure A.5.
- NOTE 3 R_{ext} may be 4 W if the maximum field is up to 7,5 A/m (rms).
- NOTE 4 The parasitic capacitance of the antenna is not shown in Figure A.5.

A.2.2 Impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$



Component Table:

	Value	Unit	Remarks
C1a	82	pF	Voltage range 200 V
C1b	8,2	pF	Voltage range 200 V
C2	150	pF	Voltage range 200 V
C3	10	pF	Voltage range 200 V
C4	2-27	pF	Voltage range 200 V
R_{ext}	4,7	Ω	Power range 20 W

Figure A.6 — Impedance matching network for bit rates of $fc/64$, $fc/32$ and $fc/16$

- NOTE 1 R_{ext} may be built by a parallel circuit of each two resistors of $4,7 \Omega$ 5 W in series.
- NOTE 2 R_{ext} should be placed on the GND side of the antenna as drawn.
- NOTE 3 The parasitic capacitance of the antenna is not shown in Figure A.6.

Annex B (informative)

Test PCD Antenna tuning

Figures B.1 and B.2 show the two steps of a simple phase tuning procedure to match the impedance of the antenna to that of the driving generator. After the two steps of the tuning procedure the signal generator shall be directly connected to the antenna output for the tests.

Step 1:

A high precision resistor of $50 \Omega \pm 1 \%$ (e.g. 50Ω BNC resistor) is inserted in the signal line between the signal generator output and an antenna connector. The two probes of the oscilloscope are connected to both sides of the serial reference resistor. The oscilloscope displays a Lissajous figure when it is set in Y to X presentation. The signal generator is set to:

- Wave form: Sinusoidal;
- Frequency: 13,56 MHz;
- Amplitude: 2 V (rms) - 5 V (rms).

The output is terminated with a second high precision resistor of $50 \Omega \pm 1 \%$ (e.g. 50Ω BNC terminating resistor). The probe, which is in parallel to the output connector has a small parasitic capacitance C_{probe} . A calibration capacitance C_{cal} in parallel to the reference resistor compensates this probe capacitor if $C_{\text{cal}} = C_{\text{probe}}$. The probe capacitor is compensated when the Lissajous figure is completely closed.

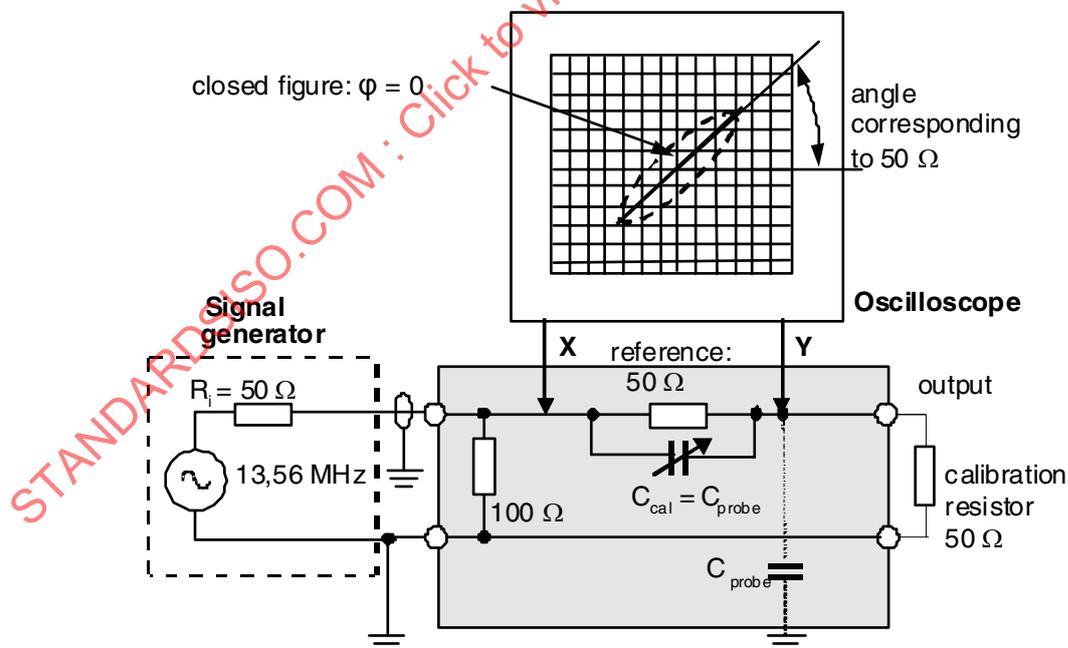


Figure B.1 — Calibration set-up (Step 1)

NOTE The ground cable has to be run close to the probe to avoid induced voltages caused by the magnetic field.

Step 2:

Using the same values as set for step 1, in the second step the matching circuitry is connected to the antenna output. The capacitor C4 on the antenna board is used to tune the phase to zero.

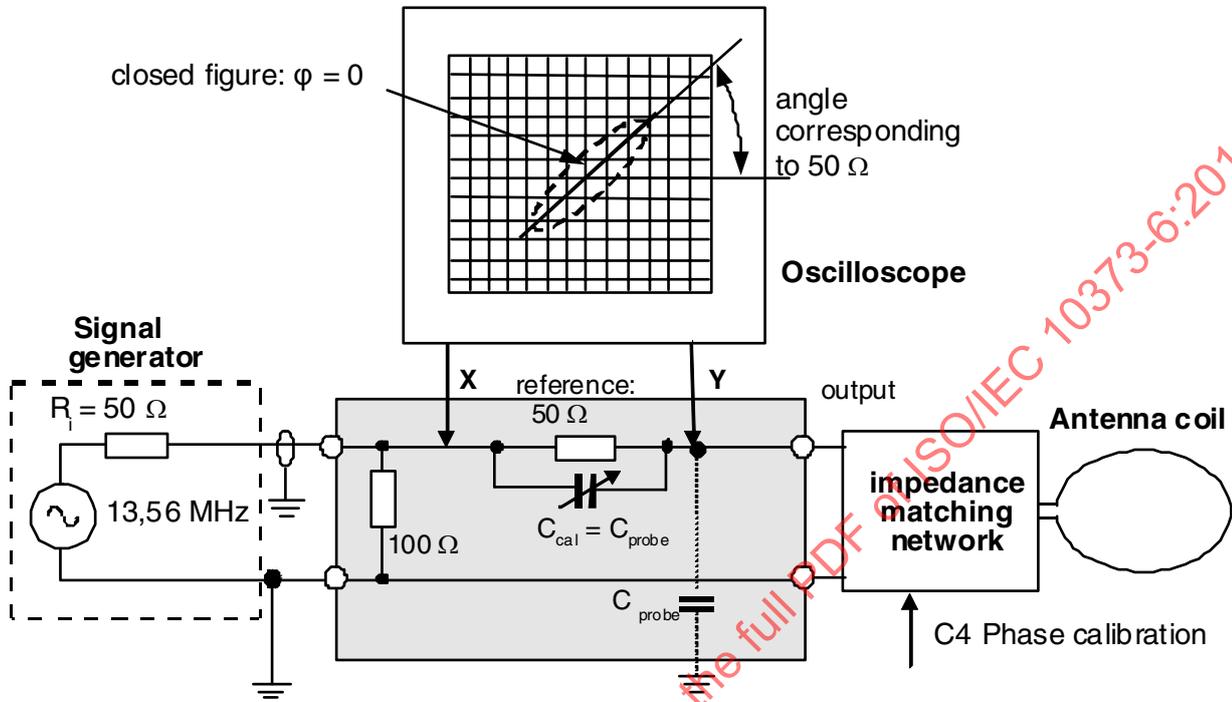


Figure B.2 — Calibration set-up (Step 2)

Annex C (normative)

Sense coil

C.1 Sense coil layout

Figure C.1 illustrates sense coil layout.

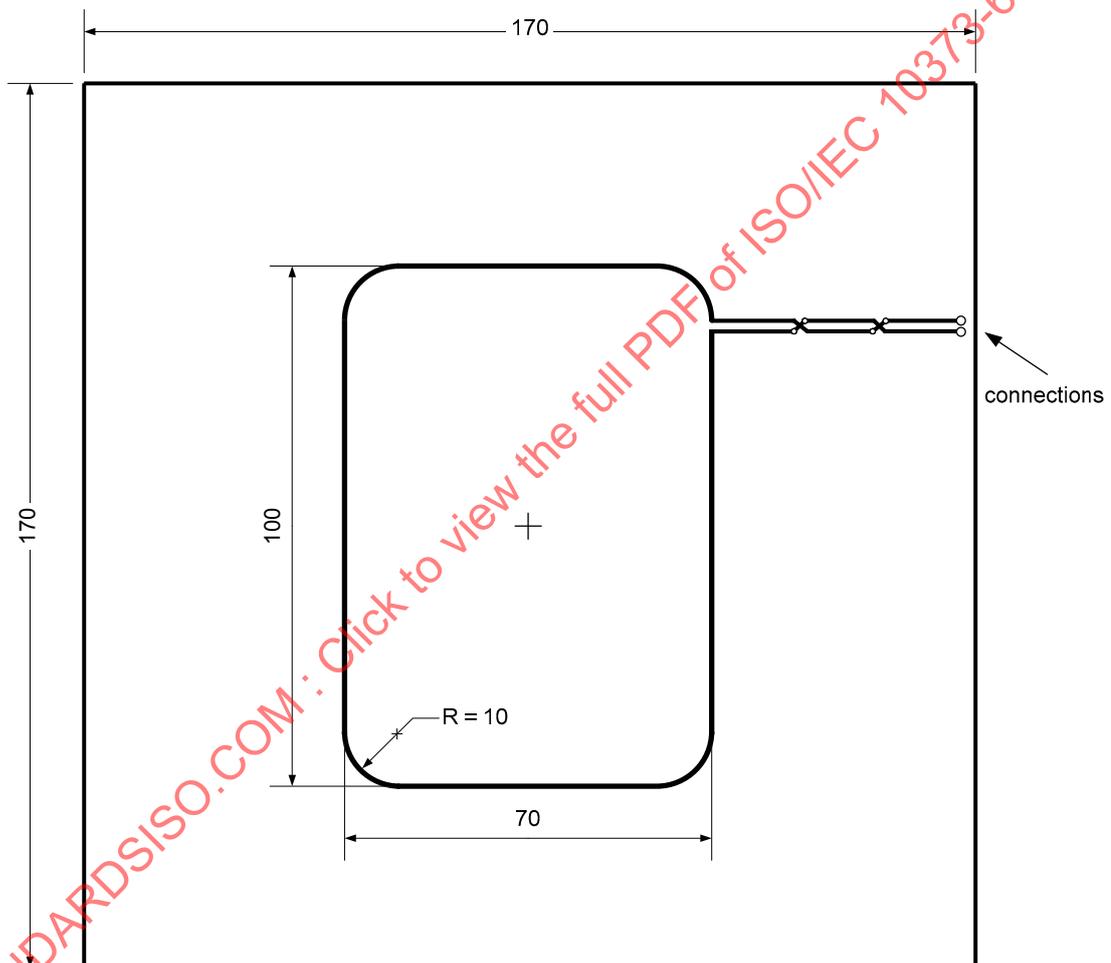


Figure C.1 — Layout for sense coils a and b

Dimensions in millimeters (drawings are not to scale).

The sense coil track width is 0,5 mm with relative tolerance $\pm 20\%$ (except for through-plated holes). Size of the coils refers to the outer dimensions.

Printed circuit board (PCB): FR4 material, thickness 1,6 mm, double sided with 35 μm copper.

NOTE Such printed circuit boards are available from various commercial sources.

C.2 Sense coil assembly

Figure C.2 illustrates sense coil assembly.

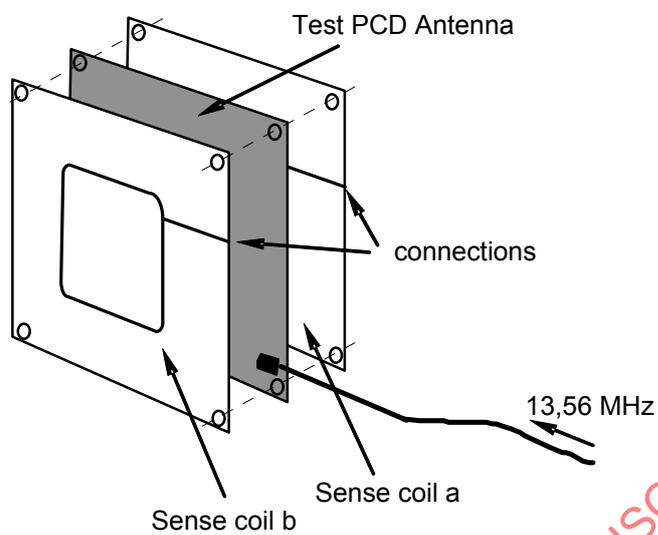


Figure C.2 — Sense coil assembly

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

Annex D (normative)

Reference PICC antenna layout

Figure D.1 illustrates Reference PICC antenna layout.

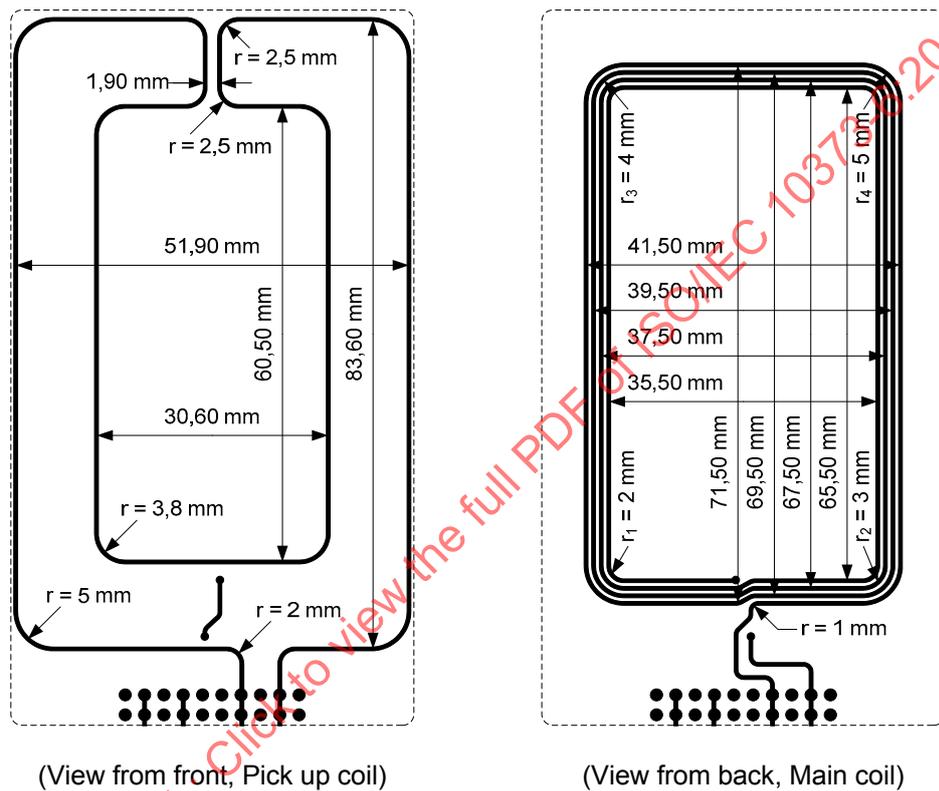


Figure D.1 — Antenna layout

Dimensions to track center (drawings are not to scale).

The Pick up coil and the Main coil shall be concentric.

The two coils track width and spacing shall be 0,5 mm with a relative tolerance of $\pm 20\%$.

Printed circuit board (PCB): FR4 material, thickness 0,76 mm with a relative tolerance of $\pm 10\%$, double sided with 35 μm copper.

NOTE 1 At 13,56 MHz the inductance of the Main coil L1 is 2,3 $\mu\text{H} \pm 10\%$ and the resistance is 1,8 $\Omega \pm 10\%$.

NOTE 2 At 13,56 MHz the inductance of the Pick up coil L2 is 375 nH $\pm 10\%$ and the approximate resistance is 0,65 $\Omega \pm 10\%$.

NOTE 3 Such printed circuit boards are available from various commercial sources.

Annex E
(normative)

Modulation index and waveform analysis tool

E.1 Overview

The working principle of the modulation index and waveform analysis tool is illustrated in Figure E.1.

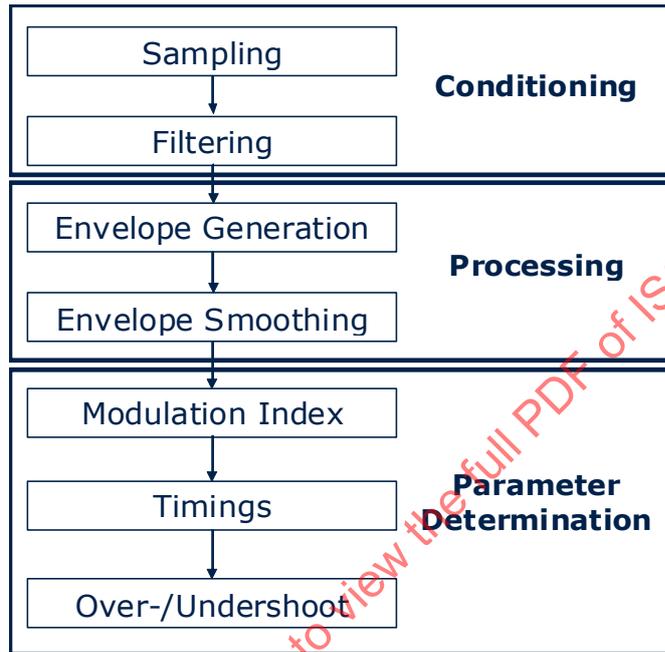


Figure E.1 — Modulation index and waveform analysis tool block diagram

Each block is described in following clauses.

E.2 Sampling

The oscilloscope used for signal capturing shall fulfill the requirements defined in 5.1.1. The time and voltage data of one modulation pulse (see Figure E.2) shall be transferred to a suitable computer.

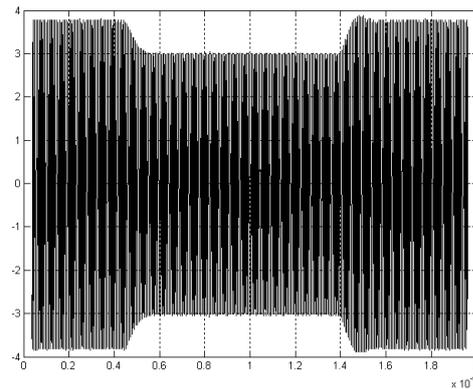


Figure E.2 — Modulation pulse

E.3 Filtering

A 4th order, Butterworth type band pass filter with center frequency of 13,56 MHz and 10 MHz 3-dB bandwidth shall be used for filtering the DC and higher harmonic components. The filter characteristic is illustrated in Figure E.3.

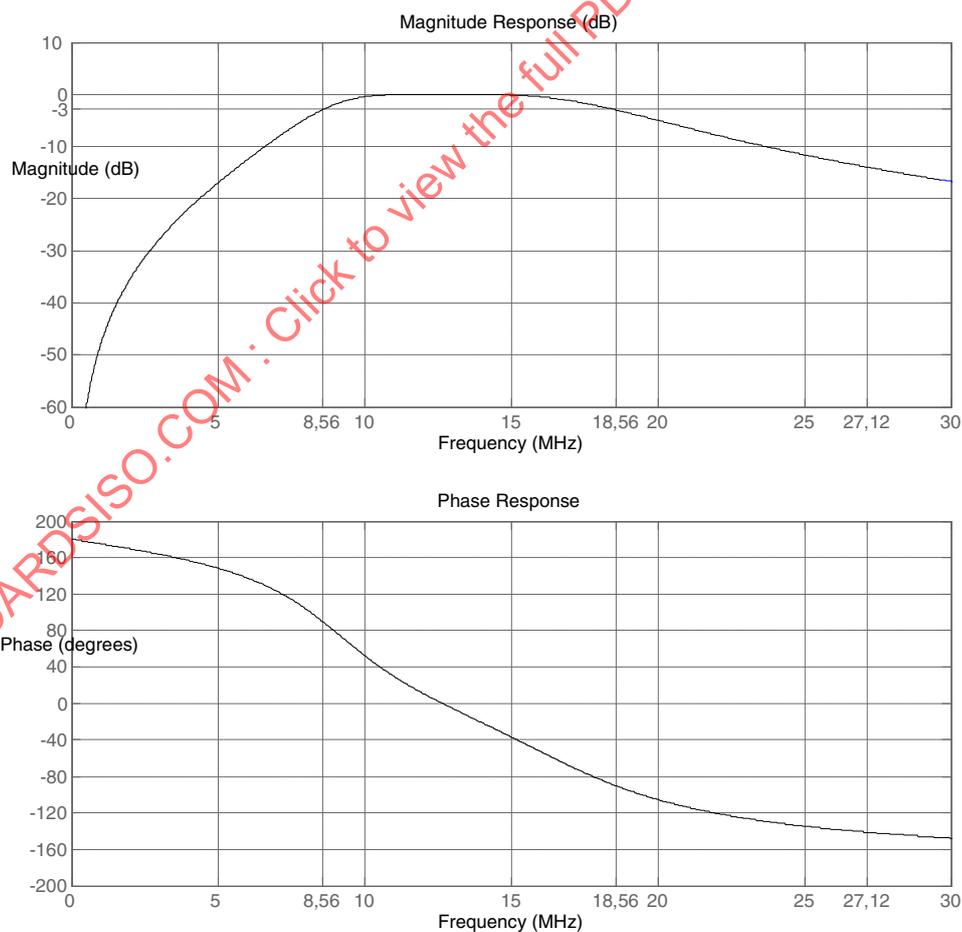


Figure E.3 — Filter characteristics

E.4 Envelope generation

The filtered signal shall be Hilbert transformed and the magnitude of this complex transform represents the signal envelope.

E.5 Envelope smoothing

The signal envelope shall be smoothed with a moving average filter and the filter period shall be one carrier period. The smoothed envelope signal is illustrated Figure E.4.

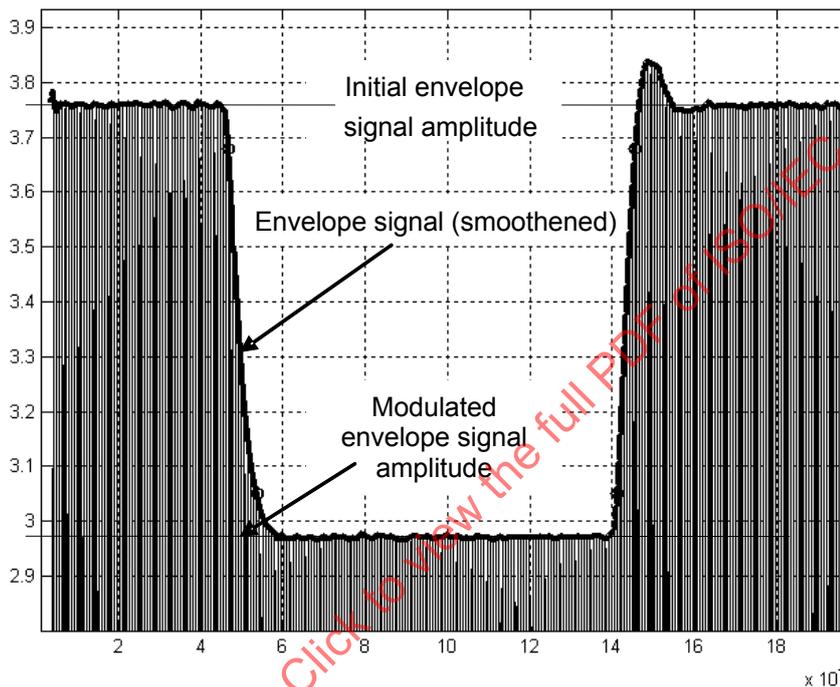


Figure E.4 — Envelope smoothing

E.6 Modulation index determination

The initial and modulated envelope signal amplitudes shall be determined by calculating the histogram of the smoothed envelope signal. The most frequent values correspond to the initial and modulated envelope signal amplitude. For Type A modulation signals only the initial envelope signal amplitude shall be determined using the histogram approach.

E.7 Timing determination

The rise and fall times should be determined according to the definitions in ISO/IEC 14443-2:2010.

E.8 Overshoot and undershoot determination

The already smoothed envelope signal shall be further smoothed by a moving average filter over 3 carrier periods before determining the overshoot and undershoot values according to the definitions in ISO/IEC 14443-2:2010.

E.9 Program of the modulation index and waveform analysis tool (informative)

The following program written in C language gives an example for the implementation of the modulation index and waveform analysis tool.

The C implementation consists of 6 different files which should be placed in the same folder.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

E.9.1 structures.h

```

/*****
*/structures.h
*/This code contains the structures to save important results
/*****
#endif STRUCTURES_H
#define STRUCTURES_H

typedef struct esl
{
    double volt;
    double time;
    struct esl *sig;
}ESL;

typedef struct times
{
    double tf; // Type B
    double tr; // Type B
    double b; // Type B
    double trstartind; // Type B
    double trendind; // Type B
    double tfstartind; // Type B
    double tfendind; // Type B

    double t1; // Type A (all bit rates)
    double t1startind; // Type A (all bit rates)
    double t1start; // Type A (all bit rates)
    double t1endind; // Type A (all bit rates)

    double t2; // Type A
    double t2startind; // Type A
    double t2start; // Type A
    double t3; // Type A
    double t3end; // Type A
    double t3endind; // Type A
    double t4; // Type A
    double t4endind; // Type A

    double t5; // Type A (higher bit rates)
    double t5startind; // Type A (higher bit rates)
    double t6; // Type A (higher bit rates)
    double t6end; // Type A (higher bit rates)
    double t6endind; // Type A (higher bit rates)
    double a; // Type A (higher bit rates)
    double tploone; //Type A (higher bit rates)
}TIMES;

typedef struct shootreader
{
    double shootind;
    double shootind_b;
    double hf_reader;
    double hr_reader;
    double above;
    double above_b;
}SHOOTREADER;

#endif

```

E.9.2 fftrm.h

```
/******  
/*fftrm.h  
/*This is the header file for fftrm.c  
/******  
  
#ifndef FFTRM_H  
#define FFTRM_H  
  
#define RE(z) ((z).r)  
#define IM(z) ((z).i)  
  
typedef float real;  
typedef double doublereal;  
typedef struct { real r, i; } complex;  
typedef struct { doublereal r, i; } doublecomplex;  
  
int zffts (int debug, doublecomplex *X, int M);  
int ziffts (int debug, doublecomplex *X, int M);  
void zfftrmc(doublecomplex *X, int M, int P, float D);  
void rmpo (int *rv, int *rvp );  
  
#endif
```

E.9.3 fftrm.c

```

/*****
/*fftrm.c
/*This code contains the necessary function for Fourier and inverse Fourier
/* transformation
/*****

#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include "fftrm.h"

#ifndef M_PI
#define M_PI 3.1415926535897932384626433832795
#endif

float *WR;
float *WI;

double real *DWR;
double real *DWI;

void rmpo( int *rv, int *rvp )
{
    int value_h;
    int n;

    n = 1;
    *rvp = -1;
    value_h = 1;

    while ( value_h > 0 )
    {
        value_h = *rv - n;
        (*rvp)++;
        n += n;
    }
}

void zfftrmc( double complex *X, int M, int P, float D )
{
    int MV2, MM1, J, I, K, L, LE, LE1, IP, IQ, IND, IND1, R;
    int I1, J1;
    float A, B;
    float WCOS, WSIN;
    float VR, VI;
    float ARG;

    static int IPOTC;
    static float DALT;

    DWR = (double real *)calloc(M, sizeof(double real));
    DWI = (double real *)calloc(M, sizeof(double real));

    /* if (IPOTC == P & D == DALT) goto warmstart; */

    IPOTC = P;
    DALT = (float)D;
    LE = 1;

```

```

IND = 0;

for (L=1;L<=P;L++)
{
    LE1 = LE;
    LE = LE*2;
    DWR[IND] = 1.0;
    DWI[IND] = 0.0;
    ARG = (float)M_PI/(float)LE1;
    WCOS = (float)cos(ARG);
    WSIN = (float)(D*sin(ARG));

    for (R=1;R<=LE1;R++)
    {
        IND1 = IND+1;
        A = (float)DWR[IND];
        B = (float)DWI[IND];
        DWR[IND1] = A*WCOS - B*WSIN;
        DWI[IND1] = B*WCOS + A*WSIN;
        ++IND;
    }
}

/* warmstart: */

MV2=M/2;
MM1=M-1;
J=1;

for (I=1; I<=MM1; I++)
{
    if (I >= J)
        goto P1;

    J1 = J-1;
    I1 = I-1;

    VR = (float)RE(X[J1]);
    VI = (float)IM(X[J1]);

    RE(X[J1]) = RE(X[I1]);
    IM(X[J1]) = IM(X[I1]);

    RE(X[I1]) = VR;
    IM(X[I1]) = VI;

P1: K = MV2;
P2: if (K >= J) goto P3;
    J = J-K;
    K = K/2;
    goto P2;
P3: J = J+K;
}
IND = 0;
LE = 1;

for (L=1; L<=P; L++)
{
    LE1 = LE;
    LE = LE*2;

```

```

for (R=0; R<LE1; R++)
{
    WCOS = (float)DWR[IND];
    WSIN = (float)DWI[IND];
    IND = IND+1;
    for (IQ=R; IQ<M; IQ+=LE)
    {
        IP = IQ+LE1;

        A = (float)RE(X[IP]);
        B = (float)IM(X[IP]);

        VR = A*WCOS - B*WSIN;
        VI = B*WCOS + A*WSIN;

        RE(X[IP]) = RE(X[IQ]) - VR;
        IM(X[IP]) = IM(X[IQ]) - VI;

        RE(X[IQ]) = RE(X[IQ]) + VR;
        IM(X[IQ]) = IM(X[IQ]) + VI;
    }
}

free(DWR);
free(DWI);
}

/*=====*/
/*__1-D FFT with respect to a spatial coordinate_____*/
/*=====*/
int zffts( int debug, doublecomplex *X, int M )
{
    int P;
    float D;

    D = -1.0;

    rmpo( &M, &P);

    if ( debug )
    {
        printf("P = %d\n",P);
        printf("FFT ... \n");
    }

    zfftrmc(X, M, P, D); /* fftrm.c */

    return 0;
}

/*=====*/
/*__1-D Inverse FFT with respect to a spatial coordinate_____*/
/*=====*/
int ziffts( int debug, doublecomplex *X, int M )
{
    int i;
    int P;
    float D;

```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

```
D = 1.0;

rmpo( &M, &P);

if ( debug )
{
    printf("P = %d\n",P);
    printf("IFFT ... \n");
}

zfftrmc( X, M, P, D); /* fftrm.c */

/*__Multiply with 1/M__*/

for (i=0; i<M; i++)
{
    RE(X[i]) /= (double)M;
    IM(X[i]) /= (double)M;
}

return 0;
}/*End of fftrm.c*/
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

E.9.4 hilbert.h

```
/******  
/*hilbert.h  
/*This code contains the necessary functions for extracting envelope  
/******  
  
#ifndef HILBERT_H_  
#define HILBERT_H_  
  
/*This function reads the sampled data recorded in the file*/  
int ReadData(void);  
  
/*This function performs the Fourier transform*/  
void Fft(void);  
  
/*This function performs the necessary phase shift*/  
void PhaseShifting(void);  
  
/*This function performs the inverse Fourier transform*/  
void Ifft(void);  
  
/*Envelope reconstruction is done by this function*/  
int EnvelopeReconstruction(void);  
  
/*Hilbert main function*/  
void hilbert(char *fnamep);  
  
#endif /* HILBERT_H_ */
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011


```

const int col=2;

int ReadData(void)
{
    float a,b;
    int i=0;
    FILE *fp1;
    i=0;
    SampledPoints=0; //IA

    if ((fp1 = fopen(InputFileName,"r")) == NULL)
    {
        printf("Cannot open input file.\n");
        return 1;
    }

    while(!feof(fp1))
    {
        fscanf(fp1,"%e,%e\n", &a, &b);

        Gtime[SampledPoints] = a;
        Gvalue[SampledPoints] = b;
        SampledPoints++;
        if (SampledPoints>= MAX_POINT) break;
    }
    fclose(fp1);

    fp1=fopen("inputfile.txt","w");
    if (!fp1)
    {
        fprintf(stdout,"Can't write the sampled data in inputfile.txt. \n");
        return 1;
    }
    for(i=0; i<SampledPoints; i++)
    fprintf(fp1,"%e\n",Gvalue[i]); /*Gtime[i] has been omitted*/
    fclose(fp1);

    if(debug)
    {
        fp1=fopen("inputtime.txt","w");
        if (!fp1)
        {
            fprintf(stdout,"Can't write the sampled data in inputtime.txt. \n");
            return 1;
        }
        for(i=0; i<SampledPoints; i++)
        fprintf(fp1,"%e\n",Gtime[i]); /*Gtime[i] has been omitted*/
        fclose(fp1);
    }

    if(debug)
    {
        if((fp1=fopen("inputfile.bin","wb"))!=NULL)
        {
            fwrite(Gvalue,sizeof(double),SampledPoints,fp1);
            fclose(fp1);
        }
    }

    if(SampledPoints<N)
    {

```

```

        for(i=SampledPoints;i<=N;i++)
        {
            Gvalue[i] = 0;
        }
    }
    return 0;
}/*End Of Function ReadData;*/

void Fft(void)
{
    doublecomplex *Gt_freq;

    FILE *fp1,*fp2,*fp3;
    int k,num1,num2;

    Gt_freq = (doublecomplex *)calloc(sizeof(doublecomplex),row);

    /* FFT Procedure Starts for Sampled Data*/
    for(k=0;k<=N;k++)
    {
        RE(Gt_freq[k])=Gvalue[k];
        IM(Gt_freq[k])=0.0;
    }

    if(debug)
    {
        if((fp3=fopen("f.bin","wb"))!=NULL)
        {
            fwrite(Gvalue,sizeof(double),row,fp3);
            fclose(fp3);
        }
    }

    zffts(fftdebug,Gt_freq,row);/*FFT is done in spatial coordinate*/

    for (k=0;k<=N;k++)
    {
        Gr[k]=RE(Gt_freq[k]);
        Gi[k]=IM(Gt_freq[k]);
    }
    /* FFT Procedure Ends for Sampled Data*/

    /* Writing The Real And Imaginary Part Of Reflected Part for Debugging*/
    /* Writing the real part of sampled data*/

    if(debug)
    {
        if((fp1=fopen("Gr.bin","wb"))!=NULL)
        {
            num1=fwrite(Gr,sizeof(double),row,fp1);
            fclose(fp1);
        }
        else
            fprintf(stdout,"Can't Open Gr.bin");

        // Writing the img part of sampled data
        if((fp2=fopen("Gi.bin","wb"))!=NULL)
        {
            num2=fwrite(Gi,sizeof(double),row,fp2);
            fclose(fp2);
        }
    }
}

```

```

    }
    else
        fprintf(stdout, "Can't Open Gi.bin");

    fprintf(stdout, "Num of Real Part Data after FFT = %d\n", num1);
    fprintf(stdout, "Num of Img Part Data after FFT = %d\n", num2);
}

free(Gt_freq);

}/* End Of The Function Fft */

void PhaseShifting(void)
{
    double *tempr, *tempi;
    int k;
    FILE *fp1;

    tempr = (double *)calloc(sizeof(double), row);
    tempi = (double *)calloc(sizeof(double), row);

    for ( k=0; k<=N; k++ )
    {
        tempr[k]=Gr[k];
        tempi[k]=Gi[k];
    }

    for ( k=0; k<=ceil(N/2); k++ )
    {
        Gr[k] = tempi[k];
        Gi[k] = -tempr[k];
    }

    for ( k=(int)ceil(N/2)+1; k<=N; k++ )
    {
        Gr[k] = -tempi[k];
        Gi[k] = tempr[k];
    }

    if(debug)
    {
        if((fp1=fopen("fifrpt.bin", "wb")) !=NULL)
        {
            fwrite(Gr, sizeof(double), row, fp1);
            fclose(fp1);
        }
        if((fp1=fopen("ffipt.bin", "wb")) !=NULL)
        {
            fwrite(Gi, sizeof(double), row, fp1);
            fclose(fp1);
        }
    }
    free (tempr);
    free (tempi);
}/*End of PhaseShifting() function*/

void Ifft(void)
{
    double *Gt_tmp; /* It takes the real part of R_ifft*/

```

```

double *Gt_tmpi;
FILE *fp1;
int k,i;

Gt_tmp = (double *)calloc(sizeof(double),row);
Gt_tmpi = (double *)calloc(sizeof(double),row);

for (k=0;k<=N;k++)
{
    Gt_ifft[k].r=Gr[k];
    Gt_ifft[k].i=Gi[k];
}

ziffts(fftdebug,Gt_ifft,row);/*IFFT of the signal in spatial coordinate*/

// End of IFFT
for (k=0;k<=N;k++)
{
    Gt_tmp[k]=Gt_ifft[k].r;
}

if(debug)
{
    fp1=fopen("ifft.txt","w");
    if (!fp1)
        fprintf(stdout,"Can't write in file");
    for(i=0; i<=N; i++)
        fprintf(fp1,"%e\n", (Gt_ifft[i].r));
    fclose(fp1);
}

if(debug)
{
    if((fp1=fopen("iffrpt.bin","wb"))!=NULL)
    {
        fwrite(Gt_tmp,sizeof(double),row,fp1);
        fclose(fp1);
    }
    if((fp1=fopen("iffipt.bin","wb"))!=NULL)
    {
        fwrite(Gt_tmpi,sizeof(double),row,fp1);
        fclose(fp1);
    }
}
free(Gt_tmp );
free(Gt_tmpi );
}/* End Of Function Ifft*/

int EnvelopeReconstruction(void)
{
    FILE *fp1;
    int k;

    doublecomplex *G; /*Input signal read from input file in complex form*/
    doublecomplex *Ganalytical;/*Analytical function of our input signal*/

    double *test;
    double *sqrtr;
    double *sqrti;

```

```

G = (doublecomplex *) calloc(sizeof(doublecomplex), row);
Ganalytical = (doublecomplex *) calloc(sizeof(doublecomplex), row);

test = (double *) calloc(sizeof(double), row);
sqrtr=(double *) calloc(sizeof(double), row);
sqrtri=(double *) calloc(sizeof(double), row);

for (k=0;k<=N;k++)
{
    RE(G[k]) = Gvalue[k];
    IM(G[k]) = 0.0;
}

for (k=0;k<=N;k++)
{
    RE(Ganalytical[k])=G[k].r;
    IM(Ganalytical[k])=Gt_ifft[k].r;
}

for (k=0;k<=N;k++)
{
sqrtr[k]=sqrt(Ganalytical[k].r*Ganalytical[k].r+Ganalytical[k].i*Ganalytical[k].i);
}

fp1=fopen("output.txt","w");
if (!fp1)
{
    fprintf(stdout,"Can't write extracted envelope in output.txt.\n");
    free(G);
    free(Ganalytical);
    free(test);
    free(sqrtr);
    free(sqrtri);
    return 1;
}
for(k=0; k<SampledPoints; k++)
    fprintf(fp1,"%e,%e\n",Gtime[k],sqrtr[k]);
fclose(fp1);

free(G);
free(Ganalytical);
free(test);
free(sqrtr);
free(sqrtri);
return 0;
}

/*Main Function*/
void hilbert(char *fnamep)
{
    int status=0,i=1;
    char fname[256];
    strcpy(fname, fnamep);
    InputFileName= fname;

    //Reading the sampled data
    do
    {
        N=(int)pow(2,i)-1;

```

```
        i++;
    }while (MAX_POINT > N);

    if (debug)
        printf("N= %d\n",N);

    row=N+1;

    Gvalue = (double *)calloc(sizeof(double), row);
    Gtime = (double *)calloc(sizeof(double), row);
    Gr = (double *)calloc(sizeof(double), row);
    Gi = (double *)calloc(sizeof(double), row);
    Gt_ifft = (doublecomplex *)calloc(sizeof(doublecomplex), row);
    Gc = (double *)calloc(sizeof(double), row);

    status = ReadData();
    if (status== 1) goto MainExit;

    /*Does FFT*/
    Fft();

    /*Appropriate Phase has been Shifted*/
    PhaseShifting();

    /*Does IFFT*/
    Ifft();

    /*Envelope Reconstruction */
    status = EnvelopeReconstruction();
    if (status== 1) goto MainExit;

    MainExit:
    free(Gvalue);
    free(Gtime);
    free(Gr);
    free(Gi);
    free(Gt_ifft);
    free(Gc);

}/*End Of Main*/
```

E.9.6 functs.c

```

/*****
*/functs.c
/*This code contains all functions which provide the program functionality.
/*Main function of the whole program can be found at the end of this file.
/*****

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include "structures.h"
#include "hilbert.h"

#define MAX_SAMPLES 200000
#ifdef M_PI
#define M_PI 3.1415926535897932384626433832795
#endif

// Reads a line from a file (f) and returns two char arrays (s and t)
// It is used to read files written in "comma separation" format.
void read_line (FILE *f, char *s, char *t)
{
    int a=0;
    int i=0;

    a=fgetc(f); /* Takes chars from the file pointed by f */
    while (isspace(a)) /* spaces at the beginning of line are taken out */
    {
        a=fgetc(f);
    }
    while (a!=',' && a!=EOF) /* spaces at the beginning of line are taken out */
    {
        t[i++]=(char)a;
        a=fgetc(f);
    }
    t[i]='\0';
    i=0;
    a=fgetc(f);
    while (a!='\n' && a!=EOF)
    {
        s[i++]=(char)a;
        a=fgetc(f);
    }
    s[i]='\0'; /* We add the line end */
}

// Reads a line from a file (f) and discards it.
void skip_line (FILE *f)
{
    int a=0;

    a=fgetc(f);
    while (a!='\n' && a!=EOF)
    {
        a=fgetc(f);
    }
}

```

```

// Creates a ESL node with the given volt and time parameters
ESL *createnodef(double voltf, double timef)
{
    ESL *new=NULL;
    new=(ESL *)malloc(sizeof(ESL));
    if (new!=NULL)
    {
        new->volt=voltf;
        new->time=timef;
        new->sig=NULL;
    }
    else
        fprintf(stderr,"Memory Error");
    return new;
}

// Frees the allocated memory for ESL nodes
void freelist(ESL *first) // frees ESL list
{
    ESL *to_free;
    while (first->sig!=NULL)
    {
        to_free=first;
        first=first->sig;
        free(to_free);
    }
}

// Creates a TIME node with the given volt and time parameters
void createtime(TIMES *new, double tr, double tf, double b, double trstartind,
double trendind, double tfstartind, double tfendind, double t1, double
t1startind, double t1start, double t1endind, double t2, double t2startind, double
t2start, double t3, double t3end, double t3endind, double t4, double t4endind,
double t5, double t5startind, double t6, double t6end, double t6endind, double a,
double tploone)
{
    new->tf=tf;
    new->tr=tr;
    new->b=b;
    new->trstartind=trstartind;
    new->trendind=trendind;
    new->tfstartind=tfstartind;
    new->tfendind=tfendind;
    new->t1=t1;
    new->t1startind=t1startind;
    new->t1start=t1start;
    new->t1endind=t1endind;
    new->t2=t2;
    new->t2startind=t2startind;
    new->t2start=t2start;
    new->t3=t3;
    new->t3end=t3end;
    new->t3endind=t3endind;
    new->t4=t4;
    new->t4endind=t4endind;
    new->t5=t5;
    new->t5startind=t5startind;
    new->t6=t6;
    new->t6end=t6end;
    new->t6endind=t6endind;
}

```

```

    new->a=a;
    new->tploone=tploone;
}

// Inserts a ESL node (new) in a list pointed by "first"
void insert_node(ESL **first, ESL *new)
{
    ESL *p=NULL;
    ESL *previous=NULL;
    if (new!=NULL)
    {
        p=*first;
        if (p==NULL)
        {
            *first=new;
        }
        else
        {
            while (p!=NULL)
            {
                previous=p;
                p=p->sig;
            }
            previous->sig=new;
        }
    }
}

/* Multiplies order polynomials supposing (x^2 + b*x + c) */
/* b and c are complex values stored in a table where even elements are */
/* real and odd elements imaginary */
double *mult_poli (int num_pol, double *b, double *c)
{
    int i=0;
    int y=0;
    double *real;
    double *imag;
    double *vector;
    double *new_real;
    double *new_imag;
    double real_b=0;
    double real_c=0;
    double imag_b=0;
    double imag_c=0;

    real=(double *)calloc(4*num_pol, sizeof(double));
    imag=(double *)calloc(4*num_pol, sizeof(double));
    new_real=(double *)calloc(4*num_pol, sizeof(double));
    new_imag=(double *)calloc(4*num_pol, sizeof(double));
    vector=(double *)calloc(4*num_pol, sizeof(double));

    real[0]=c[0];
    real[1]=b[0];
    real[2]=1;
    imag[0]=c[1];
    imag[1]=b[1];
    imag[2]=0;

```

```

for (i=3; i<(4*num_pol); i++)
{
    real[i]=0;
    imag[i]=0;
}

for (y=1; y<num_pol; y++)
{
    // Selects values b and c
    real_b=b[2*y];
    real_c=c[2*y];
    imag_b=b[2*y+1];
    imag_c=c[2*y+1];

    for (i=0; i<=(2*num_pol-2); i++)
    {
        // Starts with coeff "c"
        new_real[i]+=real[i]*real_c-imag[i]*imag_c;
        new_imag[i]+=real[i]*imag_c+imag[i]*real_c;
        // Continues with coeff "b"
        new_real[i+1]+=real[i]*real_b-imag[i]*imag_b;
        new_imag[i+1]+=real[i]*imag_b+imag[i]*real_b;
        // Finishes with coeff "1"
        new_real[i+2]+=real[i];
        new_imag[i+2]+=imag[i];
    }

    // Update Values
    for (i=0; i<(4*num_pol); i++)
    {
        real[i]=new_real[i];
        imag[i]=new_imag[i];
        new_real[i]=0;
        new_imag[i]=0;
    }
}

for (y=0; y<(2*num_pol); y++)
{
    vector[2*y]=imag[y];
    vector[2*y+1]=real[y];
}

free (new_imag);
free (new_real);
free (imag);
free (real);
return (vector);
}

// Part of the calculation of the butterworth coeffs.
double *butter_d_coeffs(double freq1, double freq2)
{
    int butter_order=2;
    int index=0;
    double theta=0; // M_PI *(freq2-freq1)/2.0
    double cp=0; // cosine of phi
    double *vec_r=0; // z^-2 coefficients
    double *vec_t=0; // z^-1 coefficients
    double *dcoeff=0; // d coefficients

```

```

double pole_ang=0;          // pole angle
double divisor=0;

cp = cos(M_PI*(freq2+freq1)/2.0);
theta = M_PI*(freq2-freq1)/2.0;

vec_r=(double *)calloc(2*butter_order,sizeof(double));
vec_t=(double *)calloc(2*butter_order,sizeof(double));

for(index=0;index<butter_order;++index)
{
    pole_ang=M_PI*(double)(2*index+1)/(double)(2*butter_order);
    divisor=sin(2*theta)*sin(pole_ang)+1.0;
    vec_r[2*index]=cos(2*theta)/divisor;
    vec_r[2*index+1]=sin(2*theta)*cos(pole_ang)/divisor;
    vec_t[2*index]=-2.0*cp*(cos(theta)+sin(theta)*sin(pole_ang))/divisor;
    vec_t[2*index+1]=-2.0*cp*sin(theta)*cos(pole_ang)/divisor;
}

dcoeff=mult_poli(butter_order,vec_t,vec_r);

dcoeff[4]=dcoeff[1];
dcoeff[3]=dcoeff[3];
dcoeff[2]=dcoeff[5];
dcoeff[1]=dcoeff[7];
dcoeff[0]=1;

for(index=5;index<=2*butter_order;index++)
    dcoeff[index]=0;

free(vec_t);
free(vec_r);

return(dcoeff);
}

// Calculates the Butterworth filter coefficients
void butterworth_coeffs(double freq1, double freq2, double *dfiltercoeff, double
*cfiltercoeff)
{
    // n filter order
    // freq1, freq2 lower/uppercutoff frequencies

    double sf;          // scaling factor
    double *dcoeff;    // d coefficients
    double cotan=0;    // cotangent of theta

    /* calculate the d coefficients */
    dcoeff=butter_d_coeffs(freq1,freq2);

    /* d coefficients for 4th order butterworth */
    dfiltercoeff[0]=dcoeff[0]; // Always 1
    dfiltercoeff[1]=dcoeff[1];
    dfiltercoeff[2]=dcoeff[2];
    dfiltercoeff[3]=dcoeff[3];
    dfiltercoeff[4]=dcoeff[4];

    /* scaling factor for the c filter coefficients (Butterworth 4th order */
    cotan=1.0/tan(M_PI*(freq2-freq1)/2.0);
    sf=(1.0/(((cotan+sqrt(2)/2)*(cotan+sqrt(2)/2))+1/2));
}

```

```

/* c coefficients for 4th order butterworth*/
cfiltercoeff[0]=1*sf;
cfiltercoeff[1]=0*sf;
cfiltercoeff[2]=-2*sf;
cfiltercoeff[3]=0*sf;
cfiltercoeff[4]=1*sf;

free(dcoeff);
}

// Checks if the data input is adequate to our algorithms
int datacheck(int posval, int negval, int samplesp, double tlast, FILE *pointfile)
{
double diff=0.0;
char timestr1[25];
char timestr2[25];
char voltstr[25];
double timestrf1=0;
double timestrf2=0;
double cut_sample=0;
double delta_t=0;
double val_t=0;
int loop=0;
double linf=0;
int lind=0;

// Checks that there are (nearly) much positive as negative values
if (posval>negval)
diff=(posval-negval)/((posval+negval)/2);
else
diff=(negval-posval)/((posval+negval)/2);

if (diff>0.8)
fprintf(stdout, "Data Corrupted: Too little negative (or positive)
values\n");

// L=n*pi with P=2*pi and n=1,2,3... - Cuts data
rewind (pointfile);
read_line(pointfile,voltstr,timestr1);
read_line(pointfile,voltstr,timestr1);
read_line(pointfile,voltstr,timestr1); // Skips csv header if present or
not

read_line(pointfile,voltstr,timestr1);
while (voltstr[0]!='\0')
{
read_line(pointfile,voltstr,timestr2);
loop++;
}
loop=loop+3;

rewind(pointfile);
for (lind=0; lind<loop; lind++)
{
read_line(pointfile,voltstr,timestr2);
}

timestrf1=atof(timestr1); // t(4)
timestrf2=atof(timestr2); // t(end)
cut_sample=((1/13.56e6)/((timestrf2-timestrf1)/(loop-1)));
linf=samplesp;

```

```

while (linf>0)
{
    linf=linf-cut_sample;
}
linf+=cut_sample;
samplesp=samplesp-linf-3;

// At least 7 points per sample
delta_t=tlast-timestrfl;
val_t=delta_t/samplesp;
if (val_t>(1/13.56e6)/7)
    fprintf(stdout, "More samples points needed - Nyquist\n");

return (samplesp);
}

// Finds the most frequent value(s) of the given signal, Hmax (Types A/B) and
Hmin (Type B).
void Hmaxfinder(double *env, double *Hmax, double *Hmin, int numsamples)
{
    int hist[2001]={0}; // IA Changed memory access violation. Increased +1
    int hi_low_i=0;
    double tophist=0;
    double bothist=100;
    double diffhist=0;
    double value=0;
    int histind=0;
    int max_i=0;
    int min_i=0;
    double max=0;
    double min=0;

    // Finds higher and lower values of samples
    for (hi_low_i=0; hi_low_i<MAX_SAMPLES; hi_low_i++)
    {
        if (env[hi_low_i]!=0)
        {
            if (env[hi_low_i]<bothist)
                bothist=env[hi_low_i];
            if (env[hi_low_i]>tophist)
                tophist=env[hi_low_i]; // Finds limits for the histogram
        }
    }
    diffhist=tophist-bothist;

    for (hi_low_i=0; hi_low_i<numsamples; hi_low_i++)
    {
        if (env[hi_low_i]!=0)
        {
            value=env[hi_low_i];
            histind=(int) (2000*((value-bothist)/diffhist)); // Performs a lineal
quantization
            hist[histind]++;
        }
    }

    for (hi_low_i=0; hi_low_i<1000; hi_low_i++)
    {
        if (hist[hi_low_i]>min) // Searches most frequent value in the lower
half of the form
        {

```

```

        min=hist[hi_low_i];
        min_i=hi_low_i;
    }
    *Hmin=(bothist+(diffhist/2000)*(min_i));
}

for (hi_low_i=1001; hi_low_i<2000; hi_low_i++)
{
    if (hist[hi_low_i]>max)    // Search most frequent value in the upper half
of the form
    {
        max=hist[hi_low_i];
        max_i=hi_low_i;
    }
    *Hmax=(bothist+(diffhist/2000)*(max_i));
}
}

// Linear convolution (z= x convolve y)
void LinearConvolution(double X[],double Y[], double Z[], int lenx, int leny)
{
    double *zptr,s,*xp,*yp;
    int lenz;
    int i,n,n_lo,n_hi;

    lenz=lenx+leny-1;
    zptr=Z;

    for (i=0;i<lenz;i++)
    {
        s=0.0;
        n_lo=0>(i-lenx+1)?0:i-lenx+1;
        n_hi=lenx-1<i?lenx-1:i;
        xp=X+n_lo;
        yp=Y+i-n_lo;

        for (n=n_lo;n<=n_hi;n++)
        {
            s+=*xp * *yp;
            xp++;
            yp--;
        }

        *zptr=s;
        zptr++;
    }
}

int envfilt(double *output, double *toutput, int filterlength, double tini,
double tend, int lengthp, double *envelope)
{
    int cofpi=0;
    int xx=0;
    double cofp=0.0;
    int lengthp1=0;
    double lengthf=0;
    double cof[2000]={0};
    double points=0.0;
    int pointsi=0;
    int lengthtotal=0;

```

```

cofp=(73.75e-9)/((tend-tini)/(lengthp));
cofpi=cofp+0.5;
lengthf=cofpi*filterlength;
points=(5*73.75e-9)/((tend-tini)/(lengthp-1));
pointsi=(int)points+1;
lengthp1=lengthp;

for (xx=0; xx<lengthf; xx++)
    cof[xx]=1/lengthf;

for (xx=lengthf+1; xx<2000; xx++)
    cof[xx]=0;

LinearConvolution(cof, output, envelope, lengthf, lengthp);

for (xx=0; xx<(pointsi); xx++) // "Cuts" envelope
{
    envelope[xx]=0.0;
    toutput[xx]=0.0;
    envelope[lengthp1-xx]=0.0;
    toutput[lengthp1-xx]=0.0;
}

for (xx=lengthp1+1; xx<MAX_SAMPLES; xx++)
{
    envelope[xx]=0.0;
    toutput[xx]=0.0;
}

lengthtotal=lengthp1-2*(pointsi);
return (lengthtotal);
}

// Performs the search of a certain level (target) in the envelope, i.e. 5% ,60%,
90% in Type A, 106 kbit/s
int localizador(double *env, double *toutput, double target, ESL **crosses, int
env_length)
{
    int flag=0;
    double diff;
    ESL *new;
    double v;
    double t;
    int crosscounter=0;
    int locat_index=0;
    int locat_index_start=0;

    while (env[locat_index]==0.0) // Leaves 0s out
        locat_index++;

    locat_index_start=locat_index;
    if (env[locat_index]-target>0)
    {
        for (locat_index=locat_index_start;
locat_index<env_length+locat_index_start-1; locat_index++)
        {
            diff=env[locat_index]-target;
            if (diff<0 && flag==0 && env[locat_index]!=0.0) // At the beginning or
after an odd occurrence, envelope is over "target" level
            {
                flag=1; // down!
            }
        }
    }
}

```

```

        v=target;
        t=toutput[locat_index-1]+((toutput[locat_index]-toutput[locat_index-1])/(env[locat_index]-env[locat_index-1])*(target-env[locat_index-1]));
        new=createnodef(v,t);
        insert_node(crosses,new);
        crosscounter++;
    }
    if (diff>0 && flag==1 && env[locat_index]!=0.0) // After first (or
even) occurrence, envelope is under "target" level
    {
        flag=0; // up!
        v=target;
        t=toutput[locat_index-1]+((toutput[locat_index]-toutput[locat_index-1])/(env[locat_index]-env[locat_index-1])*(target-env[locat_index-1]));
        new=createnodef(v,t);
        insert_node(crosses,new);
        crosscounter++;
    }
    // Returns all occurrences with time and volt level in a list
}
}
else
    fprintf(stdout,"Signal is not ---|___|--- \n");

    return (crosscounter); // Also returns how many occurrences
appeared
}

// Function that calculates the relevant times
void tfinder(char type, double *env, double *toutput, double tini, double Hmax,
double Hmin, int rate, int env_length, TIMES *timeres)
{
    double *envc=NULL;
    double *envc2=NULL;
    ESL *crosses=NULL;
    ESL *crosses2=NULL;
    ESL *crosses3=NULL;
    ESL *crosses_WORK=NULL;
    double ninety=0.0;
    double five=0.0;
    double sixty=0.0;
    double tp90one=0.0;
    double tp90two=0.0;
    double tp5one=0.0;
    double tp5two=0.0;
    double tp60two=0.0;
    double vp90one=0.0;
    double vp90two=0.0;
    double vp5one=0.0;
    double vp5two=0.0;
    double vp60two=0.0;
    double tphione=0.0;
    double tphitwo=0.0;
    double vphione=0.0;
    double vphitwo=0.0;
    double tpmidone=0.0;
    double vpmidone=0.0;
    double tploone=0.0;
    double tplotwo=0.0;
    double vplotwo=0.0;
    double t1=0.0;

```

```

double t2=0.0;
double t3=0.0;
double t4=0.0;
double t5=0.0;
double t6=0.0;
int flag=0;
int flag2=0;
int flag3=0;
int flag_improv=0;
int x_improv=0;
double minvolt=0.0;
double highrate_low=0.0;
double highrate_mid=0.0;
double highrate_hi=0.0;
double a=0.0;
double t6end=0.0;
double t5startind=0.0;
double t6endind=0.0;
double b=0.0;
double B_low=0.0;
double B_hi=0.0;
double tr=0.0;
double tf=0.0;
double tfstartind=0.0;
double tfendind=0.0;
double trstartind=0.0;
double trendind=0.0;
double t2startind=0.0;
double t2start=0.0;
double t1startind=0.0;
double t1start=0.0;
double t3end=0.0;
double t4endind=0.0;
double t3endind=0.0;
double t1endind=0.0;
double oscmin=0.0;
double osctmin=0.0;
double oscmax=0.0;
double osctmax=0.0;
ESL *crossescopy=NULL;
double tim3=0.0;
int index_A=0;
int index_A2=0;
int index_chain=0;
int i=0;

b=Hmin;
envc=env;
envc2=envc;

switch (type)
{
  case 'A':
  {
    switch (rate)
    {
      case 106:
      {
        ninety=Hmax*0.9;
        five=Hmax*0.05;
        sixty=Hmax*0.6;
      }
    }
  }
}

```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

```

        flag2=localizador(envc,toutput,five,&crosses2,env_length);    //
Finds 5% of Hmax
        if (flag2==2)        // if there are two occurrences, there's no
problem...
        {
            tp5one=crosses2->time;        // Temporary values are stored
for future use
            vp5one=crosses2->volt;
            tp5two=crosses2->sig->time;
            vp5two=crosses2->sig->volt;
            freelist(crosses2);
        }
        else if (flag2==0)        // ...if there is no occurrence...
            fprintf(stdout,"5 percent of Hmax not reached - maybe wrong
type or bitrate? \n");

        else if (flag2>2)        // ...if there are more than two
occurrences...
        {
            // ...it must be checked that "peaks" comply
the ISO restrictions

            while (toutput[index_A]<crosses2->sig->time)
                index_A++;

            oscmin=envc2[index_A];
            for (index_chain=0; index_chain<flag2-2; index_chain++)
            {
                crosses2=crosses2->sig;
            }

            while (toutput[index_A]<=crosses2->time)
            {
                if (envc2[index_A]>oscmx)
                {
                    oscmx=envc2[index_A];
                    osctmx=toutput[index_A];
                }
                index_A++;
            }

            while (envc2[index_A2]==0)
            {
                index_A2++;
            }

            while (envc2[index_A2]>oscmx)
            {
                index_A2++;
            }

            oscmin=envc2[index_A2];
            osctmin=toutput[index_A2];

            if (osctmx-osctmin>5e-7)
                fprintf(stdout,"Monotony not fulfilled \n");

            tp5one=crosses2->time;        // Temporary values are stored
for future use
            vp5one=crosses2->volt;
            tp5two=crosses2->sig->time;

```

```

        vp5two=crosses2->sig->volt;
        freelist(crosses2);
    }

    flag=localizador(envc,toutput,ninety,&crosses,env_length);    //
Finds 90% of Hmax
    if (flag>=2)
    {
        crosses_WORK=crosses;    // Copy of crosses to work with
        while (x_improv<flag)
        {
            if (crosses_WORK->time<tp5one)
            {
                tp90one=crosses_WORK->time;    // Temporary values are
stored for future use
                vp90one=crosses_WORK->volt;
            }

            if (crosses_WORK->time>tp5two && flag_improv==0)
            {
                tp90two=crosses_WORK->time;    // Temporary values are
stored for future use
                vp90two=crosses_WORK->volt;
                flag_improv=1;
            }
            crosses_WORK=crosses_WORK->sig;
            x_improv++;
        }
        freelist(crosses);
    }

    else    // ...otherwise...
    {
        fprintf(stdout,"90 %% of Hmax not found - Noise Too High \n");
    }

    flag3=localizador(envc,toutput,sixty,&crosses3,env_length);    //
Finds 60% of Hmax
    if (flag3==2)    // if there are two occurrences,
there's no problem...
    {
        tp60two=crosses3->sig->time;    // Temporary values are
stored for future use
        vp60two=crosses3->sig->volt;
        freelist(crosses3);
    }

    t1=tp5two-tp90one;    // Definitive values are calculated and
stored for display
    t2=tp5two-tp5one;
    t3=tp90two-tp5two;
    t4=tp60two-tp5two;

    t1start=tp90one;    // Other important values for the coming
functions
    t2start=tp5one;
    t3end=tp90two;
    t1startind=vp90one;
    t1endind=vp5two;
    t2startind=vp5one;

```

STANDARD.PDF.COM: Click to view the full PDF of ISO/IEC 10373-6:2011

```

        t3endind=vp90two;
        t4endind=vp60two;

createtime(timeres,0,0,0,0,0,0,0,t1,t1startind,t1start,t1endind,t2,t2startind,t2s
tart,t3,t3end,t3endind,t4,t4endind,0,0,0,0,0,0,0);

    }
    break;

    case 212:
    case 424:
    case 848:
    {
        ninety=Hmax*0.9;
        while (env[index_A]==0.0)           // Finds first value different of
0.0
            index_A++;

        minvolt=env[index_A];
        while (env[index_A]!=0.0)         // All values are considered
        {
            if (env[index_A]<minvolt)
            {
                minvolt=env[index_A];    // Finds minimal voltage
            }
            index_A++;
        }

        highrate_low=minvolt+0.1*(Hmax-minvolt);    // Calculates target
        flag=localizador(envc,toutput,highrate_low,&crosses,env_length);
// Finds target
        if (flag==2)                       // ...if there are two occurrences,
there's no problem...
        {
            tploone=crosses->time;         // Temporary values are stored
for future use
            tplotwo=crosses->sig->time;
            vplotwo=crosses->sig->volt;
        }

        else if (flag>2)                   // if there are more than two
occurrences...
        {
            // ...it must be checked that "peaks" comply
the ISO restrictions
            while (toutput[index_A2]<crosses->time || toutput[index_A2]==0)
                index_A2++;

            oscmin=envc2[index_A2];
            while (envc2[index_A2]<=oscmin)
            {
                oscmin=envc2[index_A2];
                index_A2++;
            }
            osctmin=toutput[index_A2];

            crossescopy=crosses;
            for (i=1; i<(flag-1); i++)
                crossescopy=crossescopy->sig;

            tim3=crossescopy->time;

```

```

while (toutput[index_A2]<tim3)
    index_A2++;

oscmx=envc2[index_A2];
while (toutput[index_A2]<tim3)
{
    if (oscmx<envc2[index_A2])
    {
        oscmx=envc2[index_A2];
        osctmx=toutput[index_A2];
    }
    index_A2++;
}

if (oscmx-oscmn>(0.09*(Hmax-oscmn)))
    fprintf(stdout,"Monotony not fulfilled \n");

for (i=1; i<(flag-1); i++)
    crosses=crosses->sig;

tploone=crosses->time;           // Temporary values are stored
for future use
tplotwo=crosses->sig->time;
vplotwo=crosses->sig->volt;
}
freelist(crosses);

highrate_hi=ninety+0.1*minvolt; // Calculates target
flag=localizador(envc,toutput,highrate_hi,&crosses2,env_length);
// Finds target
if (flag>=2)
{
    crosses_WORK=crosses2;
    while (x_improv<flag)
    {
        if (crosses_WORK->time<tploone)
        {
            tphione=crosses_WORK->time;           // Temporary values are
stored for future use
            vphione=crosses_WORK->volt;
        }
        if (crosses_WORK->time>tplotwo && flag_improv==0)
        {
            tphitwo=crosses_WORK->time;           // Temporary values are
stored for future use
            vphitwo=crosses_WORK->volt;
            flag_improv=1;
        }
        crosses_WORK=crosses_WORK->sig;
        x_improv++;
    }
    freelist(crosses2);
}
else // ...otherwise...
{
    fprintf(stdout,"90 %% of Hmax not reached! - Noise Too High?
\n");
}

```

```

        highrate_mid=(Hmax+minvolt)/2;        // Calculates target
        flag=localizador(envc,toutput,highrate_mid,&crosses3,env_length);
// Finds target
        if (flag==2)                            // ...if there are two occurrences,
there's no problem...
        {
            tpmidone=crosses3->time;            // Temporary values are stored
for future use
            vpmidone=crosses3->volt;
            freelist(crosses3);
        }
        else                                    // ...otherwise...
            fprintf(stdout,"Noise Too High \n");

        t1=tplotwo-tphone;                    // Definitive values are calculated
and stored for display
        t5=tplotwo-tpmidone;
        t6=tphitwo-tplotwo;
        a=minvolt;

        t6end=tphitwo;                        // Other important values for the coming
functions
        t1start=tphone;
        t1startind=vphone;
        t5startind=vpmidone;
        t1lendind=vplotwo;
        t6endind=vphitwo;

createtime(timeres,0,0,0,0,0,0,0,t1,t1startind,t1start,t1lendind,0,0,0,0,0,0,0,t
5,t5startind,t6,t6end,t6endind,a,tploone);
    }
    break;
}
break;

case 'B':
{
    B_low=b+0.1*(Hmax-b);                    // Calculates target
    flag=localizador(envc,toutput,B_low,&crosses,env_length); // Finds
target
    if (flag>=2)
    {
        crosses_WORK=crosses;
        tploone=crosses_WORK->time;          // Temporary values are stored
for future use
        while (x_improv<flag)
        {
            tplotwo=crosses_WORK->time;      // Temporary values are stored
for future use
            vplotwo=crosses_WORK->volt;
            crosses_WORK=crosses_WORK->sig;
            x_improv++;
        }
        freelist(crosses);
    }
    else
    {
        fprintf(stdout,"Monotony not fulfilled\n");
    }
}

```

```

        B_hi=Hmax-0.1*(Hmax-b);           // Calculates target
        flag=localizador(envc,toutput,B_hi,&crosses2,env_length); // Finds
target
        if (flag>=2)
        {
            x_improv=0;
            flag_improv=0;
            crosses_WORK=crosses2;
            while (x_improv<flag)
            {
                if (crosses_WORK->time<tploone)
                {
                    for future use
                    tphione=crosses_WORK->time;           // Temporary values are stored
                    vphione=crosses_WORK->volt;
                }

                if (crosses_WORK->time>tplotwo && flag_improv==0)
                {
                    for future use
                    tphitwo=crosses_WORK->time;           // Temporary values are stored
                    vphitwo=crosses_WORK->volt;
                    flag_improv=1;
                }
                crosses_WORK=crosses_WORK->sig;
                x_improv++;
            }
            freelist(crosses2);
        }
        else
        {
            fprintf(stdout,"Monotony not fulfilled\n");
        }

        tf=tploone-tphione;           // Definitive values are calculated and stored
        for display
        tr=tphitwo-tplotwo;
        tfstartind=tphione;           // Other important values for the coming
        functions
        tfendind=tploone;
        trstartind=tplotwo;
        trendind=tphitwo;

        createtime(timeres,tr,tf,b,trstartind,trendind,tfstartind,tfendind,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0);
    }
    break;
}

// Checks monotony on the falling edge
void monocheck(double *env, double *toutput, double Hmax, TIMES *timesp, int
rate, char type)
{
    double tinit=0.0;
    double tend=0.0;
    double compare=0.0;

```

```

double timer0=0.0;
double timer1=0.0;
double volt0=0.0;
double volt1=0.0;
int counter=0;
int flag_mono=0;

switch (type)
{
  case 'A':
  {
    switch (rate)
    {
      case 106:
      {
        while (env[counter]==0)
          counter++;

        tinit=timesp->t1start;
        tend=timesp->t2start;

        while (toutput[counter]<tinit)
          counter++; //find first value

        while (toutput[counter]<tend)
        {
          compare=env[counter];
          if (compare<env[counter+1])
          {
            timer0=toutput[counter];
            volt0=env[counter];
            while (volt0<env[counter+1]) // growing values...
            {
              counter++;
              volt0=env[counter];
            }
            timer1=toutput[counter]; // ...max.value -> time
            if (timer1-timer0>5e-6)
              fprintf(stdout,"Monotony not fulfilled \n");
          }
          else
            counter++;
        }
        break;
      }
      case 212:
      case 424:
      case 848:
      {
        while (env[counter]==0)
          counter++;

        tinit=timesp->t1start;
        tend=timesp->t1endind;
        while (toutput[counter]<tinit)
          counter++; //find first value

        while (env[counter]>tend)
        {
          compare=env[counter];

```

```

        if (compare<env[counter+1])
        {
            volt0=env[counter];
            volt1=volt0;
            while (volt0<env[counter+1]) // growing values...
            {
                counter++;
                volt0=env[counter];
            }
            if (volt1-volt0>0.09*(Hmax-volt0))
                fprintf(stdout,"Monotony not fulfilled \n");
        }
        else
            counter++;
    }
}
break;
}
break;

case 'B':
{
    while (env[counter]==0)
        counter++;

    tinit=timesp->tfstartind;
    tend=timesp->tfendind;
    while (toutput[counter]<tinit)
        counter++; //find first value

    while (toutput[counter]<tend)
    {
        compare=env[counter];
        if (compare<env[counter+1])
            flag_mono=1;
    }
    if (flag_mono==1)
        fprintf(stdout,"Monotony not fulfilled \n");
}
break;
}
}

// Function that calculates the overshoot times
void overshoot(TIMES *timesp, double Hmax, double *env2, double *toutput, int
rate, char type, int samples, SHOOTREADER *shootreader)
{
    double shootind=0.0;
    double shootind_b=0.0;
    double hr_reader=0.0;
    double hf_reader=0.0;
    double above=0.0;
    double above_b=10.0;
    double start=0.0;
    int index_samples=0;

    switch (type)
    {
        case 'A':
        {

```

```

switch (rate)
{
  case (106):
  {
    start=timesp->t3end;
    while (toutput[index_samples]<=start)
      index_samples++;

    while (index_samples<=samples)
    {
      if (env2[index_samples]>above)
      {
        above=env2[index_samples];
        shootind=toutput[index_samples];
      }
      index_samples++;
    }
  }
  break;
  case (212):
  case (424):
  case (848):
  {
    start=timesp->t6end;
    while (toutput[index_samples]<=start)
      index_samples++;

    while (index_samples<=samples)
    {
      if (env2[index_samples]>above)
      {
        above=env2[index_samples];
        shootind=toutput[index_samples];
      }
      index_samples++;
    }

    if (above<Hmax) // In very strange cases if there's no
overshoot, the highest point
      above=Hmax; // in the curve can be cutted off by envfilt,
producing a negative hr
  }
  break;
}
break;
case 'B':
{
  start=timesp->trendind;
  while (toutput[index_samples]<start) // Starts at the rising
edge
    index_samples++;

  while (index_samples<=samples)
  {
    if (env2[index_samples]>above)
    {
      above=env2[index_samples];
      shootind=toutput[index_samples];
      hr_reader=(above-Hmax)/(Hmax-timesp->b);
    }
  }
}
}

```

```

        if (hr_reader<0) // In very strange cases if there's no
overshoot, the highest point
            hr_reader=0; // in the curve can be cutted off by envfilt,
producing a negative hr
    }
    index_samples++;
}

index_samples=0;
start=timesp->tfendind;
while (toutput[index_samples]==0)
    index_samples++;
while (toutput[index_samples]<start)
    index_samples++;

while (toutput[index_samples]<(timesp->trstartind))
{
    if (env2[index_samples]<above_b && env2[index_samples]!=0)
    {
        above_b=env2[index_samples];
        shootind_b=toutput[index_samples];
        hf_reader=(timesp->b-above_b)/(Hmax-timesp->b);
    }
    index_samples++;
}
}
break;

}
shootreader->shootind=shootind;
shootreader->shootind_b=shootind_b;
shootreader->hr_reader=hr_reader;
shootreader->hf_reader=hf_reader;
shootreader->above=above;
shootreader->above_b=above_b;
}

// Calculates the modulation index "m"
double modulation(char type, double Hmax, double b)
{
    double m=0;
    switch (type)
    {
        case 'A':
        {
            // m is not defined for Type A
        }
        break;
        case 'B':
        {
            m=100*(Hmax-b)/(Hmax+b); // In %
        }
        break;
    }
    return (m);
}

// Displays on screen the results of the calculations
void display(char type, int rate, SHOOTREADER *shootreader2, TIMES *timesp,
double Hmax, double m)
{

```

```

double ovs=0;
double ovsb1=0;
double ovsb2=0;
fprintf (stdout, "\n"); // 2nd set of functions, on debug purposes
switch (type)
{
  case 'A':
  {
    fprintf(stdout, "---RESULTS-----\n");
    fprintf(stdout, "Type A - Bitrate %d\n", rate);
    fprintf(stdout, "---Overshoot-----\n");
    ovs=(((shootreader2->above)-Hmax)/(Hmax-timesp->a))*100;
    if (ovs>0)
      fprintf(stdout, "Overshoot = %f %% \n", ovs);
    else
      fprintf(stdout, "Overshoot = 0 %% \n");
    switch (rate)
    {
      case (106):
      {
        fprintf(stdout, "---timings-----\n");
        fprintf(stdout, "t1 = %f microsec. \n", (timesp->t1)*1e6);
        fprintf(stdout, "t1 = %f/fc \n", (timesp->t1)*13.56e6);
        fprintf(stdout, "t2 = %f microsec. \n", (timesp->t2)*1e6);
        fprintf(stdout, "t2 = %f/fc \n", (timesp->t2)*13.56e6);
        fprintf(stdout, "t3 = %f microsec. \n", (timesp->t3)*1e6);
        fprintf(stdout, "t3 = %f/fc \n", (timesp->t3)*13.56e6);
        fprintf(stdout, "t4 = %f microsec. \n", (timesp->t4)*1e6);
        fprintf(stdout, "t4 = %f/fc \n", (timesp->t4)*13.56e6);
        fprintf(stdout, "---amplitudes-----\n");
        fprintf(stdout, "Hmax = %f volts \n", Hmax);
        fprintf(stdout, "Max. Amplitude = %f volts \n", (shootreader2-
>above));
      }
      break;
      case (212):
      case (424):
      case (848):
      {
        fprintf(stdout, "hovs = %f \n", (((shootreader2->above-
Hmax)/Hmax));
        fprintf(stdout, "---timings-----\n");
        fprintf(stdout, "t1 = %f microsec. \n", (timesp->t1)*1e6);
        fprintf(stdout, "t1 = %f/fc \n", (timesp->t1)*13.56e6);
        fprintf(stdout, "t5 = %f microsec. \n", (timesp->t5)*1e6);
        fprintf(stdout, "t5 = %f/fc \n", (timesp->t5)*13.56e6);
        fprintf(stdout, "t6 = %f microsec. \n", (timesp->t6)*1e6);
        fprintf(stdout, "t6 = %f/fc \n", (timesp->t6)*13.56e6);
        fprintf(stdout, "---amplitudes-----\n");
        fprintf(stdout, "Hmax = %f volts \n", Hmax);
        fprintf(stdout, "a = %f %% of Hinitial \n", ((timesp->a)/Hmax));
      }
      break;
    }
  }
  break;
  case 'B':
  {
    fprintf(stdout, "---RESULTS-----\n");
    fprintf(stdout, "Type B - Bitrate %d\n", rate);
    fprintf(stdout, "---timings-----\n");

```

```

    fprintf(stdout,"tf = %f microsec. \n", (timesp->tf)*1e6);
    fprintf(stdout,"tf = %f/fc \n", (timesp->tf)*13.56e6);
    fprintf(stdout,"tr = %f microsec. \n", (timesp->tr)*1e6);
    fprintf(stdout,"tr = %f/fc \n", (timesp->tr)*13.56e6);
    fprintf(stdout,"---modulation-----\n");
    fprintf(stdout,"m = %f %% \n", m);
    fprintf(stdout,"---amplitudes-----\n");
    fprintf(stdout,"a = %f volts \n", Hmax);
    fprintf(stdout,"b = %f volts \n", timesp->b);
    fprintf(stdout,"---Overshoots-----\n");
    fprintf(stdout,"hf = %f %% of Hinitial-b\n", (shootreader2-
>hf_reader)*100);
    fprintf(stdout,"hr = %f %% of Hinitial-b\n", (shootreader2-
>hr_reader)*100);
    ovsb1=(timesp->b-(shootreader2->above_b))*1000;
    ovsb2=((shootreader2->above)-Hmax)*1000;
    if (ovsb1>0)
        fprintf(stdout,"hf = %f millivolts \n", ovsb1);
    else
        fprintf(stdout,"hf = 0 millivolts \n");
    if (ovsb2>0)
        fprintf(stdout,"hr = %f millivolts \n", ovsb2);
    else
        fprintf(stdout,"hr = 0 millivolts \n");
    }
    break;
}
}

// Main Function
int main (int argc, char *argv[])
{
    char type;
    int rate;
    char voltstr[25];          // intermediate char array to modify the voltage
values
    char timestr[25];        // intermediate char array to modify the time
values
    double snum=0;
    double tnum=0;
    double t=0;
    int filterlength=0;
    double Hmax=0;
    double Hmin=0;
    double Hmax2=0;
    double Hmin2=0;
    FILE *pointfile=NULL;
    FILE *input_u2=NULL;
    FILE *poutput=NULL;
    double m=0.0;
    int length=0;
    double val=0;
    int posval=0;
    int negval=0;
    double tini=0;
    double tfin=0;
    int samples=0;
    int out_i=0;
    int length_total=0;
    int sample_ini=0;
    int sample_end=0;

```

```

int flag_cut=0;
int samplesp=0;
int fi=0;           // Filter generic index
double b1=0;       // Filter parameters
double b2=0;
double b3=0;
double b4=0;
double b5=0;
double a1=0;
double a2=0;
double a3=0;
double a4=0;
double a5=0;
double freq1=0;
double freq2=0;
double as[5]={0};
double bs[5]={0};
double t0=0;
double tlast=0;
int lineskip=0;

double *voutput=malloc (sizeof(double)*MAX_SAMPLES);
double *toutput=malloc (sizeof(double)*MAX_SAMPLES);
double *envelope=malloc (sizeof(double)*MAX_SAMPLES);
double *vfilter=malloc (sizeof(double)*MAX_SAMPLES);
double *tfilter=malloc (sizeof(double)*MAX_SAMPLES);
TIMES *timesp=(TIMES *)malloc(sizeof(TIMES));
TIMES *timesp2=(TIMES *)malloc(sizeof(TIMES));
SHOOTREADER *shootreader2=(SHOOTREADER *)malloc(sizeof(SHOOTREADER));

if (voutput!=NULL && toutput!=NULL && envelope !=NULL && vfilter!=NULL &&
tfilter!=NULL && timesp!=NULL && timesp2!=NULL && shootreader2!=NULL)
{
    memset(voutput, 0, MAX_SAMPLES);
    memset(toutput, 0, MAX_SAMPLES);
    memset(envelope, 0, MAX_SAMPLES);
    memset(vfilter, 0, MAX_SAMPLES);
    memset(tfilter, 0, MAX_SAMPLES);

    type=*argv[1];
    rate=atoi(argv[2]);
    if ((type!='A' && type!='B') || (rate!=106 && rate!=212 && rate!=424 &&
rate!=848))
    {
        fprintf(stdout, "Wrong Type (A or B) or Bitrate (106, 212, 424, 848)");
    }
    else
    {
        pointfile=fopen(argv[3],"r");
        input_u2=fopen("intermediate.txt","w");           // modified-
intermediate amplitude vector

        if(pointfile!=NULL && input_u2!=NULL)
        {
            //1. LOAD DATA + CHECKING DATA (WITHOUT FILTER)
            for (lineskip=0; lineskip<10; lineskip++)           // Skips the first 10
lines which are the header of csv files
            {
                skip_line (pointfile);
            }
            read_line (pointfile,voltstr, timestr);
        }
    }
}

```

```

        t0=atof(timestr);
        while (!feof(pointfile)) // We are reading the lines
of the voltage input file
        {
            if (voltstr[0]!='\0')
            {
                snum=atof(voltstr);
                tnum=atof(timestr);
                if(snum<0)
                    negval++;
                else
                    posval++;
                vfilter[samplesp]=snum;
                tfilter[samplesp]=tnum;
                samplesp++;
                read_line (pointfile,voltstr, timestr);
            }
            tlast=tfilter[samplesp-1];
        }
        samplesp=samplesp+3;

        samplesp=datacheck(posval, negval, samplesp, tlast, pointfile);

        tlast=tfilter[samplesp];

        //2. DATA FILTER 10 MHz BANDPASS
        freq1=8.56e6/(1/(2*((tlast-t0)/(samplesp-1))));
        freq2=18.56e6/(1/(2*((tlast-t0)/(samplesp-1))));
        butterworth_coeffs(freq1, freq2, as, bs);

        b1=bs[0];
        b2=bs[1];
        b3=bs[2];
        b4=bs[3];
        b5=bs[4];

        a1=as[0];
        a2=as[1];
        a3=as[2];
        a4=as[3];
        a5=as[4];

        for (fi=0; fi<samplesp; fi++)
        {
            if (fi<7 || fi>samplesp-7)
                voutput[fi]=0;
            else
                voutput[fi]=(b1*vfilter[fi]+b2*vfilter[fi-1]+b3*vfilter[fi-2]+b4*vfilter[fi-3]+b5*vfilter[fi-4]-a2*voutput[fi-1]-a3*voutput[fi-2]-a4*voutput[fi-3]-a5*voutput[fi-4])/a1;
        }

        rewind (pointfile);
        lineskip=0;
        for (lineskip=0; lineskip<10; lineskip++) // Skips the first 10
lines which are the header of csv files
        {
            skip_line (pointfile);
        }
        for (fi=0; fi<(samplesp-7); fi++) /* We are reading the lines
of the voltage input file */

```

```

    {
        val=voutput[fi];
        read_line (pointfile,voltstr,timestr);
        fprintf(input_u2,"%s,%f\n",timestr,val);
        length++;
    }

//3. HILBERT TRANSFORM AND THE COMPLEX ENVELOPE
rewind(input_u2);
hilbert("intermediate.txt");          // performs Hilbert transform

poutput=fopen("output.txt","r");      // Hilbert transform output
vector

read_line (poutput,voltstr,timestr);
tini=atof(timestr);
rewind (poutput);

if(poutput!=NULL)
{
    while (!feof(poutput))           // We are reading the lines of the
voltage input file */
    {
        read_line (poutput,voltstr,timestr);
        if (timestr[0]!='\0')
        {
            snum=atof(voltstr);
            voutput[samples]=snum;
            t=atof(timestr);
            toutput[samples]=t;
            samples++;//==>US // Same variable as the one in Hmaxfinder
            tfin=t;
        }
    }
}
else
    fprintf(stdout,"Error in Hilbert transform\n");

//4. USING A SMOOTHING FILTER (MOV. AVG) TO REDUCE THE NOISE
filterlength=1;
length_total=envfilt(voutput, toutput, filterlength, tini, tfin,
samples, envelope);

//5. 100% OF H_INITIAL
Hmaxfinder(envelope, &Hmax, &Hmin, length_total);

//6. COMPUTING THE ISO BASED TIMES
tfinder(type,envelope,toutput,tini,Hmax,Hmin,rate,length_total,timesp);

//7. CHECKING FOR ISO DEFINED MONOTONY
monocheck(envelope, toutput, Hmax, timesp, rate, type);

out_i=0;
while (out_i<MAX_SAMPLES)           // Finds how many zeros are at the
beginning of vector envelope
{
    if (envelope[out_i]==0 && flag_cut==0)
    {
        sample_ini=out_i;
        tini=toutput[sample_ini+1];
    }
}

```

```

    }

    if (envelope[out_i]!=0)
    {
        flag_cut=1;
        sample_end=out_i;
        tfin=toutput[sample_end];
    }
    out_i++;
}

samples=sample_end-sample_ini-1;          //==>US

for (out_i=0; out_i<samples; out_i++)
{
    voutput[out_i]=envelope[out_i+sample_ini+1];
    toutput[out_i]=toutput[out_i+sample_ini+1];
}
for (out_i=samples+1; out_i<MAX_SAMPLES; out_i++)
{
    voutput[out_i]=0.0;
    toutput[out_i]=0.0;
}

tini=toutput[0];
tfin=toutput[samples];

//8. OVERSHOOT OF THE READER
fprintf (stdout, "\n");    // 2nd set of functions, "New Line" printed
for debug purposes
filterlength=3;
length_total=envfilt(voutput, toutput, filterlength, tini, tfin,
samples, envelope);    // 2nd Filtering to find the alternate envelope
Hmaxfinder(envelope, &Hmax2, &Hmin2, length_total);
tfinder(type, envelope, toutput, tini, Hmax2, Hmin2, rate, length_total,
timesp2);
monocheck(envelope, toutput, Hmax2, timesp2, rate, type);
// The parameters of the alternate envelope are calculated
overshoot(timesp2, Hmax2, envelope, toutput, rate, type, samples,
shootreader2);    // This time the over- and undershoots are found

//9. MODULATION
m=modulation(type, Hmax, timesp->b);

//10. DISPLAY
display(type, rate, shootreader2, timesp, Hmax, m);
}
else if (pointfile==NULL || input_u2!=NULL)
    fprintf(stdout, "file(s) could not be opened \n");

    fclose(pointfile);
    fclose(input_u2);
}
}

else
    fprintf(stdout, "Memory could not be allocated");

free (voutput);
free (toutput);
free (envelope);

```

```
free (vfilter);  
free (tfilter);  
free (timesp);  
free (timesp2);  
free (shootreader2);  
  
return 0;  
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 10373-6:2011

Annex F (informative)

Program for the evaluation of the spectrum

The following program written in C language gives an example for the calculation of the magnitude of the spectrum from the PICC.

```

/*****
/**** This program calculates the Fourier coefficients      ***/
/**** of load modulated voltage of a PICC according      ***/
/**** the ISO/IEC 10373-6 Test methods                   ***/
/**** The coefficients are calculated at the frequencies: ***/
/**** Carrier:      fc (=13.5600 for 13.56 MHz)          ***/
/**** Upper sideband: 14.4075 MHz                       ***/
/**** Lower sideband: 12.7125 MHz                       ***/
/****
/**** Input:                                             ***/
/**** File in CSV Format containing a table of two       ***/
/**** columns (time and test PCD output voltage vd, clause 7) ***/
/****
/**** data format of input-file:                         ***/
/**** ----- ***/
/**** - one data-point per line:                         ***/
/****   (time[seconds], sense-coil-voltage[volts])      ***/
/**** - contents in ASCII, no headers                   ***/
/**** - data-points shall be equidistant time          ***/
/**** - minimum sampling rate: 100 MSamples/second     ***/
/**** - modulation waveform centered                   ***/
/****   (max. tolerance: half of subcarrier cycle)     ***/
/****
/**** example for spreadsheet file (start in next line): ***/
/****   (time)      (voltage)                            ***/
/**** 3.00000e-06,1.00                               ***/
/**** 3.00200e-06,1.01                               ***/
/**** ..... ***/
/****
/**** RUN:                                             ***/
/**** "exefilename" [filename1[.csv] ... filename[.csv] ] ***/
/****
/**** ISO/IEC 10373-6 DFT CALCULATION                   ***/
/**** Program Version 2.1 Release 2008                  ***/
/****

```

```

#include <stdio.h>
#include <string.h>
#include <math.h>

#define MAX_SAMPLES 50000
#define MAX_POINTS 500
#define MAX_MOYENNE 200

double pi; /* pi=3.14.... */

/* Array for time and sense coil voltage vd */
double vtime[MAX_SAMPLES]; /* time array */
double vd[MAX_SAMPLES]; /* Array for different coil voltage */

```

```

/*****
/****   Read CSV File   Function   ****/
/****   ****/
/****   Description:   ****/
/****   This function reads the table of time and sense coil ****/
/****   voltage from a File in CSV Format ****/
/****   ****/
/****   Input: filename ****/
/****   ****/
/****   Return: Number of samples (sample Count) ****/
/****           0 if an error occurred ****/
/****   ****/
/****   Displays Statistics: ****/
/****   ****/
/****   Filename, SampleCount, Sample rate, Max/Min Voltage ****/
/****   ****/

int readcsv(char* fname)
{
    double a,b;
    double max_vd,min_vd;
    int i;
    FILE *sample_file;

    /***** Open File *****/
    if (!strchr(fname, '.')) strcat(fname, ".csv");

    if ((sample_file = fopen(fname, "r")) == NULL)
    {
        printf("Cannot open input file %s.\n",fname);
        return 0;
    }

    /***** Read CSV File *****/
    max_vd=-1e-9F;
    min_vd=-max_vd;
    i=0;

    while (!feof(sample_file))
    {
        if (i>=MAX_SAMPLES)
        {
            printf("Warning: File truncated !!!\n");
            printf("To much samples in file %s\b\n",fname);
            break;
        }
        fscanf(sample_file,"%Lf,%Lf\n", &a, &b);
        vtime[i] = a;
        vd[i] = b;
        if (vd[i]>max_vd) max_vd=vd[i];
        if (vd[i]<min_vd) min_vd=vd[i];
        i++;
    }
    fclose(sample_file);

    /***** Displays Statistics *****/
    printf("\n*****\n");

    printf("\nStatistics: \n");
}

```

```

printf(" Filename      : %s\n",fname);
printf(" Sample count: %d\n",i);
printf(" Sample rate  : %1.0f MHz\n",1e-6/(vtime[1]-vtime[0]));
printf(" Max(vd)      : %4.0f mV\n",max_vd*1000);
printf(" Min(vd)      : %4.0f mV\n",min_vd*1000);
return i;
}/***** End ReadCsv *****/

```

```

/*****/
/**   DFT : Discrete Fourier Transformation   ***/
/*****/
/**   Description:   ***/
/**   This function calculate the Fourier coefficient   ***/
/**   ***/
/**   Input: Number of samples   ***/
/**   Global Variables:   ***/
/**   ***/
/**   Displays Results:   ***/
/**   ***/
/**   Carrier coefficient   ***/
/**   Upper sideband coefficient   ***/
/**   Lower sideband coefficient   ***/
/**   ***/
/*****/

```

```

void dft(int count)
{
    double c0_real,c0_imag,c0_abs,c0_phase;
    double c1_real,c1_imag,c1_abs,c1_phase;
    double c2_real,c2_imag,c2_abs,c2_phase;
    int N_data,center,start;
    double w0,wu,wl;
    double Wb;          /* Bartlett window coefficient */

    int i,k;

    double fc;          /* add variable for carrier frequency */

    fc=13.56e6;

    w0=(double) (fc*2.0)*pi; /* carrier 13.56 MHz */
    wu=(double) (1.0+1.0/16.0)*w0; /* upper sideband 14.41 MHz */
    wl=(double) (1.0-1.0/16.0)*w0; /* lower sideband 12.71 MHz */
    c0_real=0; /* real part of the carrier Fourier coefficient */
    c0_imag=0; /* imag part of the carrier Fourier coefficient */
    c1_real=0; /* real part of the up. sideband Fourier coefficient */
    c1_imag=0; /* imag part of the up. sideband Fourier coefficient */
    c2_real=0; /* real part of the lo. sideband Fourier coefficient */
    c2_imag=0; /* imag part of the lo. sideband Fourier coefficient */

    center=(count+1)/2; /* center address */

    /***** signal selection *****/

    /* Number of samples for six subcarrier periods */

    N_data=(int) (0.5+6*16.0F/(vtime[2]-vtime[1])/fc);

    /* Note: (vtime[2]-vtime[1]) is the scope sample rate */

```



```

start=center - (int) N_data / 2;

/***** DFT *****/

for( i=0;i<=N_data-1;i++)
{
/* Bartlett window */
if ((N_data & 1) == 0)
{
/* N_data is even */
if (i < (int) N_data /2)
{
Wb=2.0F*i/(double) (N_data - 1);
}
else
{
Wb=2.0F*(N_data-i-1)/(double) (N_data - 1);
}
}
else
{
/*N_data is odd */
if (i < (int) N_data /2)
{
Wb=2.0F*i/(double) (N_data - 1);
}
else
{
Wb=2.0F-2.0F*i/(double) (N_data - 1);
}
}
}

k=i+start;

c0_real=c0_real+vd[k] * (double) cos (w0*vtime[k]) *Wb;
c0_imag=c0_imag+vd[k] * (double) sin (w0*vtime[k]) *Wb;
c1_real=c1_real+vd[k] * (double) cos (wu*vtime[k]) *Wb;
c1_imag=c1_imag+vd[k] * (double) sin (wu*vtime[k]) *Wb;
c2_real=c2_real+vd[k] * (double) cos (wl*vtime[k]) *Wb;
c2_imag=c2_imag+vd[k] * (double) sin (wl*vtime[k]) *Wb;
}

/***** DFT scale *****/

c0_real=4.0F*c0_real/(double) N_data;
c0_imag=4.0F*c0_imag/(double) N_data;
c1_real=4.0F*c1_real/(double) N_data;
c1_imag=4.0F*c1_imag/(double) N_data;
c2_real=4.0F*c2_real/(double) N_data;
c2_imag=4.0F*c2_imag/(double) N_data;
/* Note: 4.0F includes the correction coef. of the Bartlett window */

/***** absolute Fourier coefficient *****/
c0_abs=(double) sqrt (c0_real*c0_real + c0_imag*c0_imag);
c1_abs=(double) sqrt (c1_real*c1_real + c1_imag*c1_imag);
c2_abs=(double) sqrt (c2_real*c2_real+c2_imag*c2_imag);

```

```

/***** Phase of Fourier coefficient *****/
c0_phase=(double)atan2(c0_imag,c0_real);
c1_phase=(double)atan2(c1_imag,c1_real);
c2_phase=(double)atan2(c2_imag,c2_real);

/***** Result Display *****/

printf("\n\nResults: \n");
printf("Carrier ");
printf("Abs: %7.3fmV ",1000*c0_abs);
printf("Phase: %3.0fdeg\n",c0_phase/pi*180);
printf("Upper sideband ");
printf("Abs: %7.3fmV ",1000*c1_abs);
printf("Phase: %3.0fdeg\n",c1_phase/pi*180);
printf("Lower sideband ");
printf("Abs: %7.3fmV ",1000*c2_abs);
printf("Phase: %3.0fdeg\n\n",c2_phase/pi*180);
printf("\n*****\n");
return;
}/***** End DFT *****/

/*****
*** MAIN Program ***
*****/
int main(unsigned short paramCount,char *paramList[])
{
char fname[256];
unsigned int i,sample_count;

pi = (double)atan(1.0)*4; /* calculate pi */

printf("\n*****\n");
printf("\n**** ISO/IEC 10373-6 PICC Test-Program ****\n");
printf("\n**** Version: 2.1 NOVEMBER 2008 ****\n");
printf("\n**** ****\n");
printf("\n*****\n");
/***** No Input Parameter *****/
if (paramCount==1)
{
printf("\nCSV File name :");
scanf("%s",fname);
if (!strchr(fname, '.')) strcat(fname, ".csv");
if (!(sample_count=readcsv(fname))) return 0;
dft(sample_count);
}
else
{
/***** Input Parameter Loop *****/
for (i=1;i<paramCount;i++)
{
strcpy(fname,paramList[i]);

if (!strchr(fname, '.')) strcat(fname, ".csv");
if (!(sample_count=readcsv(fname))) break;
dft(sample_count);
}
}
return 0;
}/***** End Main *****/

```



Annex G (normative)

Additional PICC test methods

G.1 PICC-test-apparatus and accessories

This clause defines the test apparatus and test circuits for verifying the operation of a PICC according to ISO/IEC 14443-3:—. The test apparatus includes:

- Calibration coil (see 5.2);
- Test PCD assembly (see 5.3);
- Digital sampling oscilloscope (see 5.1.1).

Care shall be taken to ensure that the results are not affected by the RF performance of the test circuits.

G.1.1 Emulating the I/O protocol

The PICC-test-apparatus shall be able to emulate the Type A and Type B protocols, which are required to test a PICC.

G.1.2 Generating the I/O character timing in reception mode

The PICC-test-apparatus shall be able to generate the I/O bit stream according to ISO/IEC 14443-3:—. Timing parameters: start bit length, guard time, bit width, request guard time, start of frame width, end of frame width shall be configurable.

G.1.3 Measuring and monitoring the RF I/O protocol

The PICC-test-apparatus shall be able to measure and monitor the timing of the logical low and high states of the RF Input/Receive line relative to the clock frequency. The PICC-test-apparatus shall be able to monitor the PICC subcarrier.

G.1.4 Protocol Analysis

The PICC-test-apparatus shall be able to analyze the I/O-bit stream in accordance with protocol Type A and Type B as specified in ISO/IEC 14443-3:— and ISO/IEC 14443-4:2008 and extract the logical data flow for further protocol analysis.

G.1.5 RFU fields

RFU fields should be constantly monitored during the testing and shall always be verified to contain the assigned default value. A test shall fail and the tested PICC shall be declared non-compliant in case an RFU field is not set to its default value at any time.

G.1.5.1 RFU values

Functional fields should be constantly monitored during the testing and shall always be verified to contain only functional values documented in the standard or proprietary values documented in the standard. A test shall

fail and the tested PICC shall be declared non-compliant in case a functional field is not set to said values (and thus is set to an RFU or restricted value) at any time.

G.1.5.2 Timing measurements

The PICC-test-apparatus shall continuously monitor the following frame format and timing values:

For PICC Type A:

- Frame delay time PCD to PICC (see ISO/IEC 14443-3:—, 6.2.1.1);
- Frame formats (see ISO/IEC 14443-3:—, 6.2.3);
- Frame waiting time (see ISO/IEC 14443-4:2008, 7.2).

For PICC Type B:

- Character, frame format and timing (see ISO/IEC 14443-3:—, 7.1);
- Frame waiting time (see ISO/IEC 14443-4:2008, 7.2).

A test shall fail and the tested PICC be declared non-compliant in case one of the listed timing constraints is violated.

G.1.5.3 Timing measurement report

Fill out Table G.59 for PICC Type A and/or Table G.60 for PICC Type B with the measured timing values.

G.2 Relationship of test methods versus base standard requirement

Table G.1 lists the applicable tests for Type A PICCs.

Table G.2 lists the applicable tests for Type B PICCs.

Table G.3 lists the applicable tests for Type A and Type B PICCs.

The ISO/IEC 14443-4:2008 PICC should also comply with ISO/IEC 14443-3:— and should be subjected to both the part 3 and part 4 tests for the applicable Type.

A PICC compliant with ISO/IEC 14443-3:— but not with ISO/IEC 14443-4:2008 and in ACTIVE or ACTIVE* state (see G.3.3.7, G.3.3.12 and G.4.4.6) may respond with any frame (including Mute) to frames not related to ISO/IEC 14443-3:—.

Table G.1 — Test methods for logical operation of the PICC Type A protocol

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause(s)
G.3.2	Polling	ISO/IEC 14443-3:—	5
G.3.3	Testing of the PICC Type A state transitions	ISO/IEC 14443-3:—	6.3, 6.4, 6.5
G.3.4	Handling of Type A anticollision	ISO/IEC 14443-3:—	6.4.2
G.3.5	Handling of RATS	ISO/IEC 14443-4:2008	5.6.1.2
G.3.6	Handling of PPS request	ISO/IEC 14443-4:2008	5.6.2.2
G.3.7	Handling of FSD	ISO/IEC 14443-4:2008	5.1

Table G.2 — Test methods for logical operation of the PICC Type B protocol

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause(s)
G.4.2	Polling	ISO/IEC 14443-3:—	5
G.4.3	PICC Reception	ISO/IEC 14443-3:—	7.1
G.4.4	Testing of the PICC Type B state transitions	ISO/IEC 14443-3:—	7.4 – 7.12
G.4.5	Handling of Type B anticollision	ISO/IEC 14443-3:—	7.4 – 7.12
G.4.6	Handling of ATTRIB	ISO/IEC 14443-3:—	7.10
G.4.7	Handling of Maximum Frame Size	ISO/IEC 14443-3:—	7.10.4

Table G.3 — Test methods for logical operation of PICC Type A or B

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause(s)
G.5.2	PICC reaction to ISO/IEC 14443-4 Scenarios	ISO/IEC 14443-4:2008	7
G.5.3	Handling of PICC error detection	ISO/IEC 14443-4:2008	7.5.6
G.5.4	PICC reaction on CID	ISO/IEC 14443-4:2008	7.1.1.2
G.5.5	PICC reaction on NAD	ISO/IEC 14443-4:2008	7.1.1.3

G.3 Test method for initialization of the PICC Type A

G.3.1 Introduction

The tests in this chapter determine whether a PICC Type A conforms to the ISO/IEC 14443-3:— standard and the activation sequence of ISO/IEC 14443-4:2008, Clause 5. If compliance with ISO/IEC 14443-4:2008 is not required, all tests containing ISO/IEC 14443-4 commands need not be applied.

G.3.2 Scenario G.1: Polling

G.3.2.1 Scope

This test is to determine the behavior of the PICC Type A on receiving REQA commands according to ISO/IEC 14443-3:—, Clause 5.

G.3.2.2 Procedure

Perform the following steps for 3 different operating fields of 1,5, 4,5 and 7,5 A/m (rms):

- a) Place the PICC into the field and adjust it.
- b) Switch the RF operating field off for a minimum time for resetting a PICC (see ISO/IEC 14443-3:—, 5.4).
- c) Switch the RF operating field on.
- d) Do delay of 5 ms and send a valid REQA command frame.
- e) Record the presence and the content of the PICC response.
- f) Switch the RF operating field off for a minimum time for resetting a PICC (see ISO/IEC 14443-3:—, 5.4).

- g) Switch the RF operating field on.
- h) Wait 5 ms and send a valid REQB command frame (using Type B modulation and bit coding).
- i) Wait 5 ms and send a valid REQA command frame.
- j) Record the presence and the content of the PICC response.

G.3.2.3 Test report

Fill the appropriate row in Table G.61 according to Table G.4.

Table G.4 — Result criteria for Scenario G.1: Polling

Explanation	Test result
Only when the PICC's response is a valid ATQA in both steps e) and j)	PASS
If the PICC's response isn't a valid ATQA in any of steps e) or j)	FAIL

G.3.3 Testing of the PICC Type A state transitions

G.3.3.1 Scope

These tests verify the correct implementation of the PICC Type A state diagram as described in ISO/IEC 14443-3:—, 6.3.

G.3.3.2 General test outline

For an exhaustive test of the PICC Type A state machine the correctness of every possible state transition at every state shall be verified. Verifying a specific state using a specific state transition will be done as follows:

First, reset the PICC and place it in the test initial state (TIS). This is one of the states from StateSet where the transitions (T) have to be verified. Then execute a transition (T) from TransitionSet. After execution of the state transition, check if the PICC is in the expected target state TTS. There is a difficulty in how to perform this check, because it is impossible to directly inspect the state machine of the PICC. The solution to this problem is to make some additional state transitions and checking the answer of the PICC. The transitions for this purpose are selected in such way that the state can be determined from the PICC answers as precisely as possible.

G.3.3.2.1 Functions for putting the PICC in the Test Initial State (TIS)

Putting the PICC into the State TIS will be done by a sequence of transition commands specified in the following table. The general method is as follows:

In order to put the PICC into State TIS, lookup the corresponding state transition sequence in Table G.5. Then successively apply the state transitions described in the State Transition Sequence column by looking up the corresponding commands in Table G.6. Always check the content and integrity of the PICC response.

Table G.5 — State Transition Sequence Table

TIS	State Transition Sequence
POWER-OFF	---
IDLE	POWER-OFF → IDLE
READY(1)	POWER-OFF → IDLE → READY(1)
READY(2)	POWER-OFF → IDLE → READY(1) → READY(2)
READY(3)	POWER-OFF → IDLE → READY(1) → READY(2) → READY(3)
ACTIVE	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE
PROTOCOL	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → PROTOCOL
HALT	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → HALT
READY*(1)	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → HALT → READY*(1)
READY*(2)	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → HALT → READY*(1) → READY*(2)
READY*(3)	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → HALT → READY*(1) → READY*(2) → READY*(3)
ACTIVE*	POWER-OFF → IDLE → READY(1) → ... → READY(CascadeLevels) → ACTIVE → HALT → READY*(1) → ... → READY*(CascadeLevels) → ACTIVE*

Table G.6 — State Transition Table

State → Next State	PICC-test-apparatus	PICC
POWER-OFF → IDLE	Power On (RF Field on) →	
	←	Mute
IDLE → READY(1)	REQA →	
	←	ATQA
READY(1) → READY(2)	SELECT(1) ^a →	
	←	SAK (cascade)
READY(2) → READY(3)	SELECT(2) ^a →	
	←	SAK (cascade)
READY(CascadeLevels) → ACTIVE	SELECT (CascadeLevels) ^a →	
	←	SAK (complete)
ACTIVE → PROTOCOL	RATS(0,0) →	
	←	ATS
ACTIVE → HALT	HLTA →	
	←	Mute
HALT → READY*(1)	WUPA →	
	←	ATQA
READY*(1) → READY*(2)	SELECT(1) →	
	←	SAK (cascade)

State → Next State	PICC-test-apparatus	PICC
READY*(2) → READY*(3)	SELECT(2) → ←	SAK(cascade)
READY*(CascadeLevels) → ACTIVE*	SELECT (CascadeLevels) → ←	SAK (complete)

^a Any SELECT command may be preceded with an anticollision command to retrieve the PICC UID, especially for random UID.

G.3.3.2.2 Functions for checking the validity of the test target state TTS

Table G.7 describes the state transitions, which are used to check whether the PICC is in the state S. The content of the PICC answer (i.e. ATQA, SAK ...) should be thoroughly checked for ISO/IEC 14443-3:— and ISO/IEC 14443-4:2008 conformance. Note, that these tests may cause the PICC to change state.

The READY(I)/ READY*(I) states and the ACTIVE/ACTIVE* states cannot be distinguished with one test run. In order to distinguish the "*" -states from the non-"*" -states perform the following steps:

- a) Rerun the test a second time, without checking the TTS.
- b) Send REQA command. The PICC response shall be Mute.
- c) Send REQA command.
- d) If the PICC response is Mute then the PICC state was a "*" -state.
- e) Else the PICC was a non-"*" -state.

The HALT state cannot be distinguished from READY*(I) state and from ACTIVE* state with one test run. In order to distinguish the HALT state perform the following steps:

- f) Rerun the test a second time, without checking the TTS.
- g) Send WUPA command. The PICC response shall be ATQA.

Table G.7 — Checking the TTS

State S	PICC-test-apparatus	PICC
IDLE	REQA → ←	ATQA
READY(I), I < CascadeLevels	SELECT(I) ^a → ←	SAK (cascade)
READY(I), I = CascadeLevels	SELECT(I) ^a → ←	SAK (complete)
ACTIVE	RATS (0,0) → ←	ATS
PROTOCOL	I(0) ₀ (TEST_COMMAND1(1)) → ←	I(0) ₀ (TEST_RESPONSE1(1))

State S	PICC-test-apparatus	PICC
HALT	REQA	→
		←
	WUPA	→
		←
READY*(I), I < CascadeLevels	SELECT (I)	→
		←
READY*(I), I = CascadeLevels	SELECT (I)	→
		←
ACTIVE*	RATS(0,0)	→
		←
^a Any SELECT command may be preceded with an anticollision command to retrieve the PICC UID, especially for random UID.		

G.3.3.3 Scenario G.2: Behavior of the PICC Type A in the IDLE state

G.3.3.3.1 Scope

This test is to determine the behavior of the PICC Type A in the IDLE state according to ISO/IEC 14443-3:—, 6.3.2.

G.3.3.3.2 Procedure

Perform the following steps for every row of Table G.8:

- Put the PICC into IDLE state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- Check if the PICC is in the state TTS.

Table G.8 — Transitions from IDLE state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA	→	1172/ <i>fc</i>	READY(1)
		←		
WUPA	WUPA	→	1236/ <i>fc</i>	READY(1)
		←		
HLTA	HLTA	→		IDLE
		←		

Transition	PICC-test-apparatus	PICC	FDT	TTS
AC ^c	(‘93’ NVB UIDTX _i [[1..n ₁]]) ^a	→ ← Mute		IDLE
nAC ^d	(‘93’ NVB ~UIDTX _i [[1.. n ₁]]) ^a	→ ← Mute		IDLE
SELECT ^c	SELECT(1)	→ ← Mute		IDLE
nSELECT ^d	(‘93 70’ ~UIDTX _i [[1..32]] BCC CRC_A)	→ ← Mute		IDLE
RATS	RATS(0,0)	→ ← Mute		IDLE
PPS	PPS(0,0,0)	→ ← Mute		IDLE
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1))	→ ← Mute		IDLE
DESELECT	S(DESELECT)	→ ← Mute		IDLE
Error condition	(‘26’) ^b	→ ← Mute		IDLE

^a Let $1 \leq n_1 \leq 32$.

^b The value is sent in a standard frame and not in a short frame.

^c This test is skipped for PICCs exploiting random UID.

^d For PICCs exploiting a random UID perform this test with a fixed arbitrary UID.

G.3.3.3.3 Test report

Fill the appropriate row in Table G.61 according to Table G.9.

Table G.9 — Result criteria for behavior of the PICC Type A in the IDLE state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.4 Scenario G.3: Behavior of the PICC Type A in the READY(1) state

G.3.3.4.1 Scope

This test is to determine the behavior of the PICC Type A in the READY state on cascade level 1 according to ISO/IEC 14443-3:—, 6.3.3.

G.3.3.4.2 Procedure

Perform the following steps for all PICCs and every row of Table G.10:

- Put the PICC into READY(1) state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- Check if the PICC is in the state TTS.

Table G.10 — Transitions from READY(1) state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		IDLE
WUPA	WUPA → ←	Mute		IDLE
HLTA	HLTA → ←	Mute		IDLE
AC ^g (split after (0)b)	('93' NVB UIDTX ₁ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₁ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY(1)
AC ^g (split after (1)b)	('93' NVB UIDTX ₁ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₁ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY(1)
nAC ^g (wrong UID)	('93' NVB ~UIDTX ₁ [[1..n ₃]]) ^f → ←	Mute		IDLE
SELECT ^g	SELECT(1) → ←	SAK ^d	FDT ^c	TTS ^e
nSELECT ^g (wrong UID)	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute		IDLE
Error condition	('93 70' UIDTX ₁ BCC ~CRC_A) → ←	Mute		IDLE
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		IDLE
DESELECT	S(DESELECT) → ←	Mute		IDLE

Transition	PICC-test-apparatus	PICC	FDT	TTS
RATS	RATS(0,0) →	Mute		IDLE
	←			
PPS	PPS(0,0,0) →	Mute		IDLE
	←			

^a Let $1 \leq n_1 \leq 32$, UIDTX₁[[n₁]] = 0. If such a number does not exist, the test may be skipped.

^b Let $1 \leq n_2 \leq 32$, UIDTX₁[[n₂]] = 1. If such a number does not exist, the test may be skipped.

^c FDT is $1172/f_c$ (~86,43 μs) if last bit = (0)b and $1236/f_c$ (~91,15 μs) if last bit = (1)b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).

^d Cascade bit of SAK shall be (0)b for single size UID PICCs and (1)b for double and triple size UID PICCs.

^e Single size UID PICC shall be in ACTIVE state; double and triple size UID PICCs shall be in READY state.

^f Let $1 \leq n_3 \leq 32$.

^g Any AC or SELECT command may be preceded with an anticollision command to retrieve the PICC UID, especially for random UID.

G.3.3.4.3 Test report

Fill the appropriate row in Table G.61 according to Table G.11.

Table G.11 — Result criteria for behavior of the PICC Type A in the READY(1) state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.5 Scenario G.4: Behavior of the PICC Type A in the READY(2) state

G.3.3.5.1 Scope

This test is to determine the behavior of the PICC Type A in the READY state on cascade level 2 according to ISO/IEC 14443-3:—, 6.3.3. This test is only for PICCs with double or triple size UID.

G.3.3.5.2 Procedure

Perform the following steps for all PICCs with double and triple size UID and every row of Table G.12:

- a) Put the PICC into READY(2) state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.12 — Transitions from READY(2) state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		IDLE
WUPA	WUPA → ←	Mute		IDLE
HLTA	HLTA → ←	Mute		IDLE
AC ^g (split after (0)b)	('95' NVB UIDTX ₂ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₂ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY(2)
AC ^g (split after (1)b)	('95' NVB UIDTX ₂ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₂ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY(2)
nAC ^g (wrong UID)	('95' NVB ~UIDTX ₂ [[1..n ₃]]) ^f → ←	Mute		IDLE
SELECT ^g	SELECT(2) → ←	SAK ^d	FDT ^c	TTS ^e
nSELECT ^g (wrong UID)	('95 70' ~UIDTX ₂ BCC CRC_A) → ←	Mute		IDLE
Error condition	('95 70' UIDTX ₂ BCC ~CRC_A) → ←	Mute		IDLE
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		IDLE
DESELECT	S(DESELECT) → ←	Mute		IDLE
RATS	RATS(0,0) → ←	Mute		IDLE
PPS	PPS(0,0,0) → ←	Mute		IDLE

^a Let $1 \leq n_1 \leq 32$, UIDTX₂[[n₁]] = 0. If such a number does not exist, the test may be skipped.

^b Let $1 \leq n_2 \leq 32$, UIDTX₂[[n₂]] = 1. If such a number does not exist, the test may be skipped.

^c FDT is 1172/fc (~86,43 μs) if last bit = (0)b and 1236/fc (~91,15 μs) if last bit = (1)b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).

^d Cascade bit of SAK shall be (0)b for double size UID PICCs and (1)b for triple size UID PICCs.

^e Double size UID PICCs shall be in ACTIVE state; triple size UID PICCs shall be in READY state.

^f Let $1 \leq n_3 \leq 32$.

^g Any AC or SELECT command may be preceded with an anticollision command to retrieve the PICC UID, especially for random UID.

G.3.3.5.3 Test report

Fill the appropriate row in Table G.61 according to Table G.13.

Table G.13 — Result criteria for behavior of the PICC Type A in the READY(2) state

Explanation	Test result
If the PICC has a single size UID	Not applicable (N/A)
When the PICC has a double or triple size UID and only when it responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.6 Scenario G.5: Behavior of the PICC Type A in the READY(3) state

G.3.3.6.1 Scope

This test is to determine the behavior of the PICC Type A in the READY state according to ISO/IEC 14443-3:—, 6.3.3. This test is only for PICCs with triple size UID.

G.3.3.6.2 Procedure

Perform the following steps for all PICCs with triple size UID and every row of Table G.14:

- a) Put the PICC into READY(3) state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.14 — Transitions from READY(3) state

Transitions	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		IDLE
WUPA	WUPA → ←	Mute		IDLE
HLTA	HLTA → ←	Mute		IDLE
AC ^e (split after (0)b)	('97' NVB UIDTX ₃ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₃ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY(3)
AC ^e (split after (1)b)	('97' NVB UIDTX ₃ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₃ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY(3)
nAC ^e (wrong UID)	('97' NVB ~UIDTX ₃ [[1..n ₃]]) ^d → ←	Mute		IDLE
SELECT ^e	SELECT(3) → ←	SAK (complete)	FDT ^c	ACTIVE
nSELECT ^e (wrong UID)	('97 70' ~UIDTX ₃ BCC CRC_A) → ←	Mute		IDLE
Error condition	('97 70' UIDTX ₃ BCC ~CRC_A) → ←	Mute		IDLE
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		IDLE
DESELECT	S(DESELECT) → ←	Mute		IDLE
RATS	RATS(0,0) → ←	Mute		IDLE
PPS	PPS(0,0,0) → ←	Mute		IDLE
<p>^a Let $1 \leq n_1 \leq 32$, UIDTX₃[[n₁]] = 0. If such a number does not exist, the test may be skipped.</p> <p>^b Let $1 \leq n_2 \leq 32$, UIDTX₃[[n₂]] = 1. If such a number does not exist, the test may be skipped.</p> <p>^c FDT is 1172/fc (~86,43 μs) if last bit = (0)b and 1236/fc (~91,15 μs) if last bit = (1)b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).</p> <p>^d Let $1 \leq n_3 \leq 32$.</p> <p>^e Any AC or SELECT command may be preceded with an anticollision command to retrieve the PICC UID, especially for random UID.</p>				

G.3.3.6.3 Test report

Fill the appropriate row in Table G.61 according to Table G.15.

Table G.15 — Result criteria for behavior of the PICC Type A in the READY(3) state

Explanation	Test result
If the PICC has a single or double size UID	Not applicable (N/A)
When the PICC has a triple size UID and only when it responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.7 Scenario G.6: Behavior of the PICC Type A in the ACTIVE state

G.3.3.7.1 Scope

This test is to determine the behavior of the PICC Type A in the ACTIVE state according to ISO/IEC 14443-3:—, 6.3.4.

G.3.3.7.2 Procedure

Perform the following steps for every row of Table G.16:

- a) Put the PICC into ACTIVE state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC is as indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.16 — Transitions from ACTIVE state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA →			IDLE
	←	Mute		
WUPA	WUPA →			IDLE
	←	Mute		
AC	('93' NVB UIDTX _i [[1..n _i]]) ^a →			IDLE
	←	Mute ^b		
nAC	('93' NVB ~UIDTX _i [[1..n _i]]) ^a →			IDLE
	←	Mute ^b		
HLTA	HLTA →			HALT
	←	Mute		

Transition	PICC-test-apparatus	PICC	FDT	TTS
SELECT	SELECT(1) → ←	Mute ^b		IDLE
nSELECT	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute ^b		IDLE
RATS	RATS(0,0) → ←	ATS	< 65536/fc	PROTOCOL
Error condition	('E0 00' ~CRC_A) → ←	Mute ^b		IDLE
ISO/IEC 14443-4 command	I(0) _n (TEST_COMMAND1(1)) → ←	Mute ^b		IDLE
DESELECT	S(DESELECT) → ←	Mute ^b		IDLE
PPS	PPS(0,0,0) → ←	Mute ^b		IDLE
^a Let $1 \leq n_1 \leq 32$. ^b Mute or proprietary response.				

G.3.3.7.3 Test report

Fill the appropriate row in Table G.61 according to Table G.17.

Table G.17 — Result criteria for behavior of the PICC Type A in the ACTIVE state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.8 Scenario G.7: Behavior of the PICC Type A in the HALT state

G.3.3.8.1 Scope

This test is to determine the behavior of the PICC Type A in the HALT state according to ISO/IEC 14443-3:—, 6.3.5.

G.3.3.8.2 Procedure

Perform the following steps for every row of Table G.18:

- Put the PICC into HALT state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.

- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC is as indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.18 — Transitions from HALT state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		HALT
WUPA	WUPA → ←	ATQA	1236/fc	READY*(1)
HLTA	HLTA → ←	Mute		HALT
AC	('93' NVB UIDTX ₁ [[1..n ₁]]) ^a → ←	Mute		HALT
nAC	('93' NVB ~UIDTX ₁ [[1..n ₁]]) ^a → ←	Mute		HALT
SELECT	SELECT(1) → ←	Mute		HALT
nSELECT	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute		HALT
RATS	RATS(0,0) → ←	Mute		HALT
Error condition	('52') in the standard frame → ←	Mute		HALT
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		HALT
DESELECT	S(DESELECT) → ←	Mute		HALT
PPS	PPS(0,0,0) → ←	Mute		HALT
^a Let $1 \leq n_1 \leq 32$.				

G.3.3.8.3 Test report

Fill the appropriate row in Table G.61 according to Table G.19.

Table G.19 — Result criteria for behavior of the PICC Type A in the HALT state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.9 Scenario G.8: Behavior of the PICC Type A in the READY*(1) state**G.3.3.9.1 Scope**

This test is to determine the behavior of the PICC Type A in the READY* state of cascade level 1 according to ISO/IEC 14443-3:—, 6.3.6.

G.3.3.9.2 Procedure

Perform the following steps for every row of Table G.20:

- Put the PICC into READY*(1) state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- Check that the PICC is in the state TTS.

Table G.20 — Transitions from READY*(1) state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		HALT
WUPA	WUPA → ←	Mute		HALT
HLTA	HLTA → ←	Mute		HALT
AC (split after (0)b)	('93' NVB UIDTX ₁ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₁ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY*(1)
AC (split after (1)b)	('93' NVB UIDTX ₁ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₁ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY*(1)
nAC (wrong UID)	('93' NVB ~UIDTX ₁ [[1..n ₃]]) ^f → ←	Mute		HALT

Transition	PICC-test-apparatus	PICC	FDT	TTS
SELECT	SELECT(1) → ←	SAK ^d	FDT ^c	TTS ^e
nSELECT (wrong UID)	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute		HALT
Error condition	('93 70' UIDTX ₁ BCC ~CRC_A) → ←	Mute		HALT
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		HALT
DESELECT	S(DESELECT) → ←	Mute		HALT
RATS	RATS(0,0) → ←	Mute		HALT
PPS	PPS(0,0,0) → ←	Mute		HALT

^a Let $1 \leq n_1 \leq 32$, UIDTX₁[[n₁]] = 0. If such a number does not exist, the test may be skipped.

^b Let $1 \leq n_2 \leq 32$, UIDTX₁[[n₂]] = 1. If such a number does not exist, the test may be skipped.

^c FDT is $1172/f_c$ (~86,43 μs) if last bit = (0)_b and $1236/f_c$ (~91,15 μs) if last bit = (1)_b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).

^d Cascade bit of SAK shall be (0)_b for single size UID PICCs and (1)_b for double and triple size UID PICCs.

^e Single size UID PICCs shall be in ACTIVE state; double and triple size UID PICCs should be in READY state.

^f Let $1 \leq n_3 \leq 32$.

G.3.3.9.3 Test report

Fill the appropriate row in Table G.61 according to Table G.21.

Table G.21 — Result criteria for behavior of the PICC Type A in the READY*(1) state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.10 Scenario G.9: Behavior of the PICC Type A in the READY*(2) state

G.3.3.10.1 Scope

This test is to determine the behavior of the PICC Type A in the READY* state of cascade level 2 according to ISO/IEC 14443-3:—, 6.3.6. This test only applies to PICCs with double or triple size UID.

G.3.3.10.2 Procedure

Perform the following steps for every row of Table G.22:

- Put the PICC into READY*(2) state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- Check if the PICC is in the state TTS.

Table G.22 — Transitions from READY*(2) state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		HALT
WUPA	WUPA → ←	Mute		HALT
HLTA	HLTA → ←	Mute		HALT
AC (split after (0)b)	('95' NVB UIDTX ₂ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₂ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY*(2)
AC (split after (1)b)	('95' NVB UIDTX ₂ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₂ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY*(2)
nAC (wrong UID)	('95' NVB ~UIDTX ₂ [[1..n ₃]]) ^f → ←	Mute		HALT
SELECT	SELECT(2) → ←	SAK ^d	FDT ^c	TTS ^e
nSELECT (wrong UID)	('95 70' ~UIDTX ₂ BCC CRC_A) → ←	Mute		HALT
Error condition	('95 70' UIDTX ₂ BCC ~CRC_A) → ←	Mute		HALT
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		HALT
DESELECT	S(DESELECT) → ←	Mute		HALT

Transition	PICC-test-apparatus	PICC	FDT	TTS
RATS	RATS(0,0) →	Mute		HALT
	←			
PPS	PPS(0,0,0) →	Mute		HALT
	←			
<p>a Let $1 \leq n_1 \leq 32$, UIDTX₂[[n₁]] = 0. If such a number does not exist, the test may be skipped.</p> <p>b Let $1 \leq n_2 \leq 32$, UIDTX₂[[n₂]] = 1. If such a number does not exist, the test may be skipped.</p> <p>c FDT is 1172/fc (~86,43 μs) if last bit = (0)b and 1236/fc (~91,15 μs) if last bit = (1)b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).</p> <p>d Cascade bit of SAK shall be (0)b for double size UID PICCs and (1)b for triple size UID PICCs.</p> <p>e Double size UID PICCs shall be in ACTIVE state; triple size UID PICCs shall be in READY state.</p> <p>f Let $1 \leq n_3 \leq 32$.</p>				

G.3.3.10.3 Test report

Fill the appropriate row in Table G.61 according to Table G.23.

Table G.23 — Result criteria for behavior of the PICC Type A in the READY*(2) state

Explanation	Test result
If the PICC has a single size UID	Not applicable (N/A)
When the PICC has a double or triple size UID and only when if responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.11 Scenario G.10: Behavior of the PICC Type A in the READY*(3) state

G.3.3.11.1 Scope

This test is to determine the behavior of the PICC Type A in the READY* state of cascade level 3 according to ISO/IEC 14443-3:—, 6.3.6. This test is only for PICCs with triple size UID.

G.3.3.11.2 Procedure

Perform the following steps for every row of Table G.24:

- a) Put the PICC into READY*(3) state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC conforms with the value indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.24 — Transitions from READY*(3) state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		HALT
WUPA	WUPA → ←	Mute		HALT
HLTA	HLTA → ←	Mute		HALT
AC (split after (0)b)	('97' NVB UIDTX ₃ [[1..n ₁]]) ^a → ←	if n ₁ = 32 then (BCC) else (UIDTX ₃ [[n ₁ +1..32]] BCC) ^a	1172/fc	READY*(3)
AC (split after (1)b)	('97' NVB UIDTX ₃ [[1..n ₂]]) ^b → ←	if n ₂ = 32 then (BCC) else (UIDTX ₃ [[n ₂ +1..32]] BCC) ^b	1236/fc	READY*(3)
nAC (wrong UID)	('97' NVB ~UIDTX ₃ [[1..n ₃]]) ^d → ←	Mute		HALT
SELECT	SELECT(3) → ←	SAK (complete)	FDT ^c	ACTIVE*
nSELECT (wrong UID)	('97 70' ~UIDTX ₃ BCC CRC_A) → ←	Mute		HALT
Error condition	('97 70' UIDTX ₃ BCC ~CRC_A) → ←	Mute		HALT
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute		HALT
DESELECT	S(DESELECT) → ←	Mute		HALT
RATS	RATS(0,0) → ←	Mute		HALT
PPS	PPS(0,0,0) → ←	Mute		HALT

^a Let $1 \leq n_1 \leq 32$, UIDTX₃[[n₁]] = 0. If such a number does not exist, the test may be skipped.

^b Let $1 \leq n_2 \leq 32$, UIDTX₃[[n₂]] = 1. If such a number does not exist, the test may be skipped.

^c FDT is 1172/fc (~86,43 μs) if last bit = (0)b and 1236/fc (~91,15 μs) if last bit = (1)b (see margin in ISO/IEC 14443-3:—, 6.2.1.1).

^d Let $1 \leq n_3 \leq 32$.

G.3.3.11.3 Test report

Fill the appropriate row in Table G.61 according to Table G.25.

Table G.25 — Result criteria for behavior of the PICC Type A in the READY*(3) state

Explanation	Test result
If the PICC has a single or double size UID	Not applicable (N/A)
When the PICC has a triple size UID and only when it responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.12 Scenario G.11: Behavior of the PICC Type A in the ACTIVE* state

G.3.3.12.1 Scope

This test is to determine the behavior of the PICC Type A in the ACTIVE* state according to ISO/IEC 14443-3:—, 6.3.7.

G.3.3.12.2 Procedure

Perform the following steps for every row of Table G.26:

- a) Put the PICC into ACTIVE state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) If the PICC response is not Mute, check that the Frame Delay Time of the PICC is as indicated in the FDT column.
- e) Check if the PICC is in the state TTS.

Table G.26 — Transitions from ACTIVE* state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA →			HALT
	←	Mute		
WUPA	WUPA →			HALT
	←	Mute		
HLTA	HLTA →			HALT
	←	Mute		
AC	(‘93’ NVB UIDTX _i [[1..n _i]]) ^a →			HALT
	←	Mute ^b		
nAC	(‘93’ NVB ~UIDTX _i [[1..n _i]]) ^a →			HALT
	←	Mute ^b		
SELECT	SELECT(1) →			HALT
	←	Mute ^b		

Transition	PICC-test-apparatus	PICC	FDT	TTS
nSELECT	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute ^b		HALT
RATS	RATS(0,0) → ←	ATS	< 65536/fc	PROTOCOL
Error condition	('E0 00' ~CRC_A) → ←	Mute ^b		HALT
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute ^b		HALT
DESELECT	S(DESELECT) → ←	Mute ^b		HALT
PPS	PPS(0,0,0) → ←	Mute ^b		HALT
^a Let $1 \leq n_1 \leq 32$. ^b Mute or proprietary response.				

G.3.3.12.3 Test report

Fill the appropriate row in Table G.61 according to Table G.27.

Table G.27 — Result criteria for behavior of the PICC Type A in the ACTIVE* state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.3.13 Scenario G.12: Behavior of the PICC Type A in the PROTOCOL state

G.3.3.13.1 Scope

This test is to determine the behavior of the PICC Type A in the PROTOCOL state according to ISO/IEC 14443-4:2008. This test shall ensure that the activated PICC does not respond to any anticollision or initialization command.

G.3.3.13.2 Procedure

Perform the following steps for every row of Table G.28:

- Put the PICC into PROTOCOL state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- If the PICC response is not Mute, check that the Frame Delay Time of the PICC is as indicated in the FDT column.

e) Check if the PICC is in the state TTS.

Table G.28 — Transitions from PROTOCOL state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQA	REQA → ←	Mute		PROTOCOL
WUPA	WUPA → ←	Mute		PROTOCOL
AC	('93' NVB UIDTX _i [[1..n _i]]) ^a → ←	Mute		PROTOCOL
nAC	('93' NVB ~UIDTX _i [[1..n _i]]) ^a → ←	Mute		PROTOCOL
HLTA	HLTA → ←	Mute		PROTOCOL
SELECT	SELECT(1) → ←	Mute		PROTOCOL
nSELECT	('93 70' ~UIDTX ₁ BCC CRC_A) → ←	Mute		PROTOCOL
RATS	RATS(0,0) → ←	Mute		PROTOCOL
Error condition	S(DESELECT, ~CRC_A) → ←	Mute		PROTOCOL
DESELECT	S(DESELECT) → ←	S(DESELECT)	65536/fc as specified in ISO/IEC 14443-4: 2008, 8.1	HALT
PPS	PPS(0,0,0) → ←	Mute, or PPS response ^b		PROTOCOL
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	I(0) ₀ (TEST_RESPONSE1(1))	< FWT	PROTOCOL
^a Let $1 \leq n_i \leq 32$. ^b PPS response is returned if the PICC supports PPS.				

G.3.3.13.3 Test report

Fill the appropriate row in Table G.61 according to Table G.29.

Table G.29 — Result criteria for behavior of the PICC Type A in the PROTOCOL state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.4 Scenario G.13: Handling of Type A anticollision

G.3.4.1 Scope

This test is to perform a full bitwise anticollision loop according to ISO/IEC 14443-3:—, 6.5.3.

G.3.4.2 Procedure

- a) Put the PICC into the field.
- b) Put the PICC into READY(1) state.
- c) Execute AnticollisionA.
- d) Put the PICC into READY*(1) state.
- e) Execute AnticollisionA.

Pseudocode: Type A anticollision procedure

```

1 Procedure AnticollisionA
2 // TPDUSend and TPDUREcv are PCD specific functions
3 // to send and receive frames
4
5 for c = 1 to CascadeLevels do
6
7 // anticollision over UID bits
8 for p = 1 to 32 do
9 // enter desired cascade level
10 if c ≥ 2 then TPDUSend(SELECT(1))
11 if c = 3 then TPDUSend(SELECT(2))
12 // anticollision with matched bit
13 NVB[[1..4]] = (p + 16) mod 8
14 NVB[[5..8]] = (p + 16) div 8
15 TPDUSend (SEL(c) NVB UIDTXc[[1..p]])
16 if TPDUREcv() ≠ (UIDTXc[[p+1..32]] BCC) then return FAIL
17 // anticollision with unmatched bit
18 TPDUSend(SEL(c) NVB UIDTXc[[1..p-1]] ~UIDTXc[[p]])
19 if TPDUREcv() ≠ Mute then return FAIL
20 // re-enter READY(1) (resp. READY*(1)) state
21 TPDUSend (WUPA)
22 end for
23 end for
24 return PASS

```

G.3.4.3 Test report

Fill the appropriate row in Table G.61 according to Table G.30.

Table G.30 — Result criteria for handling of Type A anticollision

Explanation	Test result
Only when every Anticollision Test procedure has returned PASS	PASS
When any Anticollision Test procedure has returned the value FAIL	FAIL

G.3.5 Handling of RATS

Handling of RATS is tested in G.3.3.7 and G.3.3.12.

Scenario G.14: RATS after wrong RATS

Scenario G.14, which was defined in ISO/IEC 10373-6:2001/Amd.1:2007, is deleted.

Scenario G.15: RATS after RATS

Scenario G.15, which was defined in ISO/IEC 10373-6:2001/Amd.1:2007, is deleted.

G.3.6 Handling of PPS request

G.3.6.1 Scope

This test is to determine the handling of the PPS request by the PICC Type A according to ISO/IEC 14443-4:2008, 5.6.2.2.

G.3.6.2 Procedure

Perform the following steps for each Scenario G.16 through G.19 listed in this subclause:

- a) Put the PICC in PROTOCOL state.
- b) Send the command as described under the PICC-test-apparatus column in each Scenario.
- c) Check that the response of the PICC conforms to the one given in the PICC column.
- d) Check if the PICC is in PROTOCOL state.

Scenario G.16: PPS without parameter change

Scenario G.16, which was defined in ISO/IEC 10373-6:2001/Amd.1:2007, is deleted.

Scenario G.17: PPS without PPS1

PICC-test-apparatus		PICC
('D0 01' CRC_A)	→	Mute or ('D0' CRC_A) ^a
	←	
^a Both responses are valid.		

Scenario G.18: PPS after PPS

PICC-test-apparatus		PICC
PPS(0,0,0)	→	Mute or ('D0' CRC_A) ^a
	←	
PPS(0,0,0)	→	Mute
	←	
^a Response depends on whether the PICC supports PPS or not. If the PICC does not support any changeable parameters it may not support the PPS request because the PCD shall not send PPS to such a PICC (see ISO/IEC 14443-4:2008, Clause 5, 6th dash)		

Scenario G.19: PPS after unreceived PPS

PICC-test-apparatus		PICC
('D0 11 00' ~CRC_A)	→	Mute
	←	
PPS(0,0,0)	→	Mute
	←	

G.3.6.3 Test report

Fill the appropriate rows in Table G.61 according to Table G.31.

Table G.31 — Result criteria for handling of PPS request

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.3.7 Scenario G.20: Handling of FSD

G.3.7.1 Scope

This test is to determine if the PICC Type A respects the FSD value as negotiated by the RATS according to ISO/IEC 14443-4:2008, 5.1.

G.3.7.2 Procedure

Perform the following steps for each FSDI = 0 to 8:

- a) Put the PICC into ACTIVE state.
- b) Send the RATS(0, fsdi) command with parameter fsdi as in the particular test.
- c) Check that the PICC answer is a valid ATS and that its size is \leq FSD.

NOTE The PICC may require additional sequences to be ready to accept TEST_COMMAND2(2).

- d) Send the I-block I(0)₀(TEST_COMMAND2(2)).
- e) Check that the size of the I-block sent by the PICC is \leq FSD.

G.3.7.3 Test report

Fill the appropriate row in Table G.61 according to Table G.32.

Table G.32 — Result criteria for Scenario G.20: Handling of FSD

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4 Test method for initialization of the PICC Type B

G.4.1 Introduction

This chapter is to test if the PICC Type B conforms to the ISO/IEC 14443-3:— standard. If compliance with ISO/IEC 14443-4:2008 is not required all tests containing ISO/IEC 14443-4 commands need not be applied.

If the PICC does not support the REQB/WUPB with $N > 1$ nor the Slot-MARKER command (see ISO/IEC 14443-3:—, 7.6.1) all tests containing these commands need not be applied.

G.4.2 Scenario G.21: Polling

G.4.2.1 Scope

This test is to determine the behavior of the PICC Type B on receiving of REQB according to ISO/IEC 14443-3:—, Clause 5.

G.4.2.2 Procedure

Perform the following steps for 3 different operating fields of 1,5, 4,5 and 7,5 A/m (rms):

- a) Place the PICC into the field and adjust it.
- b) Switch the RF operating field off for a minimum time for resetting a PICC in accordance with ISO/IEC 14443-3:—, 5.4.
- c) Switch the RF operating field on.

- d) Wait 5 ms and send a valid REQB(1) command frame.
- e) Record the presence and the content of the PICC response.
- f) Switch the RF operating field off for a minimum time for resetting a PICC in accordance with ISO/IEC 14443-3:—, 5.4.
- g) Switch the RF operating field on.
- h) Wait 5 ms and send a valid REQA command frame (with Type A modulation).
- i) Wait 5 ms and send a valid REQB(1) command frame.
- j) Record the presence and the content of the PICC response.

G.4.2.3 Test report

Fill the appropriate row in Table G.62 according to Table G.33:

Table G.33 — Result criteria for Scenario G.21: Polling

Explanation	Test result
Only when the PICC's response is a valid ATQB in both steps e) and j)	PASS
When the PICC's response isn't a valid ATQB in any of steps e) or j)	FAIL

G.4.3 Scenario G.22: PICC Reception

G.4.3.1 Scope

This test is to determine the behavior of a Type B PICC when receiving PCD messages according to ISO/IEC 14443-3:—, 7.1.1, 7.1.2, 7.1.4 and 7.1.5.

G.4.3.2 Procedure

Perform the following steps for each row of Table G.34:

- a) Place the Reference PICC into the field.
- b) Set the frame parameters of the PICC-test-apparatus according to Table G.34.
- c) Send a REQB command.
- d) Record the presence, content and timing of the PICC response.
- e) Check that the frame format of the PICC response conforms to the following:
 - The PICC response shall be a valid ATQB;
 - The SOF logic 0 timing shall be between 10 and 11 etu;
 - The SOF logic 1 timing shall be between 2 and 3 etu;
 - The EOF logic 0 timing shall be between 10 and 11 etu;

- The TR0 timing shall be in the range $64/fs \leq TR0 \leq 256/fs$;
- The TR1 timing shall be in the range $80/fs \leq TR1 \leq 200/fs$;
- The PICC shall be turn off the subcarrier between 0 and 2 etu after end EOF.

Table G.34 — Type B frame parameters

EGT [µs]	SOF (logic 0) [etu]	SOF (logic 1) [etu]	EOF [etu]
0	10	2	10
57	10	2	10
0	11	2	10
0	10	3	10
0	10	2	11

G.4.3.3 Test report

Fill the appropriate row in Table G.62 according to Table G.35.

Table G.35 — Result criteria for Scenario G.22: PICC Reception

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.4 Testing of the PICC Type B state transitions

These tests are to verify the correct implementation of the PICC Type B state diagram as described in ISO/IEC 14443-3:—, 7.4.

G.4.4.1 General Test Outline

This is the same procedure as described for the PICC Type A (see G.3.3.2).

G.4.4.1.1 Functions to set the PICC in Test Initial State TIS

Putting the PICC into the State TIS will be done by a sequence of transition commands specified in Table G.37. The general method is as follows:

In order to put the PICC into State TIS, look up the corresponding State Transition Sequence in Table G.36. Then successively apply the state transitions described in this column by looking up the corresponding commands in the State Transition Table. Always check the content and integrity of the PICC response.

Table G.36 — State Transition Sequence

TIS	State Transition Sequence
POWER-OFF	---
IDLE	POWER-OFF → IDLE
READY-REQUESTED	POWER-OFF → IDLE → READY-REQUESTED
READY-DECLARED	POWER-OFF → IDLE → READY-DECLARED
PROTOCOL	POWER-OFF → IDLE → READY-DECLARED → PROTOCOL
HALT	POWER-OFF → IDLE → READY-DECLARED → HALT

Table G.37 — State Transition

State → Next State	PICC-test-apparatus	PICC
POWER-OFF → IDLE	Power On → (RF operating Field on)	Mute
IDLE → READY-REQUESTED	REQB(16) ↔	Mute ^a
IDLE → READY-DECLARED	REQB(1) ↔	ATQB
READY-DECLARED → HALT	HLTB →	'00' CRC_B
READY-DECLARED → PROTOCOL	ATTRIB(0,0) ↔	ATA(0)

^a In case the PICC has selected slot 1, the REQB command shall be reissued until the PICC doesn't answer ATQB.

G.4.4.1.2 Functions for checking the validity of the Test Target State TTS

The following Table G.38 describes the state transitions, which are used to check whether the PICC is in the state S. The content of the PICC answer (i.e. ATQB...) should be thoroughly checked for ISO/IEC 14443-3:— and ISO/IEC 14443-4 conformance.

NOTE The tests may cause the PICC to change state.

Table G.38 — Checking the TTS

TTS	PICC-test-apparatus	PICC
IDLE	REQB(1) →	ATQB
READY-REQUESTED	SLOTMARKER (n) ^a →	ATQB
READY-DECLARED	ATTRIB(0,0) →	ATA(0)

TTS	PICC-test-apparatus	PICC
PROTOCOL	I(0) _o (TEST_COMMAND1(1)) →	I(0) _o (TEST_RESPONSE1(1)) ←
HALT	REQB(1) → WUPB(1) →	Mute ATQB

^a Since the selected PICC slot is unknown, the Slot-MARKER command shall be reissued with different slot values until an ATQB is received.

G.4.4.2 Scenario G.23: Behavior of the PICC Type B in the IDLE state

G.4.4.2.1 Scope

This test is to determine the behavior of the PICC Type B in the IDLE state according to ISO/IEC 14443-3:—, 7.4.4.

G.4.4.2.2 Procedure

Perform the following steps for every row of Table G.39:

- a) Put the PICC into IDLE state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) Check if the PICC is in the state TTS.

Table G.39 — Transitions from IDLE state

Transition	PICC-test-apparatus	PICC	TTS
REQB	REQB(1) →	ATQB ←	READY-DECLARED
WUPB	WUPB(1) →	ATQB ←	READY-DECLARED
REQB (wrong CRC)	('05 00 00' ~CRC_B) →	Mute ←	IDLE
WUPB (wrong CRC)	('05 00 08' ~CRC_B) →	Mute ←	IDLE
HLTB ^b	HLTB →	Mute ←	IDLE
ATTRIB ^b	ATTRIB(0,0) →	Mute ←	IDLE

Transition	PICC-test-apparatus	PICC	TTS
Slot-MARKER	SLOTMARKER(n) ^a	→ ← Mute	IDLE
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1))	→ ← Mute	IDLE
DESELECT	S(DESELECT)	→ ← Mute	IDLE
REQB (Unmatched AFI)	REQB(1,nAFI)	→ ← Mute	IDLE
WUPB (Unmatched AFI)	WUPB(1,nAFI)	→ ← Mute	IDLE
HLTB (Unmatched PUPI)	HLTB(~PUPI)	→ ← Mute	IDLE
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	→ ← Mute	IDLE
REQB ^d	REQB(16) ^c	→ ← Mute	READY-REQUESTED
WUPB ^d	WUPB(16) ^c	→ ← Mute	READY-REQUESTED

^a n shall run through all values $2 \leq n \leq 16$.

^b For PICCs using random PUPI apply an arbitrary one for this command.

^c Nevertheless, there is statistically a probability of 1/16 so that the PICC answers ATQB and goes to READY-DECLARED state.

^d If the PICC does not support the REQB/WUPB with $N > 1$ (see ISO/IEC 14443-3:—, 7.6.1) the test need not be applied.

G.4.4.2.3 Test report

Fill the appropriate row in Table G.62 according to Table G.40.

Table G.40 — Result criteria for Scenario G.23: Behavior of the PICC Type B in the IDLE state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.4.3 Scenario G.24: Behavior of the PICC Type B in the READY-REQUESTED sub-state

If the PICC does not support the REQB/WUPB with $N > 1$ nor the Slot-MARKER command (see ISO/IEC 14443-3:—, 7.6.1) this Scenario need not be applied.

G.4.4.3.1 Scope

This test is to determine the behavior of the PICC Type B in the READY-REQUESTED sub-state according to ISO/IEC 14443-3:—, 7.4.5.

G.4.4.3.2 Procedure

Perform the following steps for every row of Table G.41:

- a) Put the PICC into READY-REQUESTED sub-state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) Check if the PICC is in the state TTS.

Table G.41 — Transitions from READY-REQUESTED sub-state

Transition	PICC-test-apparatus	PICC	TTS
REQB	REQB(1) → ←	ATQB	READY-DECLARED
WUPB	WUPB(1) → ←	ATQB	READY-DECLARED
REQB (wrong CRC)	('05 00 00' ~CRC_B) → ←	Mute	READY-REQUESTED
WUPB (wrong CRC)	('05 00 08' ~CRC_B) → ←	Mute	READY-REQUESTED
HLTB ^b	HLTB → ←	Mute	READY-REQUESTED
ATTRIB ^b	ATTRIB(0,0) → ←	Mute	READY-REQUESTED
Slot-MARKER	SLOTMARKER(n) ^a → ←	ATQB or Mute	READY-DECLARED
ISO/IEC 14443-4 command	I(0) _b (TEST_COMMAND1(1)) → ←	Mute	READY-REQUESTED
DESELECT	S(DESELECT) → ←	Mute	READY-REQUESTED
REQB	REQB(16) ^c → ←	Mute	READY-REQUESTED
WUPB	WUPB(16) ^c → ←	Mute	READY-REQUESTED
REQB (Unmatched AFI)	REQB(1,nAFI) → ←	Mute	IDLE ^d

Transition	PICC-test-apparatus	PICC	TTS
WUPB (Unmatched AFI)	WUPB(1,nAFI)	→ ← Mute	IDLE ^d
HLTB (Unmatched PUPI)	HLTB(~PUPI)	→ ← Mute	READY-REQUESTED
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	→ ← Mute	READY-REQUESTED

^a n shall run through all values $2 \leq n \leq 16$. The PICC shall respond ATQB at exactly one value of n, else Mute.

^b For PICCs using random PUPI apply an arbitrary one for this command.

^c Nevertheless, there is statistically a probability of 1/16 so that the PICC answers ATQB and goes to READY-DECLARED state.

^d Send all Slot-MARKER commands and verify that there is no response before checking the IDLE state.

G.4.4.3.3 Test report

Fill the appropriate row in Table G.62 according to Table G.42.

Table G.42 — Result criteria for Scenario G.24: Behavior of the PICC Type B in the READY-REQUESTED sub-state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.4.4 Scenario G.25: Behavior of the PICC Type B in the READY-DECLARED sub-state

G.4.4.4.1 Scope

This test is to determine the behavior of the PICC Type B in the READY-DECLARED sub-state according to ISO/IEC 14443-3:—, 7.4.6.

G.4.4.4.2 Procedure

Perform the following steps for every row of Table G.43:

- Put the PICC into READY-DECLARED sub-state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- Check if the PICC is in the state TTS.

Table G.43 — Transitions from READY-DECLARED sub-state

Transition	PICC-test-apparatus	PICC	TTS
REQB	REQB(1)	→ ← ATQB	READY-DECLARED

Transition	PICC-test-apparatus	PICC	TTS
WUPB	WUPB(1) → ←	ATQB	READY-DECLARED
REQB (wrong CRC)	('05 00 00' ~CRC_B) → ←	Mute	READY-DECLARED
WUPB (wrong CRC)	('05 00 08' ~CRC_B) → ←	Mute	READY-DECLARED
HLTB	HLTB → ←	('00' CRC_B)	HALT
ATTRIB	ATTRIB(0,0) → ←	ATA(0)	PROTOCOL
Slot-MARKER	SLOTMARKER(n) ^a → ←	Mute	READY-DECLARED
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	Mute	READY-DECLARED
DESELECT	S(DESELECT) → ←	Mute	READY-DECLARED
REQB ^b	REQB(16) ^c → ←	Mute	READY-REQUESTED
WUPB ^b	WUPB(16) ^c → ←	Mute	READY-REQUESTED
HLTB (Unmatched PUPI)	HLTB(~PUPI) → ←	Mute	READY-DECLARED
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI) → ←	Mute	READY-DECLARED
REQB (Unmatched AFI)	REQB(1,nAFI) → ←	Mute	IDLE ^d
WUPB (Unmatched AFI)	WUPB(1,nAFI) → ←	Mute	IDLE ^d

^a n shall run through all values 2 ≤ n ≤ 16.

^b If the PICC does not support the REQB/WUPB with N > 1 (see ISO/IEC 14443-3:—, 7.6.1) the test need not be applied.

^c Nevertheless, there is statistically a probability of 1/16 so that the PICC answers ATQB and goes to READY-DECLARED state.

^d Send ATTRIB command and verify that there is no response before checking the IDLE state.

G.4.4.4.3 Test report

Fill the appropriate row in Table G.62 according to Table G.44.

Table G.44 — Result criteria for Scenario G.25: Behavior of the PICC Type B in the READY-DECLARED sub-state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.4.5 Scenario G.26: Behavior of the PICC Type B in the HALT state**G.4.4.5.1 Scope**

This test is to determine the behavior of the PICC Type B in the HALT state according to ISO/IEC 14443-3:—, 7.4.8.

G.4.4.5.2 Procedure

Perform the following steps for every row of Table G.45:

- Put the PICC into HALT state.
- Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- Check if the PICC response is as indicated in the PICC column.
- Check if the PICC is in the state TTS.

Table G.45 — Transitions from HALT state

Transition	PICC-test-apparatus	PICC	TTS
REQB	REQB(1)	→	HALT
		←	
WUPB	WUPB(1)	→	READY-DECLARED
		←	
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	→	HALT
		←	
HLTB	HLTB	→	HALT
		←	
ATTRIB	ATTRIB(0,0)	→	HALT
		←	
Slot-MARKER	SLOTMARKER(n) ^a	→	HALT
		←	
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1))	→	HALT
		←	
DESELECT	S(DESELECT)	→	HALT
		←	

Transition	PICC-test-apparatus	PICC	TTS
WUPB (Unmatched AFI)	WUPB(1,nAFI) → ←	Mute	IDLE
REQB (Unmatched AFI)	REQB(1,nAFI) → ←	Mute	HALT
HLTB (Unmatched PUPI)	HLTB(~PUPI) → ←	Mute	HALT
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI) → ←	Mute	HALT
WUPB ^b	WUPB(16) ^c → ←	Mute	READY-REQUESTED

^a n shall run through all values $2 \leq n \leq 16$.

^b If the PICC does not support the REQB/WUPB with $N > 1$ (see ISO/IEC 14443-3:—, 7.6.1) the test need not be applied.

^c Nevertheless, there is statistically a probability of 1/16 so that the PICC answers ATQB and goes to READY-DECLARED state.

G.4.4.5.3 Test report

Fill the appropriate row in Table G.62 according to Table G.46.

Table G.46 — Result criteria for Scenario G.26: Behavior of the PICC Type B in the HALT state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.4.6 Scenario G.27: Behavior of the PICC Type B in the PROTOCOL state

G.4.4.6.1 Scope

This test is to determine the behavior of the PICC Type B in the PROTOCOL state according to ISO/IEC 14443-4:2008. This test shall ensure that the activated PICC does not respond to any initialization command.

G.4.4.6.2 Procedure

Perform the following steps for every row of Table G.47:

- a) Put the PICC into PROTOCOL state.
- b) Perform the state transition by sending the command as indicated in the PICC-test-apparatus column.
- c) Check if the PICC response is as indicated in the PICC column.
- d) Check if the PICC is in the state TTS.

Table G.47 — Transitions from PROTOCOL state

Transition	PICC-test-apparatus	PICC	FDT	TTS
REQB	REQB(1) → ←	Mute		PROTOCOL
WUPB	WUPB(1) → ←	Mute		PROTOCOL
REQB (wrong CRC)	('05 00 00' ~CRC_B) → ←	Mute		PROTOCOL
WUPB (wrong CRC)	('05 00 08' ~CRC_B) → ←	Mute		PROTOCOL
HLTB	HLTB → ←	Mute		PROTOCOL
ATTRIB	ATTRIB(0,0) → ←	Mute		PROTOCOL
Slot-MARKER	SLOTMARKER(n) ^a → ←	Mute		PROTOCOL
ISO/IEC 14443-4 command	I(0) ₀ (TEST_COMMAND1(1)) → ←	I(0) ₀ (TEST_RESPONSE1(1))	< FWT	PROTOCOL
DESELECT	S(DESELECT) → ←	S(DESELECT)		HALT
WUPB (Unmatched AFI)	WUPB(1,nAFI) → ←	Mute		PROTOCOL
REQB (Unmatched AFI)	REQB(1,nAFI) → ←	Mute		PROTOCOL
HLTB (Unmatched PUPI)	HLTB(~PUPI) → ←	Mute		PROTOCOL
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI) → ←	Mute		PROTOCOL

^a n shall run through all values $2 \leq n \leq 16$.

G.4.4.6.3 Test report

Fill the appropriate row in Table G.62 according to Table G.48.

Table G.48 — Result criteria for Scenario G.27: Behavior of the PICC Type B in the PROTOCOL state

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.5 Scenario G.28: Handling of Type B anticollision

G.4.5.1 Scope

The purpose of this test is to determine the handling of a PICC Type B anticollision according to ISO/IEC 14443-3:—, 7.4.1.

The core of this test is the procedure AnticollisionB(N, outparam chi2) which is defined in the pseudo code below. The procedure performs 256 REQB(N) commands and following Slot-MARKER commands and counts how many times each of the N slots has been selected by the PICC. The procedure also checks if the PICC has mapped each REQB(N) request to exactly one slot. If this is not the case the test returns FAIL.

Since Type B anticollision is based on random selection of the slots, statistical methods shall be used for verification. As it is the nature of all statistical tests, this test can fail even in the case the PICC behaves correctly. This failure is called a “Type I error” in statistical terms. This error cannot be completely avoided. Instead, the probability of its occurrence can be controlled by the so called “significance value” α . This means, the smaller α , the less probable the “Type I error”. However, this does not mean that one should select α as small as possible. This is because the smaller α is, the more probable is that the test passes a bad PICC (i.e. a PICC that doesn’t select the slots with the right probability). In statistical terms this is called a “Type II error”.

The PICC shall additionally select each of the N slots with equal probability (i.e. 1/N). In order to verify this, the statistical χ^2 -test on all slots shall be performed. The result of this test is the value chi2 which shall be compared against the $\chi^2_{\alpha, N-1}$ quintile.

G.4.5.2 Procedure

If one of the statistical tests fails in step e) the test lab may rerun the test for this parameter N.

Perform the following steps for each value N = 2, 4, 8, 16.

- a) Set the significance level α to 0,005 and lookup from Table G.49 the corresponding $\chi^2_{\alpha, N-1}$ quintile. Other choices of α according to Table G.49 are optional for the test applicants.
- b) Reset the PICC.
- c) Execute AnticollisionB(N, chi2).
- d) If AnticollisionB returns FAIL, fail the test.
- e) If $\text{chi2} \leq \chi^2_{\alpha, N-1}$ then pass the test else fail the test.

Table G.49 — α quintile values

α	$\chi^2_{\alpha, N-1}$			
	$\chi^2_{\alpha, 1}$	$\chi^2_{\alpha, 3}$	$\chi^2_{\alpha, 7}$	$\chi^2_{\alpha, 15}$
0,1 (optional)	2,706	6,251	12,017	22,307
0,05 (optional)	3,841	7,815	14,067	24,996
0,01 (optional)	6,635	11,345	18,475	30,578
0,005	7,879	12,838	20,278	32,801

Pseudocode: Type B anticollision procedure

```

1 Procedure AnticollisionB(N, chi2)
2
3 // TPDUSend and TPDUREcv are PCD specific functions
4 // to send and receive TPDU frames
5
6 // probability for selecting slot
7 p = 1/N
8
9 // clear slot counters
10 for i from 1 to N do
11   slots[i] = 0
12 Endfor
13
14 // collect data
15 for i from 1 to 256 do
16   Reset the PICC
17   TPDUSend (REQB(N))
18   if TPDUREcv() = ATQB then
19     slots[1] = slots[1]+1
20   endif
21   for j from 2 to N do
22     TPDUSend (SLOTMARKER(j))
23     if TPDUREcv () = ATQB then
24       slots[j] = slots[j]+1
25     endif
26   endfor
27 endfor
28
29 // check that exactly
30 // one slot has been selected at each run
31 cnt = 0
32 for i from 1 to N do
33   cnt = cnt + slots[i]
34 endfor
35 if cnt ≠ 256 then
36   return FAIL
37 endif
38 Chi2 = 0
39 for i from 1 to N do
40   chi2 = chi2 + slots[i]*slots[i]
41 endfor
42 chi2 = chi2*N/256 - 256
43 return PASS

```

NOTE Continue with step e) only if PASS is returned in line 43.

G.4.5.3 Test report

Fill the appropriate row in Table G.62 according to Table G.50.

Table G.50 — Result criteria for Scenario G.28: Handling of Type B anticollision

Explanation	Test result
Only when every Anticollision Test procedure has returned PASS	PASS
When any Anticollision Test procedure has returned the value FAIL	FAIL

G.4.6 Handling of ATTRIB

G.4.6.1 Scope

This test is to determine the behavior of the PICC Type B on ATTRIB command according to ISO/IEC 14443-3:—, 7.10.

G.4.6.2 Procedure

Perform the following steps for each of Scenario G.29 through G.30 listed in this subclause:

- a) Put the PICC into READY-DECLARED sub-state.
- b) Send the command sequence as described in the PICC-test-apparatus column.
- c) Check that the response of the PICC conforms with the one given in the PICC column.
- d) Check if the PICC is in PROTOCOL state.

Scenario G.29: ATTRIB with wrong PUPI

PICC-test-apparatus		PICC
('1D' ~PUPI '00 00 01 00' CRC_B)	→	
	←	Mute
ATTRIB(0,0)	→	
	←	ATA(0)

Scenario G.30: ATTRIB after wrong ATTRIB

PICC-test-apparatus		PICC
('1D' PUPI '00 00 01 00' ~CRC_B)	→	
	←	Mute
ATTRIB(0,0)	→	
	←	ATA(0)

G.4.6.3 Test report

Fill the appropriate row in Table G.62 according to Table G.51.

Table G.51 — Result criteria for handling of ATTRIB

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.4.7 Scenario G.31: Handling of Maximum Frame Size**G.4.7.1 Scope**

This test is to determine if the PICC Type B respects the FSD size according to ISO/IEC 14443-3:—, 7.10.4.

G.4.7.2 Procedure

Perform the following steps for each FSDI = 0 to 8:

- a) Put the PICC into READY-DECLARED sub-state as described in G.4.4.1.1.
- b) Send the ATTRIB(0, fsdi) command with parameter fsdi as in the particular test.
- c) Check if the PICC answer is ATA(0).

NOTE The PICC may require additional sequences to be ready to accept TEST_COMMAND2(2).

- d) Send the I-block I(0)₀(TEST_COMMAND2(2)).
- e) Check if the size of the I-block response of the PICC response is ≤ FSD.

G.4.7.3 Test report

Fill the appropriate row in Table G.62 according to Table G.52.

Table G.52 — Result criteria for Scenario G.31: Handling of Maximum Frame Size

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.5 Test methods for logical operation of the PICC Type A or Type B**G.5.1 Introduction**

This chapter contains tests verifying that the activated PICC conforms to the ISO/IEC 14443-4:2008. This chapter applies to PICC Type A and Type B.

G.5.1.1 PICC activation process

PICC activation is the process of putting the PICC in the state where protocol blocks defined in ISO/IEC 14443-4:2008 may be exchanged. This process is dependent on the PICC type.

NOTE The PICC may require additional sequences to be ready to accept step 1 of the Scenario.

G.5.1.1.1 Activation of the PICC Type A

- a) Put the PICC into ACTIVE state as described in G.3.3.2.1.
- b) Send RATS(cid, fsdi).
- c) Check that the PICC response is a valid ATS.

G.5.1.1.2 Activation of the PICC Type B

- a) Put the PICC into READY-DECLARED sub-state as described in G.4.4.1.1.
- b) Send ATTRIB(cid, fsdi).
- c) Check that the PICC response is a valid ATA.

G.5.2 PICC reaction to ISO/IEC 14443-4 Scenarios

G.5.2.1 Scope

This test is to determine the behavior of the PICC according to ISO/IEC 14443-4:2008, Clause 7. This test uses implementations of the protocol Scenarios of ISO/IEC 14443-4:2008, Annex B.

G.5.2.2 Procedure

Perform the following steps for each of Scenario G.32 through G.54 listed in this subclause:

- a) Activate the PICC as described in G.5.1.1, use CID = 0 and FSDI = 0.
- b) For each step in the Scenario do:
 - 1) Send the command as described in the PICC-test-apparatus column.
 - 2) Check that the PICC response matches the one of the PICC column.
- c) End for.

Scenario G.32: Exchange of I-blocks

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
2	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.33: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX) (WTXM)
2	S(WTX) (WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
3	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.34: DESELECT

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
2	S(DESELECT)	→	
		←	S(DESELECT)
3	REQA or REQB(1) ^a	→	
		←	Mute
4	WUPA or WUPB(1) ^a	→	
		←	ATQA or ATQB ^a

^a For the PICC Type A, the left option shall be used. For the PICC Type B, the right option shall be used.

Scenario G.35: PCD uses chaining

Step	PICC-test-apparatus		PICC
1	I(1) ₀ (TEST_COMMAND1(2) ₁)	→	
		←	R(ACK) ₀
2	I(0) ₁ (TEST_COMMAND1(2) ₂)	→	
		←	I(0) ₁ (TEST_RESPONSE1(2))
3	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))

Scenario G.36: PICC uses chaining

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND2(2))	→	
		←	I(1) ₀ (TEST_RESPONSE2(2) ₁)
2	R(ACK) ₁	→	
		←	I(0) ₁ (TEST_RESPONSE2(2) ₂)
3	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))

Scenario G.37: Start of protocol

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1), ~CRC)	→	
		←	Mute
2	R(NAK) ₀	→	
		←	R(ACK) ₁
3	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
4	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.38: Exchange of I-blocks

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
2	I(0) ₁ (TEST_COMMAND1(1), ~CRC)	→	
		←	Mute
3	R(NAK) ₁	→	
		←	R(ACK) ₀

Step	PICC-test-apparatus		PICC
4	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))
5	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))

Scenario G.39: Exchange of I-blocks 1

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
2	R(NAK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
3	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.40: Exchange of I-blocks 2

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
2	R(NAK, ~CRC) ₀	→	
		←	Mute
3	R(NAK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))
4	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.41: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX)(WTXM)

Step	PICC-test-apparatus		PICC
2		→	
		←	S(WTX)(WTXM)
3	S(WTX)(WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
4	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.42: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX)(WTXM)
2	R(NAK, ~CRC) ₀	→	
		←	Mute
3	R(NAK) ₀	→	
		←	S(WTX)(WTXM)
4	S(WTX)(WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
5	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.43: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX)(WTXM)
2	S(WTX)(WTXM, ~CRC)	→	
		←	Mute
3	R(NAK) ₀	→	
		←	S(WTX)(WTXM)

Step	PICC-test-apparatus		PICC
4	S(WTX)(WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
5	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.44: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX)(WTXM)
2	S(WTX)(WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
3	R(NAK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE3)
4	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.45: Request for waiting time extension

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND3)	→	
		←	S(WTX)(WTXM)
2	S(WTX)(WTXM)	→	
		←	I(0) ₀ (TEST_RESPONSE3)
3	R(NAK, ~CRC) ₀	→	
		←	Mute
4	R(NAK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE3)
5	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.46: DESELECT

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND1(1))	→
		← I(0) ₀ (TEST_RESPONSE1(1))
2	S(DESELECT, ~CRC)	→
		← Mute
3	S(DESELECT)	→
		← S(DESELECT)
4	REQA or REQB(1) ^a	→
		← Mute
5	WUPA or WUPB(1) ^a	→
		← ATQA or ATQB ^a

^a For the PICC Type A, the left option shall be used. For the PICC Type B, the right option shall be used.

Scenario G.47: PCD uses chaining

Step	PICC-test-apparatus	PICC
1	I(1) ₀ (TEST_COMMAND1(3) ₁)	→
		← R(ACK) ₀
2	R(NAK) ₀	→
		← R(ACK) ₀
3	I(1) ₁ (TEST_COMMAND1(3) ₂)	→
		← R(ACK) ₁
4	I(0) ₀ (TEST_COMMAND1(3) ₃)	→
		← I(0) ₀ (TEST_RESPONSE1(3))
5	I(0) ₁ (TEST_COMMAND1(1))	→
		← I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.48: PCD uses chaining

Step	PICC-test-apparatus	PICC
1	I(1) ₀ (TEST_COMMAND1(3) ₁)	→
		← R(ACK) ₀
2	I(1) ₁ (TEST_COMMAND1(3) ₂ , ~CRC)	→
		← Mute
3	R(NAK) ₁	→
		← R(ACK) ₀
4	I(1) ₁ (TEST_COMMAND1(3) ₂)	→
		← R(ACK) ₁
5	I(0) ₀ (TEST_COMMAND1(3) ₃)	→
		← I(0) ₀ (TEST_RESPONSE1(3))
6	I(0) ₁ (TEST_COMMAND1(1))	→
		← I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.49: PCD uses chaining

Step	PICC-test-apparatus	PICC
1	I(1) ₀ (TEST_COMMAND1(3) ₁)	→
		← R(ACK) ₀
2	R(NAK, ~CRC) ₀	→
		← Mute
3	R(NAK) ₀	→
		← R(ACK) ₀
4	I(1) ₁ (TEST_COMMAND1(3) ₂)	→
		← R(ACK) ₁
5	I(0) ₀ (TEST_COMMAND1(3) ₃)	→
		← I(0) ₀ (TEST_RESPONSE1(3))
6	I(0) ₁ (TEST_COMMAND1(1))	→
		← I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.50: PICC uses chaining

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND2(3))	→	
		←	I(1) ₀ (TEST_RESPONSE2(3) ₁)
2	R(ACK, ~CRC) ₁	→	
		←	Mute
3	R(ACK) ₁	→	
		←	I(1) ₁ (TEST_RESPONSE2(3) ₂)
4	R(ACK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE2(3) ₃)
5	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.51: PICC uses chaining

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND2(3))	→	
		←	I(1) ₀ (TEST_RESPONSE2(3) ₁)
2	R(ACK) ₁	→	
		←	I(1) ₁ (TEST_RESPONSE2(3) ₂)
3	R(ACK) ₁	→	
		←	I(1) ₁ (TEST_RESPONSE2(3) ₂)
4	R(ACK) ₀	→	
		←	I(0) ₀ (TEST_RESPONSE2(3) ₃)
5	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))

Scenario G.52: PICC uses chaining

Step	PICC-test-apparatus		PICC
1	I(0) ₀ (TEST_COMMAND2(2))	→	
		←	I(1) ₀ (TEST_RESPONSE2(2) ₁)
2	R(NAK) ₀	→	
		←	I(1) ₀ (TEST_RESPONSE2(2) ₁)
3	R(ACK) ₁	→	
		←	I(0) ₁ (TEST_RESPONSE2(2) ₂)
4	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))

Scenario G.53: PICC Presence Check Method 1

NOTE This Scenario replaces an old Scenario that was removed.

Step	PICC-test-apparatus		PICC
1	I(empty) ₀	→	
		←	I() ₀
2	I(0) ₁ (TEST_COMMAND1(1))	→	
		←	I(0) ₁ (TEST_RESPONSE1(1))
3	I(empty) ₀	→	
		←	I() ₀

Scenario G.54: PICC Presence Check Method 2

NOTE This Scenario replaces an old Scenario that was removed.

Step	PICC-test-apparatus		PICC
1	R(NAK) ₀	→	
		←	R(ACK) ₁
2	R(NAK) ₀	→	
		←	R(ACK) ₁
3	I(0) ₀ (TEST_COMMAND1(1))	→	
		←	I(0) ₀ (TEST_RESPONSE1(1))

Step	PICC-test-apparatus	PICC
4	R(NAK) ₁ →	
		← R(ACK) ₀
5	I(0) ₁ (TEST_COMMAND1(1)) →	
		← I(0) ₁ (TEST_RESPONSE1(1))

G.5.2.3 Test report

Fill the appropriate rows in Table G.63 according to Table G.53.

Table G.53 — Result criteria for PICC reaction to ISO/IEC 14443-4 Scenarios

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.5.3 Handling of PICC error detection

G.5.3.1 Scope

This test is to determine the error detection mechanism of the PICC as described in ISO/IEC 14443-4:2008, 7.5.6.

G.5.3.2 Procedure

Perform the following steps for each of Scenario G.55 through G.57 listed in this subclause:

- a) Place the Reference PICC into the field.
- b) Activate the PICC as described in G.5.1.1, use CID = 0 and FSDI = 0.
- c) For each step in Scenario do:
 - 1) Send the command as described in the PICC-test-apparatus column.
 - 2) Check if the PICC response is as described in the PICC column.
- d) End for.

NOTE The comment column of the following Scenarios refers to the rules of ISO/IEC 14443-4:2008, 7.5.3 - 7.5.6.

Scenario G.55: Wrong CRC on I-block

Step	PICC-test-apparatus	PICC	Comment
1	I(0) ₀ (TEST_COMMAND1(1), ~CRC) →		7.5.6.1 a)
		← Mute	

Step	PICC-test-apparatus	PICC	Comment
2	I(0) ₀ (TEST_COMMAND1(1)) →		
		← I(0) ₀ (TEST_RESPONSE1(1))	

Scenario G.56: Wrong CRC on chained I-block

Step	PICC-test-apparatus	PICC	Comment
1	I(1) ₀ (TEST_COMMAND1(2) ₁) →		
		← R(ACK) ₀	
2	I(0) ₁ (TEST_COMMAND1(2) ₂ , ~CRC) →		7.5.6.1 a)
		← Mute	
3	I(0) ₁ (TEST_COMMAND1(2) ₂) →		
		← I(0) ₁ (TEST_RESPONSE1(2))	

Scenario G.57: Wrong CRC on S(WTX)-block

Step	PICC-test-apparatus	PICC	Comment
1	I(0) ₀ (TEST_COMMAND3) →		
		← S(WTX)(WTXM)	
2	S(WTX)(WTXM, ~CRC) →		7.5.6.1 a)
		← Mute	
3	S(WTX)(WTXM) →		
		← I(0) ₀ (TEST_RESPONSE3)	

G.5.3.3 Test report

Fill the appropriate rows in Table G.63 according to Table G.54.

Table G.54 — Result criteria for handling of PICC error detection

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.5.4 PICC reaction on CID

G.5.4.1 Scope

This test is to determine the reaction of the PICC to CID coding according to ISO/IEC 14443-4:2008, 7.1.1.2.

G.5.4.2 Procedure

Perform the following steps for each of Scenario G.58 through G.62 listed in this subclause. Use the proper CID test case table depending upon whether the PICC supports CID or not.

For each row in the CID test case tables in Table G.55 or Table G.56 do:

- a) Activate the PICC with cid_{ass} as indicated in the column Assigned CID.
- b) Perform a block exchange as described in the corresponding Scenario. Use the cid_{cmd} as described in the Command CID column in the CID test case table.
- c) Check if the PICC response matches with the response as in the PICC column in the Scenario. If two response options are indicated for the PICC, then the unique expected response will be determined from the expected PICC response column in the CID Test case table.

Table G.55 — CID test case table (for PICCs which support CID)

Test No. ^a	Assigned CID (cid_{ass})	Command CID (cid_{cmd})	Expected PICC response
1	1	1	Response 1 of the Scenario
2	0	0	Response 1 of the Scenario
3	0	NO CID	Response 1 of the Scenario
4	1	NO CID	Response 2 of the Scenario (Mute)
5	0	1	Response 2 of the Scenario (Mute)
6	1	0	Response 2 of the Scenario (Mute)
7	2	1	Response 2 of the Scenario (Mute)

^a Each test number in the table shall be tested with each of the Scenarios described.

Table G.56 — CID test case table (for PICCs which do not support CID)

Test No. ^a	Assigned CID (cid_{ass})	Command CID (cid_{cmd})	Expected PICC response
1	0	0	Response 2 of the Scenario (Mute)
2	0	NO CID	Response 1 of the Scenario
3	0	1	Response 2 of the Scenario (Mute)
4 ^b	1	NO CID	Response 1 of the Scenario

^a Each test number in the table shall be tested with each of the Scenarios described.

^b Applies to Type A PICC only

Scenario G.58: CID on I-block

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND1(1), CID = cid _{cmd}) →	← Response 1: I(0) ₀ (TEST_RESPONSE1(1), CID = cid _{cmd}) Response 2: Mute ^a
^a Response 1 or response 2 according to Table G.55 or Table G.56.		

Scenario G.59: CID on I-block with chaining

Step	PICC-test-apparatus	PICC
1	I(1) ₀ (TEST_COMMAND1(2) ₁ , CID = cid _{cmd}) →	← Response 1: R(ACK, CID = cid _{cmd}) ₀ Response 2: Mute ^a
^a Response 1 or response 2 according to Table G.55 or Table G.56.		

Scenario G.60: CID on R-block

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND2(3), CID ^a = cid _{ass}) →	← I(1) ₀ (TEST_RESPONSE2(3) ₁ , CID = cid _{ass})
2	R(ACK, CID ^a = cid _{cmd}) ₁ →	← Response 1: I(1) ₁ (TEST_RESPONSE2(3) ₂ , CID = cid _{cmd}) Response 2: Mute ^b
^a For PICC not supporting CID, use no CID.		
^b Response 1 or response 2 according to Table G.55 or Table G.56.		

Scenario G.61: CID on S(WTX)-block

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND3, CID ^a = cid _{ass}) →	← S(WTX)(WTXM, CID = cid _{ass})
2	S(WTX)(WTXM, CID ^a = cid _{cmd}) →	← Response 1: I(0) ₀ (TEST_RESPONSE3, CID ^a = cid _{cmd}) Response 2: Mute ^b
^a For PICC not supporting CID, use no CID.		
^b Response 1 or response 2 according to Table G.55 or Table G.56.		

Scenario G.62: CID on S(DESELECT)-block

Step	PICC-test-apparatus	PICC
1	S(DESELECT, CID = cid _{cmd}) →	← Response 1: S(DESELECT, CID = cid _{cmd}) Response 2: Mute ^a
^a Response 1 or response 2 according to Table G.55 or Table G.56.		

G.5.4.3 Test report

Fill the appropriate row in Table G.63 according to Table G.57.

Table G.57 — Result criteria for PICC reaction on CID

Explanation	Test result
Only when the PICC responded as indicated in the procedure	PASS
Any other case	FAIL

G.5.5 PICC reaction on NAD

G.5.5.1 Scope

This test is to determine the reaction of the PICC to NAD coding according to ISO/IEC 14443-4:2008, 7.1.1.3.

G.5.5.2 Procedure

Perform the following steps for each of Scenario G.63 through G.65 listed in this subclause.

Activate the PICC as described in clause PICC activation process G.5.1.1, use CID = 0 and FSDI = 0,

For each step in Scenario do:

- a) Send the command as described in the PICC-test-apparatus column.
- b) Check that the PICC response matches the one of the PICC column.

Let n be an arbitrary value of a valid NAD with b4 and b8 set to (0)b.

Scenario G.63: NAD on I-block (for PICCs supporting NAD)

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND1(1), NAD = n) →	← I(0) ₀ (TEST_RESPONSE1(1), containing NAD)

Scenario G.64: NAD on chained I-block (for PICCs supporting NAD)

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND2(3), NAD = n)	→
		← I(1) ₀ (TEST_RESPONSE2(3) ₁ , containing NAD)
2	R(ACK) ₁	→
		← I(1) ₁ (TEST_RESPONSE2(3) ₂ , not containing NAD)

Scenario G.65: NAD on I-block (for PICCs not supporting NAD)

Step	PICC-test-apparatus	PICC
1	I(0) ₀ (TEST_COMMAND1(1), NAD = n)	→
		← Mute

G.5.5.3 Test report

Fill the appropriate rows in Table G.63 according to Table G.58.

Table G.58 — Result criteria for PICC reaction on NAD

Explanation	Test result
If the Scenario is not applicable for the PICC	Not applicable (N/A)
When the Scenario is applicable for the PICC and only when the PICC's response is as indicated in the procedure	PASS
Any other case	FAIL

G.6 Reported results

Table G.59 — Type A specific timing table

No	Parameter	ISO Reference	Minimum value	Maximum value	Measured value(s)
1	Frame delay time PCD to PICC (for REQA,WUPA, ANTICOLLISION, SELECT commands)	ISO/IEC 14443-3:—, 6.2.1.1	Last bit (1)b -> 1236/fc	Last bit (1)b -> 1236/fc + 0,4 μs	
			Last bit (0)b -> 1172/fc	Last bit (0)b -> 1172/fc + 0,4 μs	
2	RATS and Deactivation frame waiting time	ISO/IEC 14443-4:2008, 8.1	Last bit (1)b -> 1236/fc Last bit (0)b -> 1172/fc	65536/fc (~4,8 ms)	

No	Parameter	ISO Reference	Minimum value	Maximum value	Measured value(s)
3	Frame delay time PCD to PICC (for frames other than previous rows)	ISO/IEC 14443-3:—, 6.2.1.1 and ISO/IEC 14443-4:2008, 8.1	Last bit (1)b -> $1236/f_c$ Last bit (0)b -> $1172/f_c$	$(256/fs) \times 2^{FWI}$ (~302,06 μ s $\times 2^{FWI}$)	FWI = Max FDT =

NOTE All timing values are calculated for carrier frequency $f_c = 13,56$ MHz and bit rate = $f_c/128$ (~106 kbit/s).

Table G.60 — Type B specific timing table

No	Parameter	ISO Reference	Minimum value	Maximum value	Measured value(s)
1	SOF low	ISO/IEC 14443-3:—, 7.1.4	10 etu (~94,40 μ s)	11 etu (~103,83 μ s)	
2	SOF high	ISO/IEC 14443-3:—, 7.1.4	2 etu (~18,88 μ s)	3 etu (~28,32 μ s)	
3	EOF low	ISO/IEC 14443-3:—, 7.1.5	10 etu (~94,40 μ s)	11 etu (~103,83 μ s)	
4	Bit boundaries	ISO/IEC 14443-3:—, 7.1.1	(n - 1/8) etu	(n + 1/8) etu	
5	EGT PICC to PCD	ISO/IEC 14443-3:—, 7.1.2	0 μ s	19 μ s	
6	TR0 for ATQB	ISO/IEC 14443-3:—, 7.1.6	$64/fs$ (~75,52 μ s)	$256/fs$ (~302,06 μ s)	
7	TR1 for ATQB	ISO/IEC 14443-3:—, 7.1.6	$80/fs$ (~94,40 μ s)	$200/fs$ (~235,99 μ s)	
8	TR0 Not ATQB	ISO/IEC 14443-3:—, 7.1.6 ISO/IEC 14443-3:—, 7.10.3	Value as defined in ISO/IEC 14443-3:—, Table 30	$(256/fs) \times 2^{FWI}$ (~302,06 μ s $\times 2^{FWI}$)	FWI = Max TR0 =
9	TR1 Not ATQB	ISO/IEC 14443-3:—, 7.1.6 ISO/IEC 14443-3:—, 7.10.3	Value as defined in ISO/IEC 14443-3:—, Table 31	$200/fs$ (~235,99 μ s)	
10	TR2	ISO/IEC 14443-3:—, 7.9.4.4	Value as defined in ISO/IEC 14443-3:—, Table 27	No maximum	
11	Delay from the end of EOF and Subcarrier off	ISO/IEC 14443-3:—, 7.1.7	0	2 etu	
12	Deactivation frame waiting time	ISO/IEC 14443-4:2008, 8.1	$64/fs + 80/fs$ (~169,92 μ s)	$65536/f_c$ (~4,8 ms)	

NOTE All timing values are calculated for carrier frequency $f_c = 13,56$ MHz and bit rate = $f_c/128$ (~106 kbit/s).

Table G.61 — Reported results for Type A specific test methods

Test method from ISO/IEC 10373-6		Scenario numbers	Test result
Clause	Parameter	ISO/IEC 10373-6	PASS or FAIL or N/A
G.3.2	Polling	Scenario G.1	
G.3.3	Testing of the PICC Type A state transitions	Scenario G.2	
		Scenario G.3	
		Scenario G.4	
		Scenario G.5	
		Scenario G.6	
		Scenario G.7	
		Scenario G.8	
		Scenario G.9	
		Scenario G.10	
		Scenario G.11	
Scenario G.12			
G.3.4	Handling of Type A anticollision	Scenario G.13	
G.3.6	Handling of PPS request	Scenario G.17	
		Scenario G.18	
		Scenario G.19	
G.3.7	Handling of FSD	Scenario G.20	

Table G.62 — Reported results for Type B specific test methods

Test method from ISO/IEC 10373-6		Scenario numbers	Test result
Clause	Parameter	ISO/IEC 10373-6	PASS or FAIL or N/A
G.4.2	Polling	Scenario G.21	
G.4.3	PICC Reception	Scenario G.22	
G.4.4	Testing of the PICC Type B state transitions	Scenario G.23	
		Scenario G.24	
		Scenario G.25	
		Scenario G.26	
		Scenario G.27	
G.4.5	Handling of Type B anticollision	Scenario G.28	
G.4.6	Handling of ATTRIB	Scenario G.29	
		Scenario G.30	
G.4.7	Handling of Maximum Frame Size	Scenario G.31	

Table G.63 — Reported results for test methods for logical operation of the PICC Type A or Type B

Test method from ISO/IEC 10373-6		Scenario numbers	Test result
Clause	Parameter	ISO/IEC 10373-6	PASS or FAIL or N/A
G.5.2	PICC reaction to ISO/IEC 14443-4 Scenarios	Scenario G.32	
		Scenario G.33	
		Scenario G.34	
		Scenario G.35	
		Scenario G.36	
		Scenario G.37	
		Scenario G.38	
		Scenario G.39	
		Scenario G.40	
		Scenario G.41	
		Scenario G.42	
		Scenario G.43	
		Scenario G.44	
		Scenario G.45	
		Scenario G.46	
		Scenario G.47	
		Scenario G.48	
		Scenario G.49	
		Scenario G.50	
		G.5.3	Handling of PICC error detection
Scenario G.56			
Scenario G.57			
G.5.4	PICC reaction on CID	Scenario G.58	
		Scenario G.59	
		Scenario G.60	
		Scenario G.61	
		Scenario G.62	
G.5.5	PICC reaction on NAD	Scenario G.63	
		Scenario G.64	
		Scenario G.65	
G.1.5.1	RFU values		

Annex H (normative)

Additional PCD test methods

H.1 PCD-test-apparatus and accessories

This clause defines the PCD-test-apparatus and test circuits for verifying the operation of the PCD according to ISO/IEC 14443-3:— and ISO/IEC 14443-4:2008.

H.1.1 Test method

The ISO/IEC 9646 abstract model is chosen and the local test method is used for the testing of the ISO/IEC 14443 protocol between the tested PCD and the LT.

H.1.2 PCD-test-apparatus structure

The PCD-test-apparatus consists of two parts (see Figure H.1):

- Upper Tester (may be a personal computer with a host interface suitable for a tested PCD);
- Lower Tester (LT).

Tested PCD is treated as Implementation Under Test (IUT).

When a PCD is embedded in a product, it includes the UT. For this case some tests may not be applicable. Also, in case the standard does not have a specific requirement the test method will end up in a report of capabilities only.

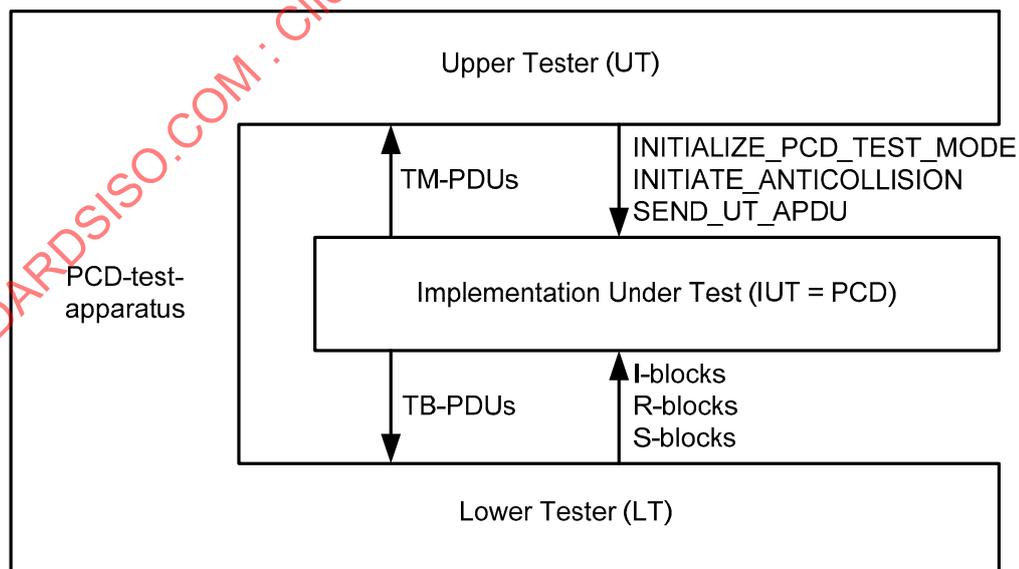


Figure H.1 — Conceptual tester architecture

The LT part of the PCD-test-apparatus includes:

- A PICC emulation hardware and software device capable of emulating of Type A and Type B protocols;
- A digital sampling oscilloscope (see 5.1.1).

H.1.3 PCD-test-apparatus interface

The UT and the IUT communicate with the TM-PDU (Test Management PDU). The definition of TM-PDUs is implementation dependent and provided by the IUT manufacturer and shall initiate the actions required in Table H.1.

Table H.1 — Logical interface commands

	TM-PDU name	Required IUT action
1	INITIALIZE_PCD_TEST_MODE	Return to Power On state (The IUT is expected to enter to anticollision loop). The IUT returns the result code of its action to the UT.
2	INITIATE_ANTICOLLISION	Initiate anticollision sequence (if the IUT starts the anticollision sequence automatically upon initialize, the sequence can be empty). The IUT returns the result code of its action to the UT.
3	SEND_UT_APDU	Transmit the UT_APDU through the RF interface to LT and return the IUT result code of its action to the UT. The response from the IUT shall include the answer of LT to the UT_APDU sent.

The PCD-test-apparatus shall be able to initialize the IUT utility information provided by the IUT manufacturer over the UT interface and to configure itself to perform the necessary procedures, protocols and analysis over its LT interface.

H.1.4 Emulating the I/O protocol

The PCD-test-apparatus at its LT interface shall be able to emulate the protocol Type A and Type B and PICC applications, which are required to run the Scenario. The LT shall be able to break the transmitted packets into chained blocks with the required length.

It shall be possible to configure the LT to simulate different options:

- NAD and CID configuration;
- Frame size, bit rates and any other parameter as required for the implementation of the test methods.

H.1.5 Generating the I/O character timing in transmission mode

The PCD-test-apparatus at its LT interface shall be able to generate the I/O bit stream according to ISO/IEC 14443-3:—. Timing parameters: start bit duration, extra guard time (Type B only), bit duration, frame delay time, start of frame width and end of frame width shall be configurable. For the purpose of tests of Type A, the LT shall be capable of simulating a bit collision at a selected bit position(s).

H.1.6 Measuring and monitoring the RF I/O protocol

The PCD-test-apparatus at its LT interface shall be able to measure and monitor the timing of the logical low and high states transmitted by the PCD.

H.1.7 Protocol Analysis

The PCD-test-apparatus shall be able to analyze the I/O-bit stream at its LT interface in accordance with protocol Type A and Type B as specified in ISO/IEC 14443-3:— and ISO/IEC 14443-4:2008 and extract the logical data flow for further protocol analysis.

H.1.8 Protocol activation procedure

H.1.8.1 Activation procedure for anticollision test methods

Activate the LT by the following sequence:

- a) Configure the LT to emulate the Type A or Type B protocol.
- b) The UT sends INITIALIZE_PCD_TEST_MODE TM-PDU to the PCD.
- c) The UT sends INITIATE_ANTICOLLISION TM-PDU to the PCD.

H.1.8.2 Activation procedure for Type A protocol test methods

Activate the LT by the following sequence:

- a) Configure the LT to emulate the Type A protocol.
- b) The UT sends INITIALIZE_PCD_TEST_MODE TM-PDU to the PCD.
- c) The UT sends INITIATE_ANTICOLLISION TM-PDU to the PCD. The PCD shall apply the anticollision sequence as defined in ISO/IEC 14443-3:—, Clause 6 (request, anticollision loop and select). The PCD shall apply the protocol activation sequence as defined in ISO/IEC 14443-4:2008, Clause 5.
- d) The PCD reports the UT the result of the activation procedure.

H.1.8.3 Activation procedure for Type B protocol test methods

Activate the LT by the following sequence:

- a) Configure the LT to emulate the Type B protocol.
- b) The UT sends INITIALIZE_PCD_TEST_MODE TM-PDU to the PCD.
- c) The UT sends INITIATE_ANTICOLLISION TM-PDU to the PCD. The PCD shall apply the anticollision sequence as defined in ISO/IEC 14443-3:—, Clause 7.
- d) The PCD reports the UT the result of the activation procedure.

H.1.9 Scenario

H.1.9.1 Description

Testing of the PCD as defined in this document requires a Scenario to be executed. This Scenario is a 'typical protocol and application specific communication', dependent on the protocol and application specific functionality foreseen for the normal use of and implemented in the PCD.

The typical Scenario is the set of command TM-PDUs defined in H.1.3.

The Scenario shall be defined by the entity carrying out these tests and shall be documented with the test results. The Scenario shall encompass a representative subset or if practical, the full functionality of the PCD expected to be utilized during normal use.

NOTE The testing entity may require information about the implemented protocol and functionality.

The UT_APDU to be sent may be one from the following:

- UT_TEST_COMMAND1, decided by the PCD-test-apparatus, specifies the ISO instruction used as the default instruction for Scenarios not needing PCD chaining. (In case PCD decides anyway to chain, the Scenario should be adapted accordingly by the test laboratory);
- UT_TEST_COMMAND2, decided by the PCD-test-apparatus, specifies the ISO instruction used as the default instruction for Scenarios dealing with PCD chaining.

H.1.9.2 Scenario example

The typical Scenario may be as follows:

```
INITIALIZE_PCD_TEST_MODE
INITIATE_ANTICOLLISION
SEND_UT_APDU (UT_TEST_COMMAND1)
SEND_UT_APDU (UT_TEST_COMMAND2)
...
```

H.1.10 UT, LT and PCD behavior

The following items summarize the behavior of the UT, the LT and the PCD:

- a) The UT runs the activation procedure as defined in H.1.8.
- b) If the activation procedure went wrong, the PCD goes to exception processing. This exception processing may include reporting the error to the UT.
- c) In case of anticollision test methods the PCD-test-apparatus ends the test at this point. For protocol test methods the UT continues to the next step.
- d) The UT sends the first command UT_APDU to the PCD.
- e) The PCD is expected to transfer this command UT_APDU to the LT using TB-PDUs. The PCD splits the current UT_APDU into the appropriate TB-PDUs (I-blocks), sends the first I-block to LT and response block is awaited. The PCD manages communication blocks according to ISO/IEC 14443-4:2008.
- f) The command UT_APDU is received by the LT. The LT sends the response UT_APDU to the PCD. The LT manages communication blocks (TB-PDUs) according to ISO/IEC 14443-4:2008 (the LT may use chaining mechanism at any time even if not mandated by either PCD or PICC maximum frame size). The PCD is expected to transfer response UT_APDU, received from the LT, back to the UT.
- g) If the command failed at protocol level (i.e. error detected by the PCD), the PCD goes to exception processing. Exception processing may include error reporting to the UT.
- h) If the command succeeded, the PCD reports the UT about successful result. In this case, if the Scenario defines additional UT_APDU to be sent to the LT, the UT sends the next UT_APDU to the PCD. This loop continues until the last test UT_APDU is sent.

H.1.11 Relationship of test methods versus base standard requirement

All tests in Tables H.2, H.3 and H.4 shall be executed and their results reported in the relevant Tables in H.6.

Table H.2 — Type A specific test methods

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause
H.2.1	Frame delay time PICC to PCD	ISO/IEC 14443-3:—	6.2.1.2
H.2.2	Request Guard Time	ISO/IEC 14443-3:—	6.2.2
H.2.3	Handling of bit collision during ATQA	ISO/IEC 14443-3:—	6.5.2
H.2.4	Handling of anticollision loop	ISO/IEC 14443-3:—	6.5.3
H.2.5	Handling of RATS and ATS	ISO/IEC 14443-4:2008	5.6.1.1
H.2.6	Handling of PPS response	ISO/IEC 14443-4:2008	5.6.2.1
H.2.7	Frame size selection mechanism	ISO/IEC 14443-4:2008	5.2.3
H.2.8	Handling of Start-up Frame Guard Time	ISO/IEC 14443-4:2008	5.2.5
H.2.9	Handling of the CID during activation by the PCD	ISO/IEC 14443-4:2008	5.6.3

Table H.3 — Type B specific test methods

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause
H.3.1	I/O transmission timing	ISO/IEC 14443-3:—	7.1
H.3.2	Frame size selection mechanism	ISO/IEC 14443-3:—	7.9
H.3.3	Handling of the CID during activation by the PCD	ISO/IEC 14443-3:—	7.10

Table H.4 — Test methods for logical operation

Test method from ISO/IEC 10373-6		Corresponding requirement	
Clause	Name	Base standard	Clause
H.4.1	Handling of the polling loop	ISO/IEC 14443-3:—	5
H.4.2	Reaction of the PCD to request for waiting time extension	ISO/IEC 14443-4:2008	7.3
H.4.3	Error detection and recovery	ISO/IEC 14443-4:2008	7.5.6
H.4.4	Handling of NAD during chaining	ISO/IEC 14443-4:2008	7.1.1.3

H.2 Type A specific test methods

H.2.1 Frame delay time PICC to PCD

The purpose of this test is to determine the timing between a PICC frame and the next PCD frame.

H.2.1.1 Apparatus

See H.1.

H.2.1.2 Procedure

Place the LT into the PCD operating volume.

During the following procedure the RF Input/Receive data shall be continuously monitored and verified correct to ISO/IEC 14443-2:2010. All signal transitions (level and timing) as well as the logical content of the communication shall be recorded.

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame.
- c) The LT answers with a valid ATQA.
- d) The LT waits until the PCD sends a valid Anticollision command according to Figure 7 in ISO/IEC 14443-3:—.
- e) Measure the time between the last modulation transmitted by the LT and the first pause transmitted by the PCD (see ISO/IEC 14443-3:—, 6.2.1.2).

H.2.1.3 Test report

Report the signal recording. Fill item 1 of Table H.21 with measured value of frame delay time and the appropriate row of Table H.23.

H.2.2 Request Guard Time

The purpose of this test is to determine the Request Guard Time of two consecutive REQA/WUPA commands. This test is relevant for PCDs, which send consecutive REQA/WUPA.

H.2.2.1 Apparatus

See H.1.

H.2.2.2 Procedure

Place the LT into the PCD operating volume.

During the following procedure the RF Input/Receive data shall be continuously monitored and verified correct to ISO/IEC 14443-2:2010. All signal transitions (level and timing) as well as the logical content of the communication shall be recorded.

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame. The LT remains Mute.
- c) The LT waits until the PCD sends a valid REQA/WUPA command frame. The LT remains Mute.
- d) Measure the time between the start bits of two consecutive REQA/WUPA (see ISO/IEC 14443-3:—, 6.2.2).

H.2.2.3 Test report

Report the signal recording. Fill item 2 in Table H.21 with measured value of request guard time and the appropriate row of Table H.23.

H.2.3 Handling of bit collision during ATQA

The purpose of this test is to determine the handling of bit collision during ATQA by the PCD.

H.2.3.1 Apparatus

See H.1.

H.2.3.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame.
- c) Maintain the LT to answer with ATQA using simulation of the bit collision at bit N (N from 1 up to 16). Collision at a bit causes a collision also in associated parity bit.

H.2.3.3 Test report

Fill the appropriate row in Table H.23 according to Table H.5.

Table H.5 — Result criteria for handling of bit collision during ATQA

Explanation	Test result
Only when the PCD starts the bit oriented anticollision loop	PASS
Any other case	FAIL

H.2.4 Handling of anticollision loop

The purpose of this test is to determine the handling of bit anticollision loop according to ISO/IEC 14443-3:—, 6.5.3.

H.2.4.1 Apparatus

See H.1.

H.2.4.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

H.2.4.2.1 Procedure 1 (single size UID)

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame.
- c) The LT answers with ATQA indicating bit frame anticollision and UID size: single (bits b8 and b7 equal (00)b).
- d) The PCD shall send ANTICOLLISION command '93 20' (cascade level 1).
- e) The LT answers with UID CL1 (uid0 uid1 uid2 uid3 BCC).
- f) The PCD shall send SELECT command '93 70' uid0 uid1 uid2 uid3 BCC CRC_A.
- g) The LT answers with SAK (cascade bit is cleared, b3 = (0)b), indicating that UID is complete.

Scenario H.1 — Handling of anticollision loop for PICC with single size UID (Procedure 1)

Test	PCD	LT	Stage
REQA/WUPA	REQA/WUPA	→	1
		← ATQA (single size UID)	
ANTICOLLISION Level 1	ANTICOLLISION command Level 1 ('93 20')	→	2
		← UID CL1 (uid0 uid1 uid2 uid3 BCC)	
SELECT	SELECT command ('93 70' uid0 uid1 uid2 uid3 BCC CRC_A)	→	3
		← SAK(complete)	

H.2.4.2.2 Procedure 2 (double size UID)

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame.
- c) The LT answers with ATQA indicating bit frame anticollision and UID size: double (bits b8 and b7 equal (01)b).
- d) The PCD shall send ANTICOLLISION command '93 20' (cascade level 1).
- e) The LT answers with UID CL1 ('88' uid0 uid1 uid2 BCC).
- f) The PCD shall send SELECT command '93 70 88' uid0 uid1 uid2 BCC CRC_A.
- g) The LT answers with SAK (cascade bit is set, b3 = (1)b).
- h) The PCD shall increase the cascade level and shall send ANTICOLLISION command '95 20' (cascade level 2).
- i) The LT answers with UID CL2 (uid3 uid4 uid5 uid6 BCC).
- j) The PCD shall send SELECT command '95 70' uid3 uid4 uid5 uid6 BCC CRC_A.

k) The LT answers with SAK (cascade bit is cleared, b3 = (0)b), indicating that UID is complete.

Scenario H.2 — Handling of anticollision loop for PICC with double size UID (Procedure 2)

Test	PCD	LT	Stage
REQA/WUPA	REQA/WUPA	→	1
		← ATQA (double size UID)	
ANTICOLLISION Level 1	ANTICOLLISION command Level 1 ('93 20')	→	2
		← UID CL1 ('88' uid0 uid1 uid2 BCC)	
SELECT	SELECT command ('93 70 88' uid0 uid1 uid2 BCC CRC_A)	→	3
		← SAK(cascade)	
ANTICOLLISION Level 2	ANTICOLLISION command Level 2 ('95 20')	→	4
		← UID CL2 (uid3 uid4 uid5 uid6 BCC)	
SELECT	SELECT command ('95 70' uid3 uid4 uid5 uid6 BCC CRC_A)	→	5
		← SAK(complete)	

H.2.4.2.3 Procedure 3 (triple size UID)

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA/WUPA command frame.
- c) The LT answers with ATQA indicating bit frame anticollision and UID size: triple (bits b8 and b7 equal (10)b).
- d) The PCD shall send ANTICOLLISION command '93 20' (cascade level 1).
- e) The LT answers with UID CL1 ('88' uid0 uid1 uid2 BCC).
- f) The PCD shall send SELECT command '93 70 88' uid0 uid1 uid2 BCC CRC_A.
- g) The LT answers with SAK (cascade bit is set, b3 = (1)b).
- h) The PCD shall increase the cascade level and shall send ANTICOLLISION command '95 20' (cascade level 2).
- i) The LT answers with UID CL2 ('88' uid3 uid4 uid5 BCC).
- j) The PCD shall send SELECT command '95 70 88' uid3 uid4 uid5 BCC CRC_A.
- k) The LT answers with SAK (cascade bit is set, b3 = (1)b).
- l) The PCD shall increase the cascade level and shall send ANTICOLLISION command '97 20' (cascade level 3).

- m) The LT answers with UID CL3 (uid6 uid7 uid8 uid9 BCC).
- n) The PCD shall send SELECT command '97 70' uid6 uid7 uid8 uid9 BCC CRC_A.
- o) The LT answers with SAK (cascade bit is cleared, b3 = (0)b), indicating that UID is complete.

Scenario H.3 — Handling of anticollision loop for PICC with triple size UID (Procedure 3)

Test	PCD	LT	Stage
REQA/WUPA	REQA/WUPA	→	
		← ATQA (triple size UID)	1
ANTICOLLISION Level 1	ANTICOLLISION command Level 1 ('93 20')	→	
		← UID CL1 ('88' uid0 uid1 uid2 BCC)	2
SELECT	SELECT command ('93 70 88' uid0 uid1 uid2 BCC CRC_A)	→	
		← SAK(cascade)	3
ANTICOLLISION Level 2	ANTICOLLISION command Level 2 ('95 20')	→	
		← UID CL2 ('88' uid3 uid4 uid5 BCC)	4
SELECT	SELECT command ('95 70 88' uid3 uid4 uid5 BCC CRC_A)	→	
		← SAK(cascade)	5
ANTICOLLISION Level 3	ANTICOLLISION command Level 3 ('97 20')	→	
		← UID CL3 (uid6 uid7 uid8 uid9 BCC)	6
SELECT	SELECT command ('97 70' uid6 uid7 uid8 uid9 BCC CRC_A)	→	
		← SAK(complete)	7

H.2.4.2.4 Procedure 4 (Full Bitwise Anticollision, single size UID)

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT waits until the PCD sends a valid REQA or WUPA command frame.
- c) The LT answers with ATQA indicating bit frame anticollision and UID size: single (bits b8 and b7 equal (00)b).
- d) The PCD shall send ANTICOLLISION command: '93 20'.
- e) The LT answers by a stream of 40 bits by emulating a collision on every bit, including parity bits.
- f) Repeat the steps g) to h) for values k from 1 to 31.
- g) The PCD shall send ANTICOLLISION command: '93' NVB UIDTX₁[[1..k]], where UIDTX₁[[1..k-1]] is either empty (i.e. k = 1) or the value already known by the PCD and UIDTX₁[[k]] is an arbitrary bit selected by the PCD.

- h) The LT answers by a stream of 40 minus k bits by emulating a collision on every bit, including parity bits.
- i) The PCD may optionally send ANTICOLLISION command: '93 60' UIDTX₁[[1..32]]. In this case the LT answers with BCC.

NOTE This optional ANTICOLLISION command does not change the Stage number.

- j) The PCD shall send SELECT command '93 70' UIDTX₁[[1..32]] BCC CRC_A, with BCC calculated by the PCD if it has not run the optional step i).
- k) The LT answers with SAK (cascade bit is cleared, b3 = (0)b), indicating that UID is complete.

Scenario H.4 — Handling of full bitwise anticollision loop for PICC (Procedure 4)

Test	PCD	LT	Stage
REQA/WUPA	REQA/WUPA	→	
		← ATQA (single size UID)	1
ANTICOLLISION	ANTICOLLISION command ('93 20')	→	
		← 40 bits full collision frame	2
ANTICOLLISION (k bits UID _{PARTIAL}) 1 ≤ k ≤ 31	ANTICOLLISION command ('93' NVB UIDTX ₁ [[1..k]])	→	
		← 40 minus k bits collision frame	k+2
OPTIONAL ANTICOLLISION (32 bits UID _{PARTIAL})	ANTICOLLISION command ('93 60' UIDTX ₁ [[1..32]])	→	
		← (BCC)	
SELECT	SELECT command ('93 70' UIDTX ₁ [[1..32]] BCC CRC_A)	→	
		← SAK(complete)	34

H.2.4.3 Test report

Fill the appropriate row in Table H.23 according to Table H.6.

Table H.6 — Result criteria for handling of anticollision loop

Explanation	Test result
Only when the PCD's behavior matches each procedure expected Scenario	PASS
Any other case	FAIL

H.2.5 Handling of RATS and ATS

The purpose of this test is to determine the handling of RATS and ATS by the PCD according to ISO/IEC 14443-4:2008, 5.6.1.1.

H.2.5.1 Apparatus

See H.1.

H.2.5.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

H.2.5.2.1 Procedure 1

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT answers relevant anticollision messages and waits until the PCD sends a valid RATS command frame.
- c) The LT does not respond to RATS (Mute).
- d) The PCD may send a valid RATS command frame.
- e) If the PCD has sent a second RATS, the LT does not respond to the RATS (Mute).
- f) The PCD shall start the deactivation sequence defined in ISO/IEC 14443-4:2008, Clause 8.
- g) Repeat the procedure with an erroneous ATS frame (use a wrong CRC_A) instead of Mute.

Scenario H.5 — Handling of RATS and ATS, Procedure 1

Test	PCD	LT
Mute	RATS command frame (e.g.'E0 01' CRC_A)	→
		←
	RATS command frame (e.g.'E0 01' CRC_A)	→
		←
start deactivation	DESELECT	→
erroneous ATS frame	RATS command frame (e.g.'E0 01' CRC_A)	→
		←
	RATS command frame (e.g.'E0 01' CRC_A)	→
		←
start deactivation	DESELECT	→
^a Determined in step g).		

H.2.5.2.2 Procedure 2

Use the following sequence:

- a) The UT performs the activation procedure according to H.1.8.1.
- b) The LT answers relevant anticollision messages and waits until the PCD sends a valid RATS command frame.
- c) The LT answers with a valid ATS without TA byte.