ISO/IEC JTC1/SC17 2010-04-12
Cards and personal identification

ISO/IEC JTC1/SC17 N 3897 **WG8 N 1663**

DOCUMENT TYPE: TEXT FOR FDIS BALLOT

TITLE: Notification that - ISO/IEC FDIS 10373-6 - Identification cards - Test methods - Part 6: Proximity cards – has been posted to the ISO server for FDIS ballot

BACKWARD POINTER: N 3502, N 3542, N 3627, N 3528, N 3652, N3729, N 3730, N 3821 and N 3896.

SOURCE: SECRETARIAT ISO/IEC JTC1/SC17

STATUS: Notification of FDIS ballot

ACTION ID: LB

WORK ITEM: 52652

DUE DATE: To be advised by ISO

DISTRIBUTION P, L and O-Members of ISO/IEC JTC1/SC17
JTC1 Secretariat
ISO/IEC ITTF

MEDIUM: SERVER

NO. OF PAGES: 207

Secretariat ISO/IEC JTC1/SC17, APACS, Mercury House, Triton Court, 14 Finsbury Square, London EC2A 1LQ, England;
Telephone +44 (0)20 7711 6292; Fax: +44 (0)20 7711 6299; e-mail: chris.starr@apacs.org.uk

ISO/IEC Form 10 - Electronic

	Explanatory Report	ISO/IEC FDIS
	ISO/IEC JTC 1/SC17 **N 3897** Will supersede: SC 17 N 3730	Secretariat: APACS for BSI

This form should be sent to ITTF, together with the committee draft, by the secretariat of the joint technical committee or sub-committee concerned

The accompanying document is submitted for circulation to member body vote as a FDIS, following consensus of the P-members of the committee obtained on:

	at the {DATE, LOCATION} meeting of ISO/IEC JTC 1/SC {YY} (See resolution number {XX} in document SC {YY} N {XXXXX})
✓	by postal ballot initiated on: 2009-07-28.

P-members in favour:	Armenia (SARM), Austria (ASI), China (SAC), Czech Republic (UNMZ), Denmark (DS), France (AFNOR), Germany (DIN), India (BIS), Italy (UNI), Japan (JISC), Korea, Republic of (KATS), Netherlands (NEN), Norway (SN), Poland (PKN), Portugal (IPQ), Romania (ASRO), Singapore (SPRING SG), South Africa (SABS), Switzerland (SNV), USA (ANSI)
P-members voting against:	Kenya (KEBS), Russian Federation (GOST R), United Kingdom (BSI)
P-members abstaining:	Australia (SA), Belgium (NBN), Canada (SCC), Finland (SFS), Israel (SII), Malaysia (DSM) Slovakia (SUTN), Spain (AENOR), Sweden (SIS)
P-members who did not vote:	

Remarks:

Resolution of comments contained in 17n3896.

Project: 31438

I hereby confirm that this draft meets the requirements of part 2 of the IEC/ISO Directives

Date:	Name/Signature of the secretary:
2010-04-12	Chris Starr

ISO/IEC JTC 1/SC 17/WG8 N 1621

Date: 2010-03-14

ISO/IEC FDIS 10373-6:2010(E)

ISO/IEC JTC 1/SC 17/WG 8

Secretariat: DIN

Identification cards — Test methods — Part 6: Proximity cards

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

Document type: International Standard
Document subtype:
Document stage: (50) Approval
Document language: E

C:\Userdata\Standards\10373-6\Editing Valence\Final\wg8N1621_Text_of_FDIS_10373-6.doc STD Version 2.1c2

Contents

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 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 including ISO/IEC 10373-6:2001/Amd.1:2007(E), ISO/IEC 10373-6:2001/Amd.2:2003(E), ISO/IEC 10373-6:2001/Amd.3:2006(E), ISO/IEC 10373-6:2001/Amd.4:2006(E) and ISO/IEC 10373-6:2001/Amd.5:2007(E). All clauses have been technically revised.

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

— Part 1: General characteristics tests

— Part 2: Cards with magnetic stripes

— Part 3: Integrated circuit(s) cards with contacts and related interface devices

— Part 5: Optical memory cards

— Part 6: Proximity cards

— Part 7: Vicinity cards

— Part 8: USB-ICC

The annexes B and F of this part of ISO/IEC 10373 are for information only.

Introduction

This part of ISO/IEC 10373 describes test methods applicable to Proximity cards and objects and Proximity coupling devices as defined in ISO/IEC 14443.

Identification cards — Test methods — Part 6: Proximity cards

1 Scope

This International Standard 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 may be ISO/IEC 7810 or one or more of the supplementary standards that define the information storage technologies employed in identification cards applications.

NOTE 1 Criteria for acceptability do not form part of this International Standard but will be found in the International Standards mentioned above.

NOTE 2 Test methods described in this International Standard 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 deals with 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 14443-3:2010 and ISO/IEC 14443-4:2008. Part 1 of the standard, General characteristics, deals with test methods which are common to one or more ICC technologies and other parts deal with other technology-specific tests.

2 Normative reference(s)

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 14443-3:2010, *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 61000-4-2:1995, *Electromagnetic compatibility (EMC) − Part 4: Testing and measurement techniques − Clause 2: Electrostatic discharge immunity test*

ISO/IEC Guide 98-3:2008, *Uncertainty of measurement − Part 3: Guide to the expression of uncertainty in measurement*

3 Definitions, abbreviations and symbols

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 14443-3:2010, ISO/IEC 14443-4:2008 and the following apply.

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

3.1 Definitions

3.1.1
base standard
the standard which the test method is used to verify conformance to

3.1.2
CascadeLevels
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 herein

3.1.4
Command Set
set describing the PICC commands during initialization and anticollision

NOTE See ISO/IEC 14443-3:2010, 6.4 for PICC Type A and ISO/IEC 14443-3:2010, 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 14443-3:2010, 6.3 for PICC Type A and ISO/IEC 14443-3:2010, 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
a 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 Abbreviations and symbols

(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 14443-3:2010, 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
$I(c)_n$**(**inf **[**,CID = cid**] [**,NAD = nad**] [**,~CRC**])**	ISO/IEC 14443-4 I-block with chaining bit c∈{1,0}, block number n∈{1,0} and information field inf. By default no CID and no NAD will be transmitted. If CID = cid∈{0...15} is specified, it will be transmitted as second parameter. If NAD = nad∈{0...'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 document 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(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(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 x 16+cid CRC_A)
READY(l)	READY state in cascade level l, l ∈ {1, 2, 3}; e.g. READY(2) is a PICC cascade level 2
READY*(l)	READY* state in cascade level l, l ∈ {1, 2, 3}; e.g. READY*(2) is a PICC cascade level 2
REQB(N)	REQB command with N as defined in ISO/IEC 14443-3:2010, 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(l) answer with the cascade bit (bit 3) set to (1)b
SAK(complete)	the SELECT(l) 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(l)	SELECT command of cascade level l, 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 14443-3:2010, 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 ± 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).

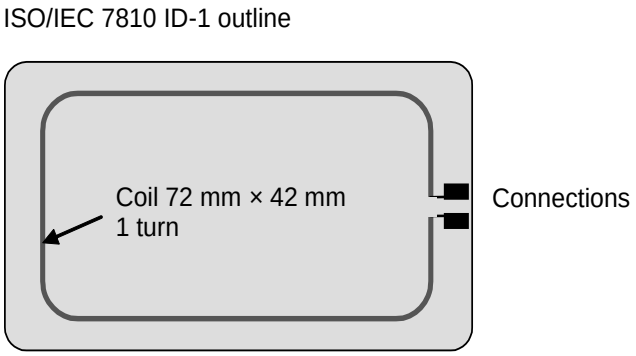


Figure 1 — Calibration coil

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 mm^2.

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.

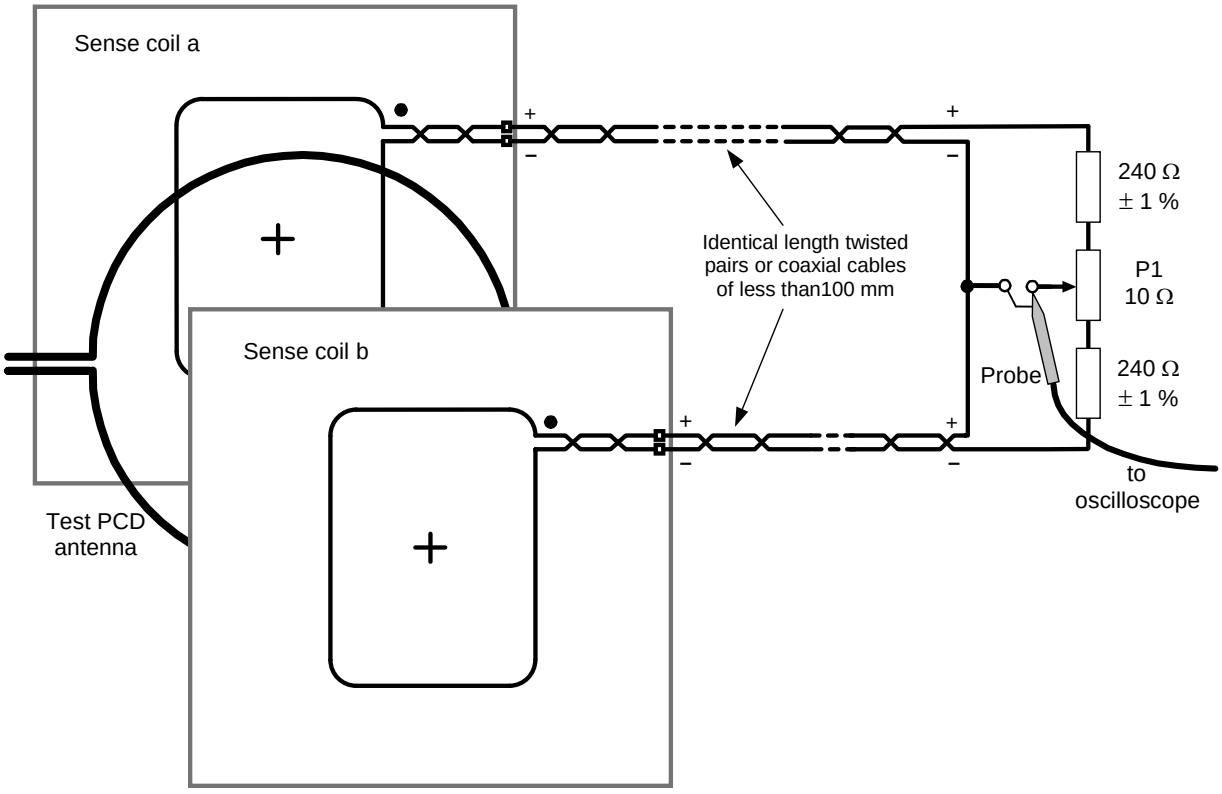


Figure 2 — Test set-up (principle)

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.

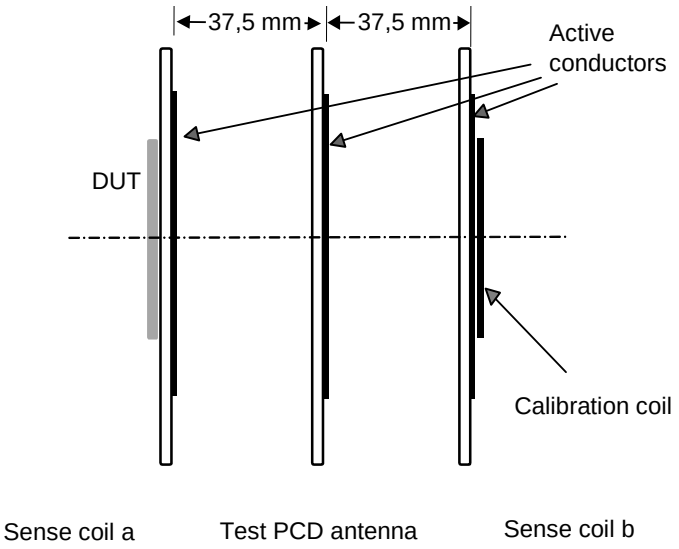


Figure 3 — Test PCD assembly

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.

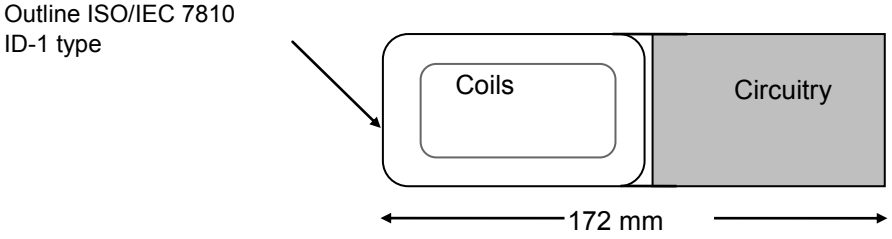


Figure 4 — Reference PICC dimensions

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.

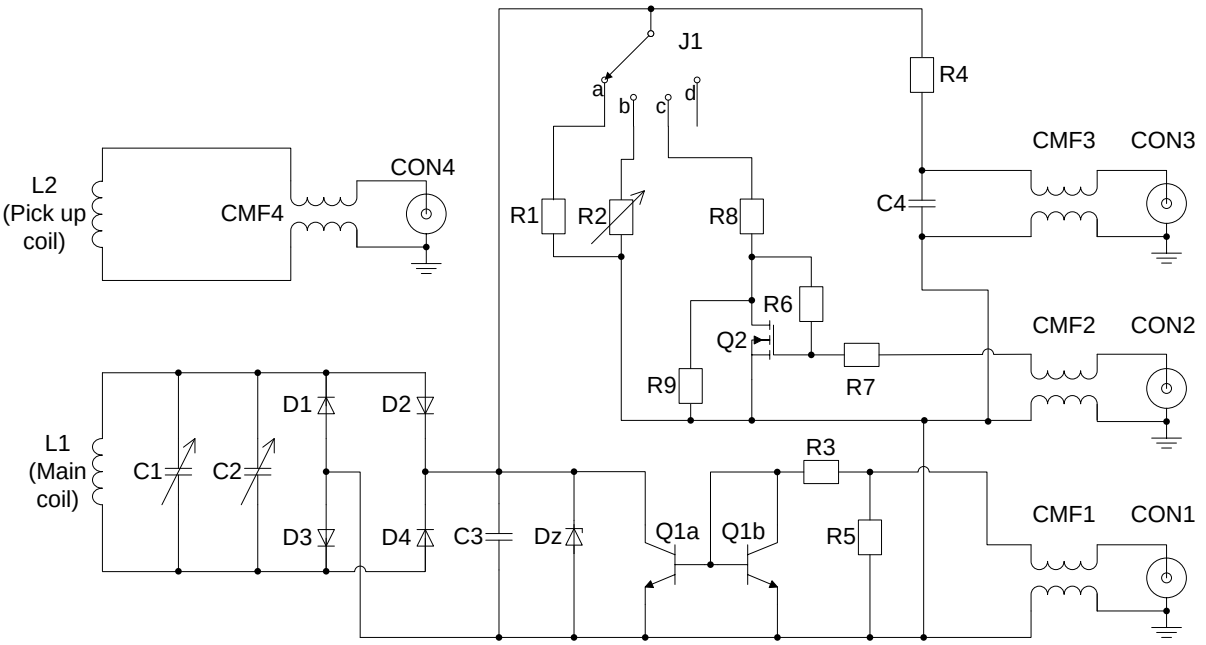


Figure 5 — Reference PICC circuit diagram

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 Cj (Junction capacitance), Cp (Package capacitance), Ls (Series inductance) and Rs (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.

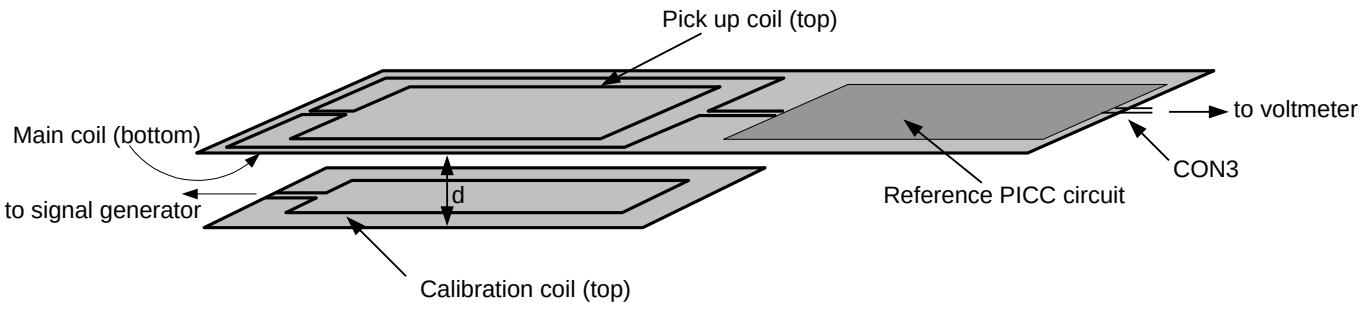


Figure 6 — Reference PICC frequency tuning set-up (principle)

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 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 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).

NOTE Static electricity test for PICC is expected to be updated in ISO/IEC 10373-1:2006/Amd.1 and it will not be specified at the next revision of ISO/IEC 10373-6.

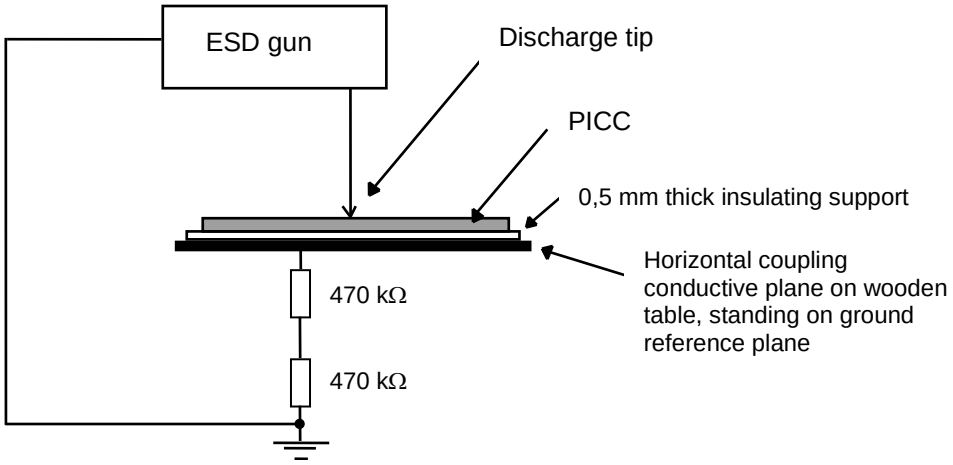


Figure 7 — ESD test circuit

6.2.2.1 Apparatus

Refer to IEC 61000-4-2:1995.

a) Main specifications of the ESD generator:

— energy storage capacitance: 150 pF ± 10 %;

— discharge resistance: 330 Ω ± 10 %;

— charging resistance: between 50 MΩ and 100 MΩ ;

— 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.

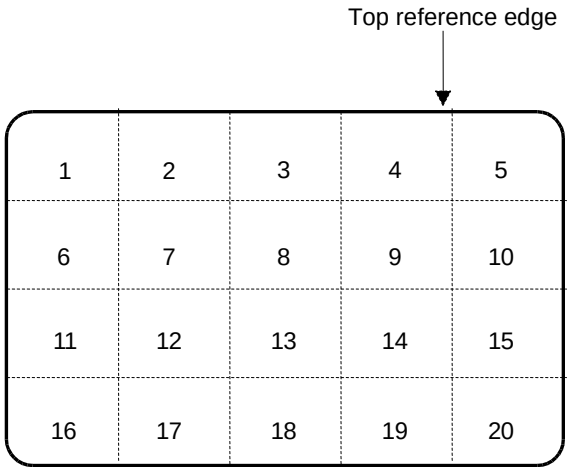


Figure 8 — Test zones on PICC for ESD test

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 14443-3:2010 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 14443-3:2010, 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 *fc* + *fs* and *fc* − *fs* 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 J.

8.2 PICC tests

See Annex G.

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.

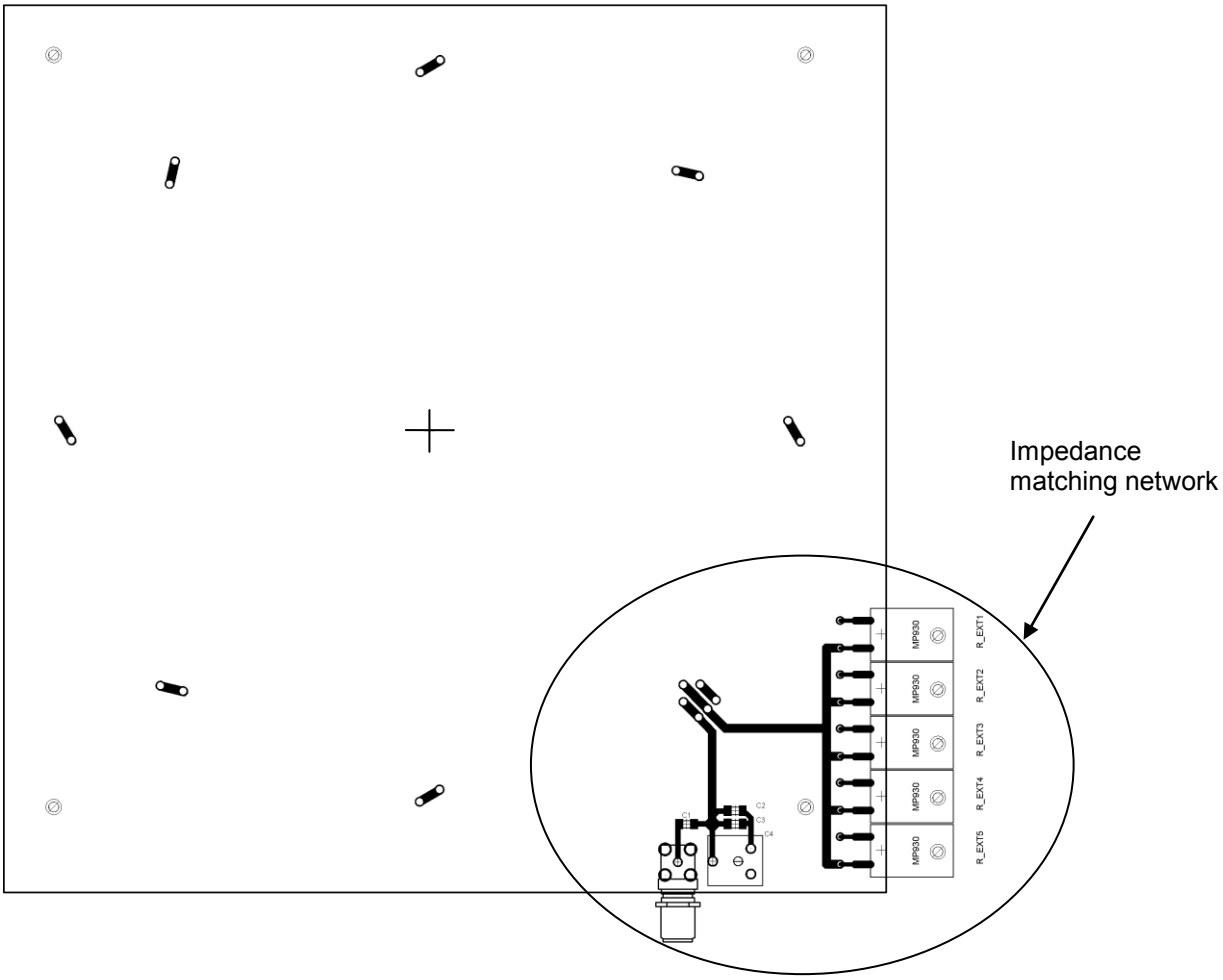


Figure A.1 — Test PCD antenna layout including impedance matching network for a bit rate of *fc*/128 (View from front)

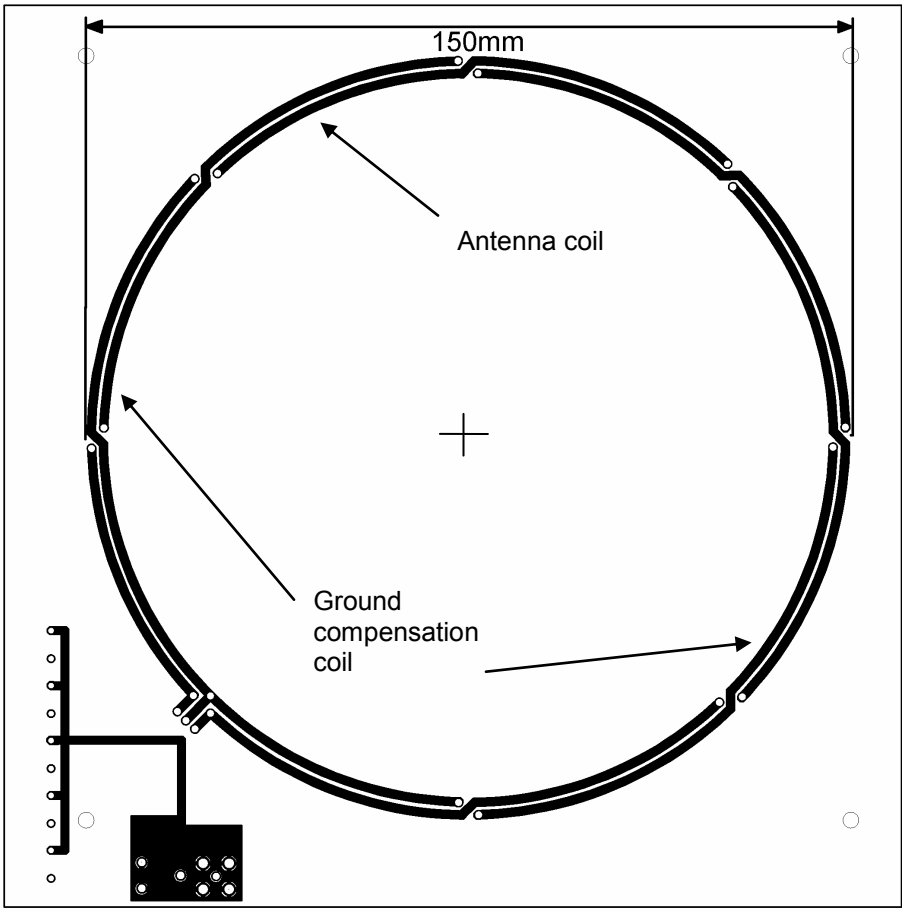


Figure A.2 — Test PCD antenna layout including impedance matching network for a bit rate of *fc*/128 (View from back)

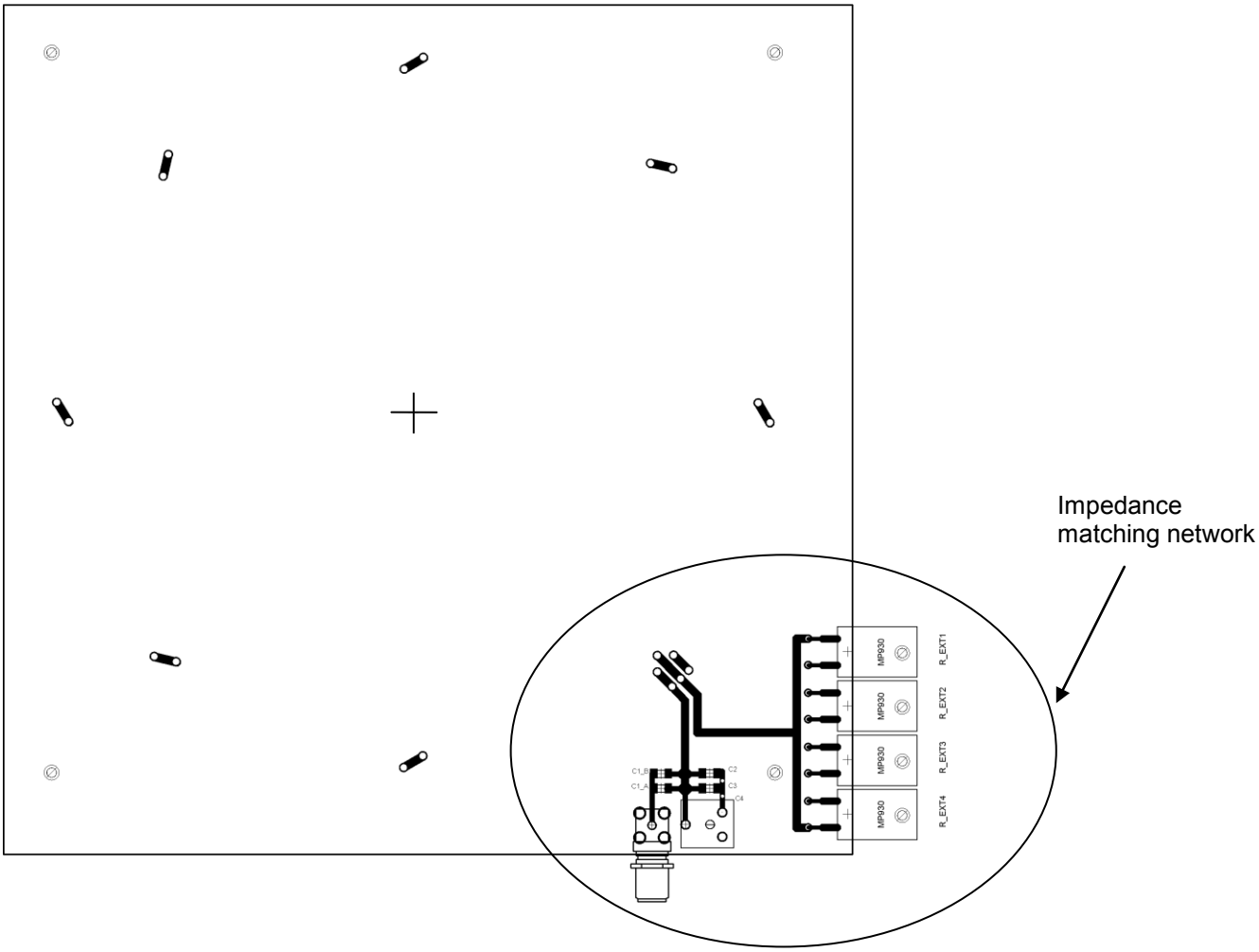


Figure A.3 — Test PCD antenna layout including impedance matching network for bit rates of *fc*/64, *fc*/32 and *fc*/16 (View from front)

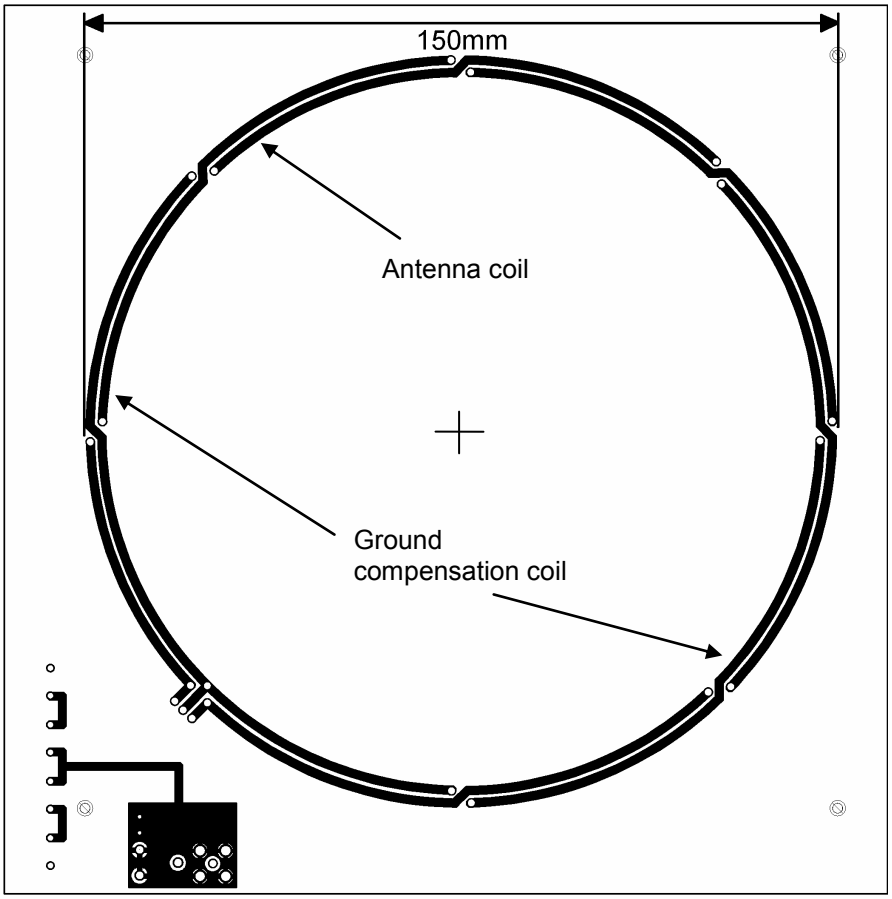


Figure A.4 — Test PCD antenna layout including impedance matching network for bit rates of *fc*/64, *fc*/32 and *fc*/16 (View from back)

A.2 Impedance matching network

The antenna impedance (R_{ant}, L_{ant}) is adapted to the signal generator output impedance (Z = 50 Ω) 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

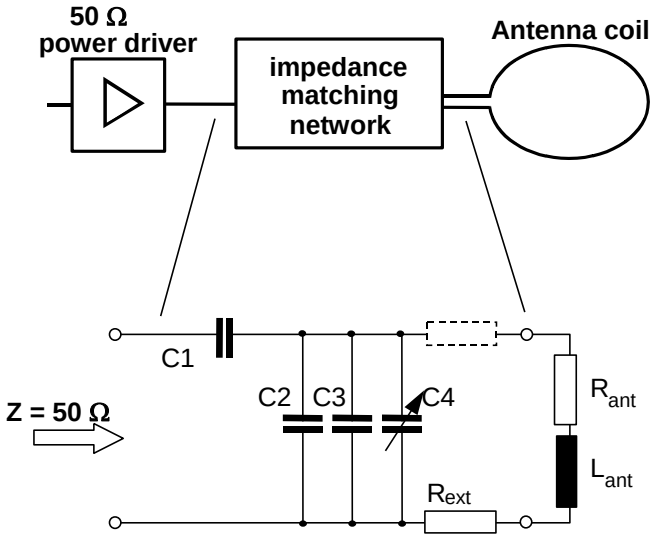
NOTE 1 The tolerance of the matched antenna impedance is ± 5 Ω and ± 10 °.

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 50 Ω 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

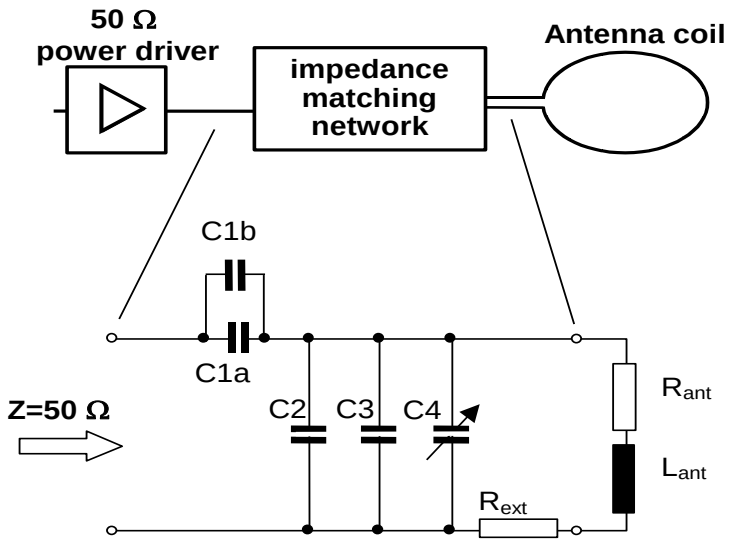
NOTE 1 R_{ext} may be built by connecting five resistors of 4,7 Ω 2 W in parallel.

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 Ω 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 Ω ± 1 % (e.g. 50 Ω 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 Ω ± 1 % (e.g. 50 Ω 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 $C_{cal} = C_{probe}$. The probe capacitor is compensated when the Lissajous figure is completely closed.

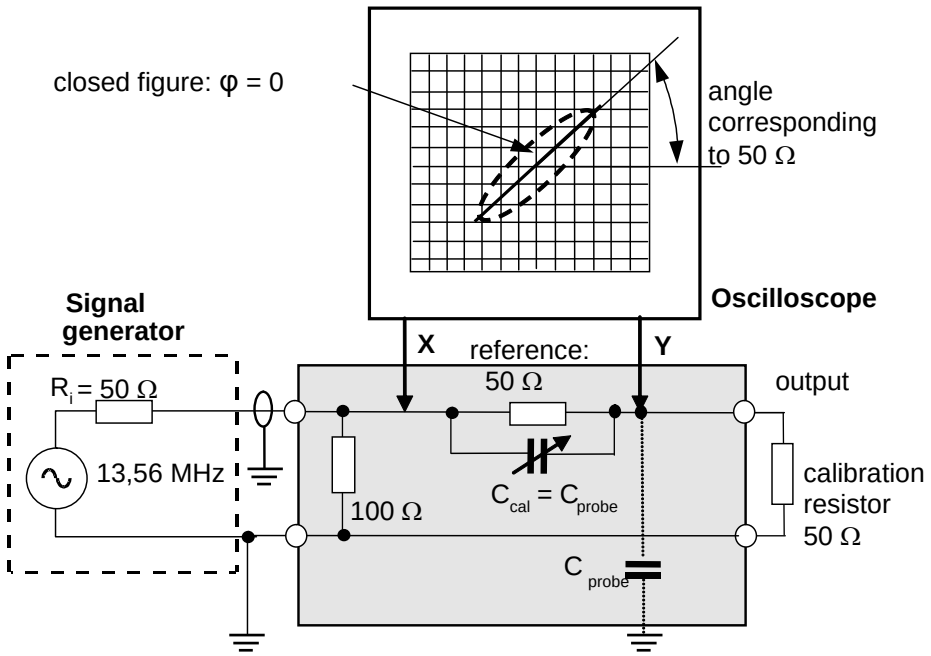


Figure B.1 — Calibration set-up (Step 1)

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.

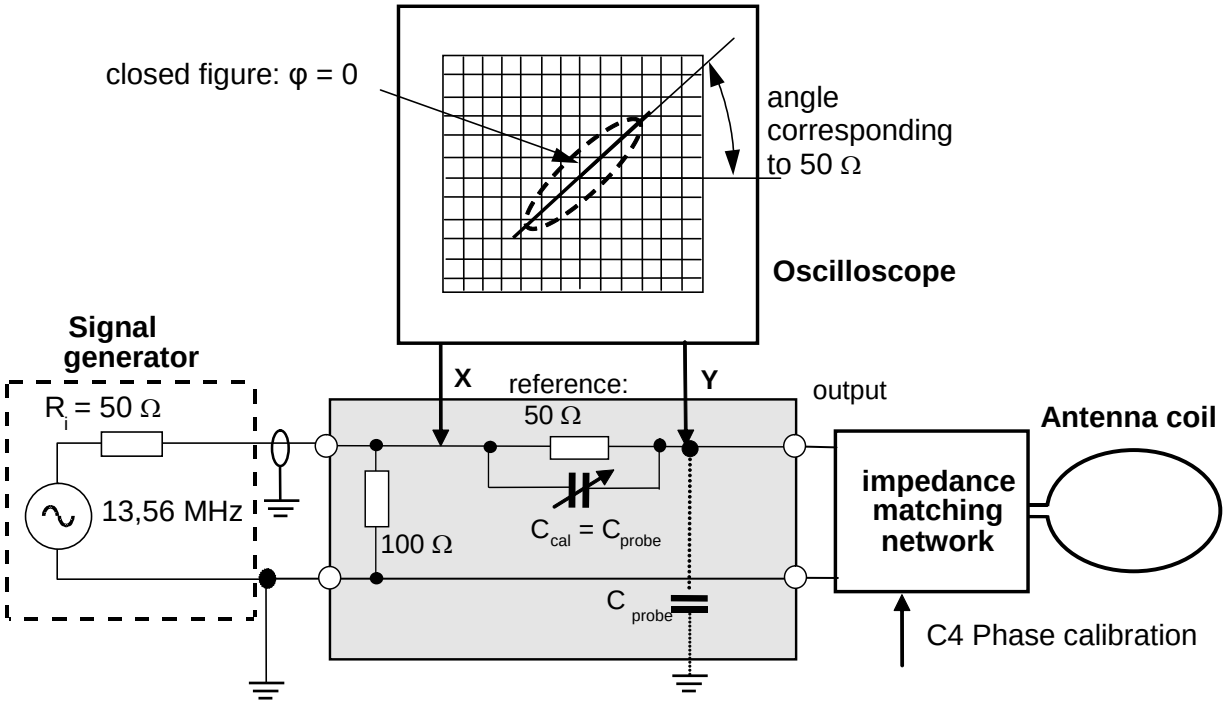


Figure B.2 — Calibration set-up (Step 2)

Annex C
(normative)

Sense coil

C.1 Sense coil layout

Figure C.1 illustrates sense coil layout.

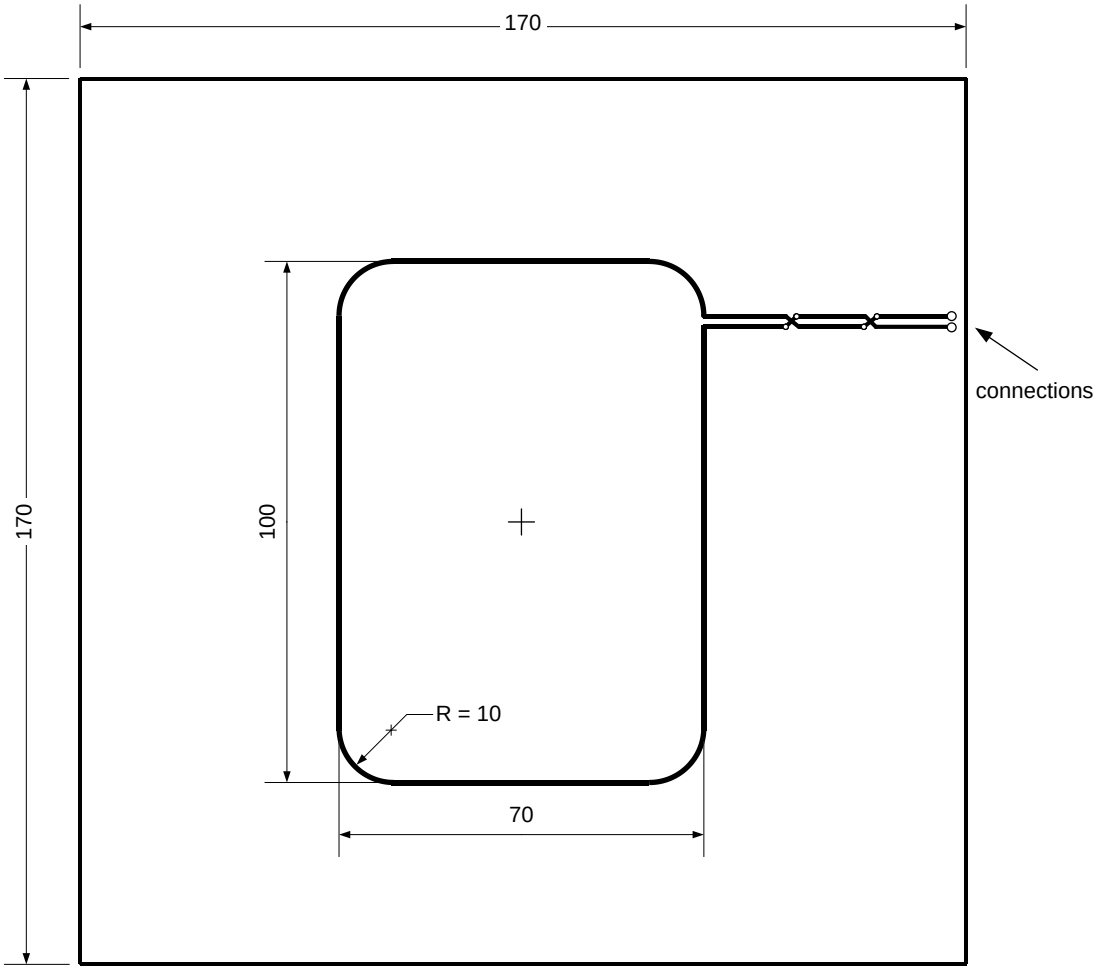


Figure C.1 — Layout for sense coils a and b

Dimensions in millimeters (drawings are not to scale).

The sense coil track width is 0,5 mm with relative tolerance ± 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.

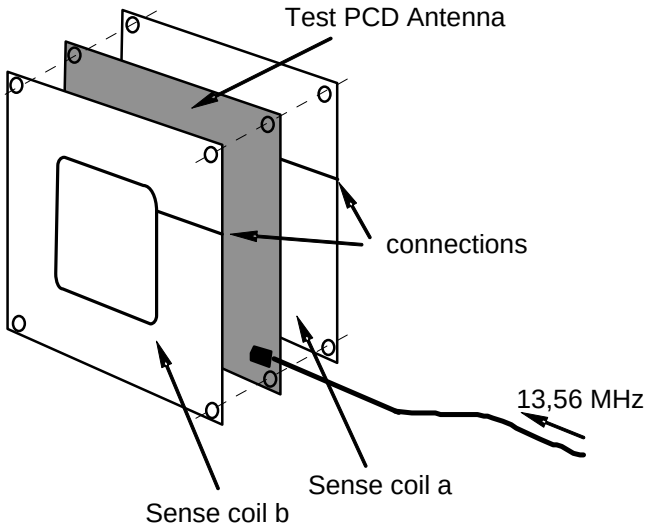


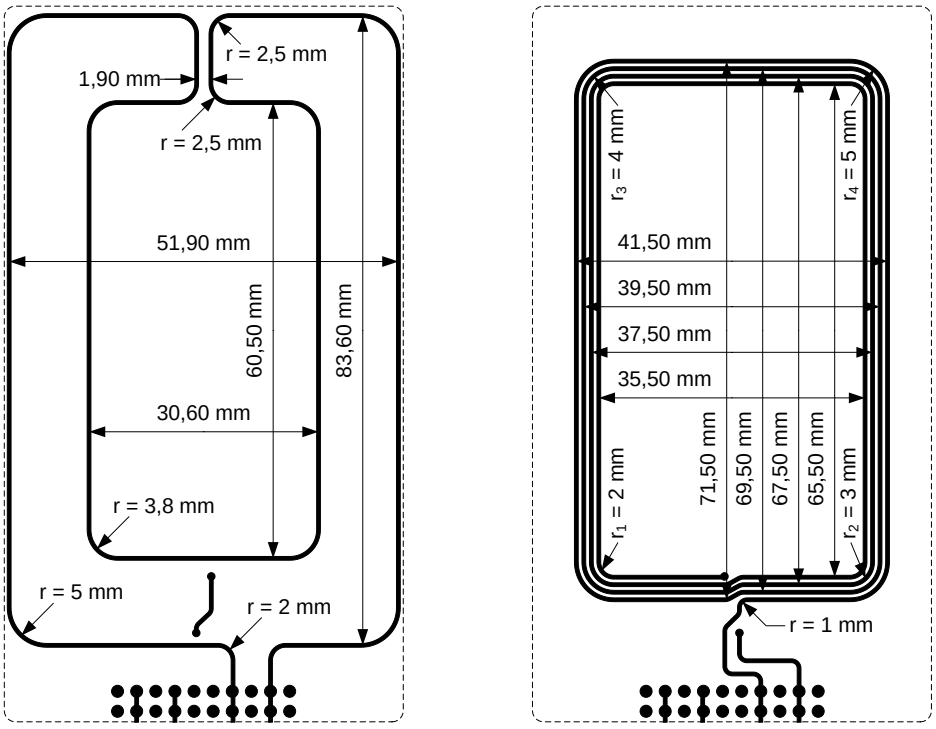
Figure C.2 — Sense coil assembly

Annex D
(normative)

Reference PICC

D.1 Circuit diagram

Figure D.1 illustrates Reference PICC antenna layout.



(View from front, Pick up coil) (View from back, Main coil)

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 ± 20 %.

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

NOTE 1 At 13,56 MHz the inductance of the Main coil L1 is 2,3 µH ± 10 % and the resistance is 1,8 Ω ± 10 %.

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

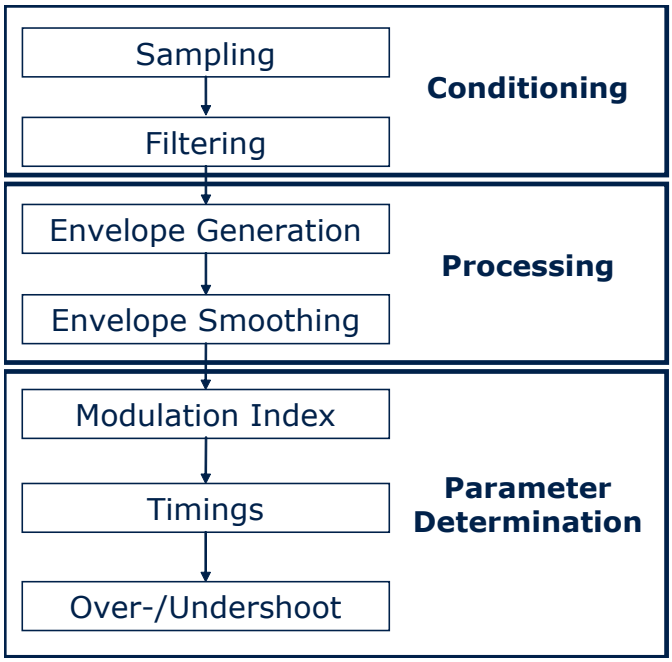


Figure E.1 — Modulation index and waveform analysis tool block diagram

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.

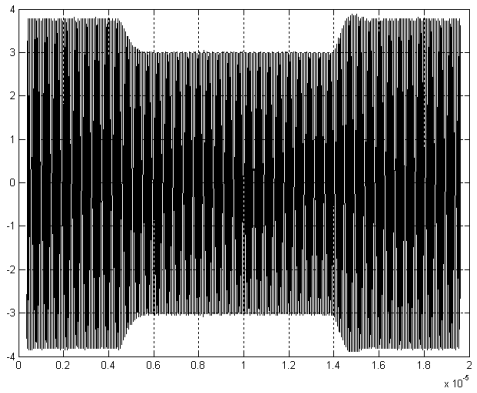


Figure E.2 — Modulation pulse

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.

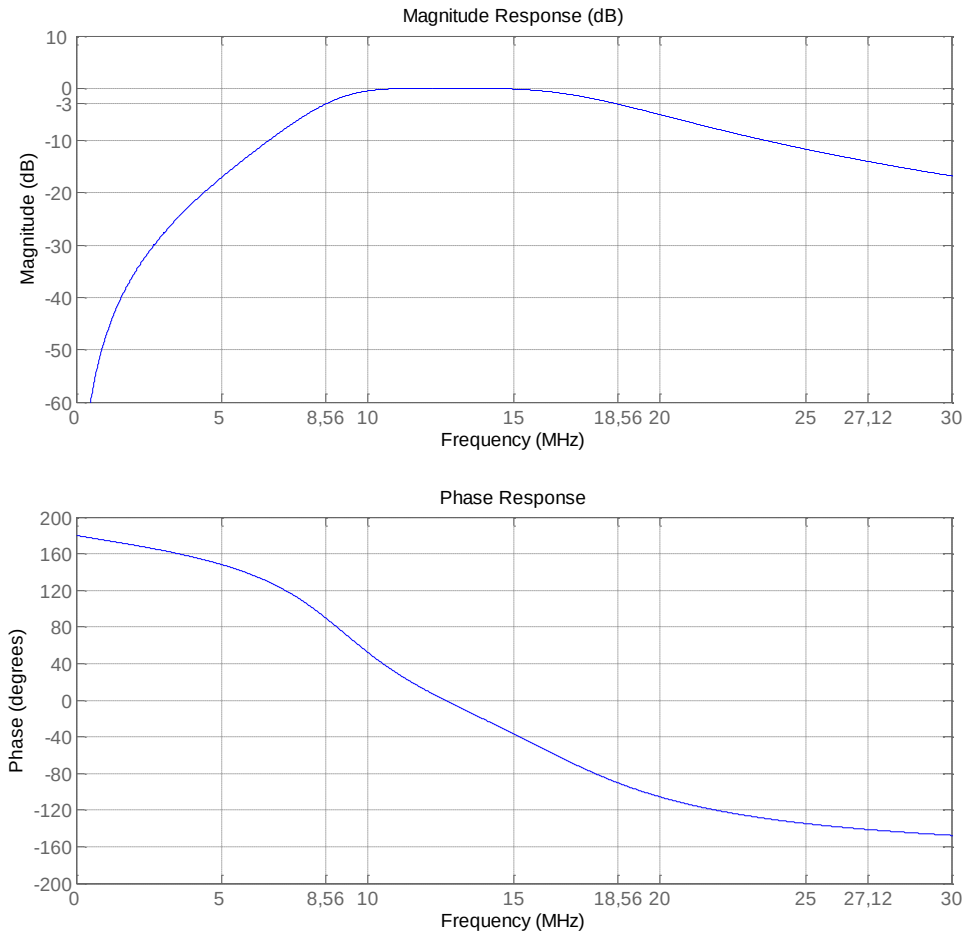


Figure E.3 — Filter characteristics

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

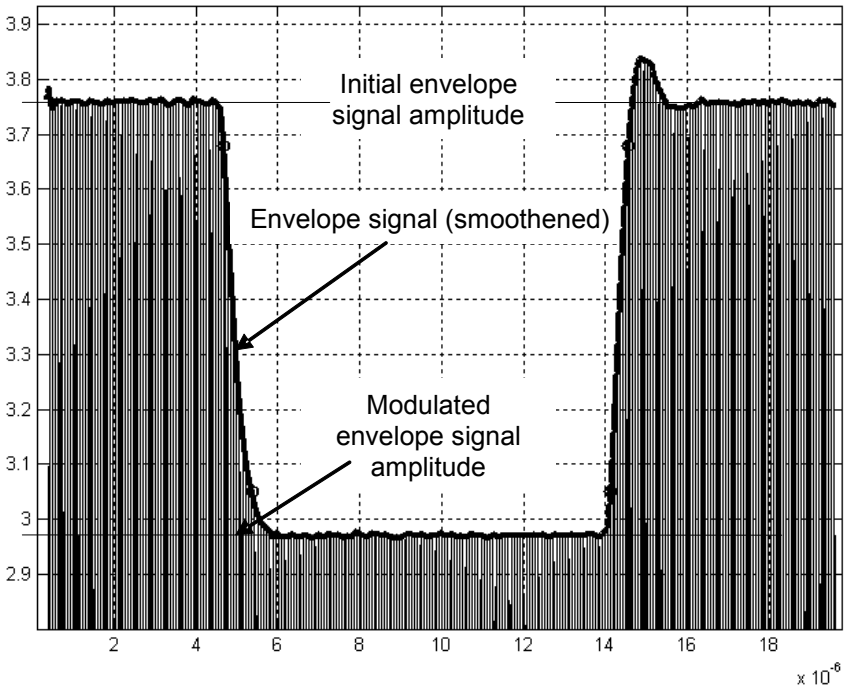


Figure E.4 — Envelope smoothing

E.6 Modulation index determination

The initial and modulated envelope signal amplitudes shall be determined by calculating the histogram of the smoothened 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 smoothened envelope signal shall be further smoothened 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.

E.9.1 structures.h

```
/*************************************************************************/
/*structures.h                                                           */
/*This code contains the structures to save important results            */
/*************************************************************************/
#ifndef 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 3.14159265358979323846264338327795
#endif

float *WR;
float *WI;

doublereal *DWR;
doublereal *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( doublecomplex *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 = (doublereal *)calloc(M,sizeof(doublereal));
   DWI = (doublereal *)calloc(M,sizeof(doublereal));

   /* 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;
```

```
   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]) /= (doublereal)M;
      IM(X[i]) /= (doublereal)M;
   }

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

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_ */
```

E.9.5 hilbert.c

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
/ * * * This program extract the envelope of modulated carrier      * * * /
/ * * * Input:                                                      * * * /
/ * * * File in text format containing a table of two columns       * * * /
/ * * * (time and test PCD output voltage vd)                       * * * /
/ * * *                                                             * * * /
/ * * * Data format of input-file:                                  * * * /
/ * * *                                                             * * * /
/ * * * One data-point per line,                                    * * * /
/ * * *                                                             * * * /
/ * * * {time[seconds], sense-coil-voltage[volts])                  * * * /
/ * * *                                                             * * * /
/ * * * Data-points shall be equidistant time                       * * * /
/ * * * Minimum sampling rate: 100 MSamples/second                  * * * /
/ * * * example for spreadsheet file (start in next line):          * * * /
/ * * * (time) , (voltage )                                         * * * /
/ * * * 3.00000e-06,1.00                                            * * * /
/ * * * 3.00200e-06,1.01                                            * * * /
/ * * *                                                             * * * /
/ * * * Run:                                                        * * * /
/ * * * hilbert Filename.txt                                        * * * /
/ * * * or                                                          * * * /
/ * * * hilbert (default file name input.txt)                       * * * /
/ * * *                                                             * * * /

/**************************************************************************/
/*hilbert.c                                                              */
/*Main program                                                           */
/**************************************************************************/

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

#define MAX_POINT 200000
#ifndef M_PI
#define M_PI 3.1415926535897932384626433832795
#endif

int debug=0;
int fftdebug=0;

double *Gvalue;
double *Gtime;
double *Gr;
double *Gi;
double *Gc;
doublecomplex *Gt_ifft;

/*File containing the input data*/
char *InputFileName ="input.txt";

int SampledPoints=0;
int N;
int row;
```

```
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 Debuging*/
   /* 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("ffrpt.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,"%.4e\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);
   sqrti=(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(sqrti);
      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(sqrti);
   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
#ifndef M_PI
#define M_PI 3.14159265358979323846264338327795
#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];

    /* scalling 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 diffr=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)
      diffr=(posval-negval)/((posval+negval)/2);
   else
      diffr=(negval-posval)/((posval+negval)/2);

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

   // L=n*p 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-timestrf1;
   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-leny+1)?0:i-leny+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;

               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]>oscmax)
                   {
                      oscmax=envc2[index_A];
                      osctmax=toutput[index_A];
                   }
                   index_A++;
                }

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

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

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

                if (osctmax-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;
               t3endind=vp90two;
               t4endind=vp60two;


createtime(timeres,0,0,0,0,0,0,0,t1,t1startind,t1start,t1endind,t2,t2startind,t2start
,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++;

                  oscmax=envc2[index_A2];
                  while (toutput[index_A2]<tim3)
                  {
                     if (oscmax<envc2[index_A2])
                     {
```

```
                        oscmax=envc2[index_A2];
                        osctmax=toutput[index_A2];
                     }
                     index_A2++;
                  }

                  if (oscmax-oscmin>(0.09*(Hmax-oscmin)))
                     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-tphione;         // 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=tphione;
             t1startind=vphione;
             t5startind=vpmidone;
             t1endind=vplotwo;
             t6endind=vphitwo;


createtime(timeres,0,0,0,0,0,0,0,t1,t1startind,t1start,t1endind,0,0,0,0,0,0,0,0,t5,t5
startind,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)
               {
                  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
         {
            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++])
             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 milivolts \n", ovsb1);
      else
         fprintf(stdout,"hf = 0 milivolts \n");
      if (ovsb2>0)
         fprintf(stdout,"hr = %f milivolts \n", ovsb2);
      else
         fprintf(stdout,"hr = 0 milivolts \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;
}
```

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 14443-3:2010. 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 14443-3:2010. 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 14443-3:2010 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 14443-3:2010, 6.2.1.1);

— Frame formats (see ISO/IEC 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010 and should be subjected to both the part 3 and part 4 tests for the applicable Type.

A PICC compliant with ISO/IEC 14443-3:2010 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 14443-3:2010.

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 14443-3:2010	5
G.3.3	Testing of the PICC Type A state transitions	ISO/IEC 14443-3:2010	6.3, 6.4, 6.5
G.3.4	Handling of Type A anticollision	ISO/IEC 14443-3:2010	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 14443-3:2010	5
G.4.3	PICC Reception	ISO/IEC 14443-3:2010	7.1
G.4.4	Testing of the PICC Type B state transitions	ISO/IEC 14443-3:2010	7.4 – 7.12
G.4.5	Handling of Type B anticollision	ISO/IEC 14443-3:2010	7.4 – 7.12
G.4.6	Handling of ATTRIB	ISO/IEC 14443-3:2010	7.10
G.4.7	Handling of Maximum Frame Size	ISO/IEC 14443-3:2010	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 14443-3:2010 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010 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:

a) Rerun the test a second time, without checking the TTS.

b) 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	⟶	
		⟵	Mute
	WUPA	⟶	
		⟵	ATQA
READY*(I), I < CascadeLevels	SELECT (I)	⟶	
		⟵	SAK (cascade)
READY*(I), I = CascadeLevels	SELECT (I)	⟶	
		⟵	SAK (complete)
ACTIVE*	RATS(0,0)	⟶	
		⟵	ATS
[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 14443-3:2010, 6.3.2.

G.3.3.3.2 Procedure

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

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) 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.8 — Transitions from IDLE state

Transition	PICC-test-apparatus		PICC	FDT	TTS
REQA	REQA	⟶			READY(1)
		⟵	ATQA	1172/*fc*	
WUPA	WUPA	⟶			READY(1)
		⟵	ATQA	1236/*fc*	
HLTA	HLTA	⟶			IDLE
		⟵	Mute		

Transition	PICC-test-apparatus		PICC	FDT	TTS
AC [c]	('93' NVB $UIDTX_1$[[1..n_1]]) [a]	⟶			IDLE
		⟵	Mute		
nAC [d]	('93' NVB ~$UIDTX_1$[[1.. n_1]]) [a]	⟶			IDLE
		⟵	Mute		
SELECT [c]	SELECT(1)	⟶			IDLE
		⟵	Mute		
nSELECT [d]	('93 70' ~$UIDTX_1$[[1..32]] BCC CRC_A)	⟶			IDLE
		⟵	Mute		
RATS	RATS(0,0)	⟶			IDLE
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			IDLE
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			IDLE
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			IDLE
		⟵	Mute		
Error condition	('26') [b]	⟶			IDLE
		⟵	Mute		

[a] Let $1 \le n_1 \le 32$.

[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 14443-3:2010, 6.3.3.

G.3.3.4.2 Procedure

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

a) Put the PICC into READY(1) 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.10 — Transitions from READY(1) state

Transition	PICC-test-apparatus		PICC	FDT	TTS
REQA	REQA	→			IDLE
		←	Mute		
WUPA	WUPA	→			IDLE
		←	Mute		
HLTA	HLTA	→			IDLE
		←	Mute		
AC [g] (split after (0)b)	('93' NVB $UIDTX_1$[[1..n_1]]) [a]	→			READY(1)
		←	if n_1 = 32 then (BCC) else ($UIDTX_1$[[n_1+1..32]] BCC) [a]	1172/*fc*	
AC [g] (split after (1)b)	('93' NVB $UIDTX_1$[[1..n_2]]) [b]	→			READY(1)
		←	if n_2 = 32 then (BCC) else ($UIDTX_1$[[n_2+1..32]] BCC) [b]	1236/*fc*	
nAC [g] (wrong UID)	('93' NVB ~$UIDTX_1$[[1..n_3]]) [f]	→			IDLE
		←	Mute		
SELECT [g]	SELECT(1)	→			TTS [e]
		←	SAK [d]	FDT [c]	
nSELECT [g] (wrong UID)	('93 70' ~$UIDTX_1$ BCC CRC_A)	→			IDLE
		←	Mute		
Error condition	('93 70' $UIDTX_1$ BCC ~CRC_A)	→			IDLE
		←	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	→			IDLE
		←	Mute		
DESELECT	S(DESELECT)	→			IDLE
		←	Mute		

Transition	PICC-test-apparatus		PICC	FDT	TTS
RATS	RATS(0,0)	⟶			IDLE
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			IDLE
		⟵	Mute		

[a] Let $1 \le n_1 \le 32$, $UIDTX_1[[n_1]] = 0$. If such a number does not exist, the test may be skipped.

[b] Let $1 \le n_2 \le 32$, $UIDTX_1[[n_2]] = 1$. 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 14443-3:2010, 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 $1 \le n_3 \le 32$.

[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 14443-3:2010, 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	⟶			IDLE
		⟵	Mute		
WUPA	WUPA	⟶			IDLE
		⟵	Mute		
HLTA	HLTA	⟶			IDLE
		⟵	Mute		
AC [g] (split after (0)b)	('95' NVB $UIDTX_2[[1..n_1]]$) [a]	⟶			READY(2)
		⟵	if $n_1 = 32$ then (BCC) else ($UIDTX_2[[n_1+1..32]]$ BCC) [a]	1172/*fc*	
AC [g] (split after (1)b)	('95' NVB $UIDTX_2[[1..n_2]]$) [b]	⟶			READY(2)
		⟵	if $n_2 = 32$ then (BCC) else ($UIDTX_2[[n_2+1..32]]$ BCC) [b]	1236/*fc*	
nAC [g] (wrong UID)	('95' NVB ~$UIDTX_2[[1..n_3]]$) [f]	⟶			IDLE
		⟵	Mute		
SELECT [g]	SELECT(2)	⟶			TTS [e]
		⟵	SAK [d]	FDT [c]	
nSELECT [g] (wrong UID)	('95 70' ~$UIDTX_2$ BCC CRC_A)	⟶			IDLE
		⟵	Mute		
Error condition	('95 70' $UIDTX_2$ BCC ~CRC_A)	⟶			IDLE
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			IDLE
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			IDLE
		⟵	Mute		
RATS	RATS(0,0)	⟶			IDLE
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			IDLE
		⟵	Mute		

[a] Let $1 \le n_1 \le 32$, $UIDTX_2[[n_1]] = 0$. If such a number does not exist, the test may be skipped.

[b] Let $1 \le n_2 \le 32$, $UIDTX_2[[n_2]] = 1$. 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 14443-3:2010, 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 $1 \le n_3 \le 32$.

[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 14443-3:2010, 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	⟶			IDLE
		⟵	Mute		
WUPA	WUPA	⟶			IDLE
		⟵	Mute		
HLTA	HLTA	⟶			IDLE
		⟵	Mute		
AC [e] (split after (0)b)	('97' NVB $UIDTX_3[[1..n_1]]$) [a]	⟶			READY(3)
		⟵	if n_1 = 32 then (BCC) else ($UIDTX_3[[n_1+1..32]]$ BCC) [a]	1172/*fc*	
AC [e] (split after (1)b)	('97' NVB $UIDTX_3[[1..n_2]]$) [b]	⟶			READY(3)
		⟵	if n_2 = 32 then (BCC) else ($UIDTX_3[[n_2+1..32]]$ BCC) [b]	1236/*fc*	
nAC [e] (wrong UID)	('97' NVB ~$UIDTX_3[[1..n_3]]$) [d]	⟶			IDLE
		⟵	Mute		
SELECT [e]	SELECT(3)	⟶			ACTIVE
		⟵	SAK (complete)	FDT [c]	
nSELECT [e] (wrong UID)	('97 70' ~$UIDTX_3$ BCC CRC_A)	⟶			IDLE
		⟵	Mute		
Error condition	('97 70' $UIDTX_3$ BCC ~CRC_A)	⟶			IDLE
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			IDLE
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			IDLE
		⟵	Mute		
RATS	RATS(0,0)	⟶			IDLE
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			IDLE
		⟵	Mute		

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

[b] Let $1 \le n_2 \le 32$, UIDTX3[[n_2]] = 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 14443-3:2010, 6.2.1.1).

[d] Let $1 \le n_3 \le 32$.

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

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 14443-3:2010, 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 $UIDTX_1[[1..n_1]]$) [a]	⟶			IDLE
		⟵	Mute [b]		
nAC	('93' NVB ~$UIDTX_1[[1..n_1]]$) [a]	⟶			IDLE
		⟵	Mute [b]		
HLTA	HLTA	⟶			HALT
		⟵	Mute		

Transition	PICC-test-apparatus		PICC	FDT	TTS
SELECT	SELECT(1)	⟶			IDLE
		⟵	Mute [b]		
nSELECT	('93 70' ~UIDTX$_1$ BCC CRC_A)	⟶			IDLE
		⟵	Mute [b]		
RATS	RATS(0,0)	⟶			PROTOCOL
		⟵	ATS	< 65536/*fc*	
Error condition	('E0 00' ~CRC_A)	⟶			IDLE
		⟵	Mute [b]		
ISO/IEC 14443-4 command	I(0)$_0$(TEST_COMMAND1(1))	⟶			IDLE
		⟵	Mute [b]		
DESELECT	S(DESELECT)	⟶			IDLE
		⟵	Mute [b]		
PPS	PPS(0,0,0)	⟶			IDLE
		⟵	Mute [b]		

[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 14443-3:2010, 6.3.5.

G.3.3.8.2 Procedure

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

a) Put the PICC into HALT 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.18 — Transitions from HALT state

Transition	PICC-test-apparatus		PICC	FDT	TTS
REQA	REQA	→			HALT
		←	Mute		
WUPA	WUPA	→			READY*(1)
		←	ATQA	1236/*fc*	
HLTA	HLTA	→			HALT
		←	Mute		
AC	('93' NVB $UIDTX_1[[1..n_1]]$) [a]	→			HALT
		←	Mute		
nAC	('93' NVB ~$UIDTX_1[[1..n_1]]$) [a]	→			HALT
		←	Mute		
SELECT	SELECT(1)	→			HALT
		←	Mute		
nSELECT	('93 70' ~$UIDTX_1$ BCC CRC_A)	→			HALT
		←	Mute		
RATS	RATS(0,0)	→			HALT
		←	Mute		
Error condition	('52') in the standard frame	→			HALT
		←	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	→			HALT
		←	Mute		
DESELECT	S(DESELECT)	→			HALT
		←	Mute		
PPS	PPS(0,0,0)	→			HALT
		←	Mute		

[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 14443-3:2010, 6.3.6.

G.3.3.9.2 Procedure

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

a) Put the PICC into READY*(1) 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 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	→			HALT
		←	Mute		
WUPA	WUPA	→			HALT
		←	Mute		
HLTA	HLTA	→			HALT
		←	Mute		
AC (split after (0)b)	('93' NVB $UIDTX_1[[1..n_1]]$) [a]	→			READY*(1)
		←	if n_1 = 32 then (BCC) else ($UIDTX_1[[n_1+1..32]]$ BCC) [a]	1172/*fc*	
AC (split after (1)b)	('93' NVB $UIDTX_1[[1..n_2]]$) [b]	→			READY*(1)
		←	if n_2 = 32 then (BCC) else ($UIDTX_1[[n_2+1..32]]$ BCC) [b]	1236/*fc*	
nAC (wrong UID)	('93' NVB ~$UIDTX_1[[1..n_3]]$) [f]	→			HALT
		←	Mute		

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 $1 \le n_1 \le 32$, UIDTX$_1$[[n_1]] = 0. If such a number does not exist, the test may be skipped.

b Let $1 \le n_2 \le 32$, UIDTX$_1$[[n_2]] = 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 14443-3:2010, 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 $1 \le n_3 \le 32$.

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 14443-3:2010, 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:

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.22 — Transitions from READY*(2) state

Transition	PICC-test-apparatus		PICC	FDT	TTS
REQA	REQA	⟶			HALT
		⟵	Mute		
WUPA	WUPA	⟶			HALT
		⟵	Mute		
HLTA	HLTA	⟶			HALT
		⟵	Mute		
AC (split after (0)b)	('95' NVB $UIDTX_2$[[1..n_1]]) [a]	⟶		1172/*fc*	READY*(2)
		⟵	if n_1 = 32 then (BCC) else ($UIDTX_2$[[n_1+1..32]] BCC) [a]		
AC (split after (1)b)	('95' NVB $UIDTX_2$[[1..n_2]]) [b]	⟶		1236/*fc*	READY*(2)
		⟵	if n_2 = 32 then (BCC) else ($UIDTX_2$[[n_2+1..32]] BCC) [b]		
nAC (wrong UID)	('95' NVB ~$UIDTX_2$[[1..n_3]]) [f]	⟶			HALT
		⟵	Mute		
SELECT	SELECT(2)	⟶		FDT [c]	TTS [e]
		⟵	SAK [d]		
nSELECT (wrong UID)	('95 70' ~$UIDTX_2$ BCC CRC_A)	⟶			HALT
		⟵	Mute		
Error condition	('95 70' $UIDTX_2$ BCC ~CRC_A)	⟶			HALT
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			HALT
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			HALT
		⟵	Mute		

Transition	PICC-test-apparatus		PICC	FDT	TTS
RATS	RATS(0,0)	⟶			HALT
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			HALT
		⟵	Mute		

[a] Let $1 \le n_1 \le 32$, $UIDTX_2[[n_1]] = 0$. If such a number does not exist, the test may be skipped.

[b] Let $1 \le n_2 \le 32$, $UIDTX_2[[n_2]] = 1$. 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 14443-3:2010, 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 $1 \le n_3 \le 32$.

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 14443-3:2010, 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	⟶			HALT
		⟵	Mute		
WUPA	WUPA	⟶			HALT
		⟵	Mute		
HLTA	HLTA	⟶			HALT
		⟵	Mute		
AC (split after (0)b)	('97' NVB $UIDTX_3[[1..n_1]]$) [a]	⟶			READY*(3)
		⟵	if $n_1 = 32$ then (BCC) else ($UIDTX_3[[n_1+1..32]]$ BCC) [a]	1172/*fc*	
AC (split after (1)b)	('97' NVB $UIDTX_3[[1..n_2]]$) [b]	⟶			READY*(3)
		⟵	if $n_2 = 32$ then (BCC) else ($UIDTX_3[[n_2+1..32]]$ BCC) [b]	1236/*fc*	
nAC (wrong UID)	('97' NVB ~$UIDTX_3[[1..n_3]]$) [d]	⟶			HALT
		⟵	Mute		
SELECT	SELECT(3)	⟶			ACTIVE*
		⟵	SAK (complete)	FDT [c]	
nSELECT (wrong UID)	('97 70' ~$UIDTX_3$ BCC CRC_A)	⟶			HALT
		⟵	Mute		
Error condition	('97 70' $UIDTX_3$ BCC ~CRC_A)	⟶			HALT
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			HALT
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			HALT
		⟵	Mute		
RATS	RATS(0,0)	⟶			HALT
		⟵	Mute		
PPS	PPS(0,0,0)	⟶			HALT
		⟵	Mute		

[a] Let $1 \le n_1 \le 32$, $UIDTX_3[[n_1]] = 0$. If such a number does not exist, the test may be skipped.

[b] Let $1 \le n_2 \le 32$, $UIDTX_3[[n_2]] = 1$. 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 14443-3:2010, 6.2.1.1).

[d] Let $1 \le n_3 \le 32$.

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 14443-3:2010, 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 $UIDTX_1[[1..n_1]]$) [a]	⟶			HALT
		⟵	Mute [b]		
nAC	('93' NVB ~$UIDTX_1[[1..n_1]]$) [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)	⟶			HALT
		⟵	Mute [b]		
RATS	RATS(0,0)	⟶		< 65536/*fc*	PROTOCOL
		⟵	ATS		
Error condition	('E0 00' ~CRC_A)	⟶			HALT
		⟵	Mute [b]		
ISO/IEC 14443-4 command	I(0)$_0$(TEST_COMMAND1(1))	⟶			HALT
		⟵	Mute [b]		
DESELECT	S(DESELECT)	⟶			HALT
		⟵	Mute [b]		
PPS	PPS(0,0,0)	⟶			HALT
		⟵	Mute [b]		

[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:

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) 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	⟶			PROTOCOL
		⟵	Mute		
WUPA	WUPA	⟶			PROTOCOL
		⟵	Mute		
AC	('93' NVB $UIDTX_1[[1..n_1]]$) [a]	⟶			PROTOCOL
		⟵	Mute		
nAC	('93' NVB ~$UIDTX_1[[1..n_1]]$) [a]	⟶			PROTOCOL
		⟵	Mute		
HLTA	HLTA	⟶			PROTOCOL
		⟵	Mute		
SELECT	SELECT(1)	⟶			PROTOCOL
		⟵	Mute		
nSELECT	('93 70' ~$UIDTX_1$ BCC CRC_A)	⟶			PROTOCOL
		⟵	Mute		
RATS	RATS(0,0)	⟶			PROTOCOL
		⟵	Mute		
Error condition	S(DESELECT, ~CRC_A)	⟶			PROTOCOL
		⟵	Mute		
DESELECT	S(DESELECT)	⟶			HALT
		⟵	S(DESELECT)	65536/*fc* as specified in ISO/IEC 14443-4: 2008, 8.1	
PPS	PPS(0,0,0)	⟶			PROTOCOL
		⟵	Mute, or PPS response [b]		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			PROTOCOL
		⟵	$I(0)_0$(TEST_RESPONSE1(1))	< FWT	

[a] Let $1 \leq n_1 \leq 32$.

[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 14443-3:2010, 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 ≤ 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 ≤ 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 14443-3:2010 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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* ≤ TR0 ≤ 256/*fs*;

— The TR1 timing shall be in the range 80/*fs* ≤ TR1 ≤ 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 14443-3:2010, 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 14443-3:2010 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	$I(0)_0$(TEST_COMMAND1(1))	⟶	
		⟵	$I(0)_0$(TEST_RESPONSE1(1))
HALT	REQB(1)	⟶	
		⟵	Mute
	WUPB(1)	⟶	
		⟵	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 14443-3:2010, 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)	⟶		READY-DECLARED
		⟵	ATQB	
WUPB	WUPB(1)	⟶		READY-DECLARED
		⟵	ATQB	
REQB (wrong CRC)	('05 00 00' ~CRC_B)	⟶		IDLE
		⟵	Mute	
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	⟶		IDLE
		⟵	Mute	
HLTB [b]	HLTB	⟶		IDLE
		⟵	Mute	
ATTRIB [b]	ATTRIB(0,0)	⟶		IDLE
		⟵	Mute	

Transition	PICC-test-apparatus		PICC	TTS
Slot-MARKER	SLOTMARKER(n) [a]	⟶		IDLE
		⟵	Mute	
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶		IDLE
		⟵	Mute	
DESELECT	S(DESELECT)	⟶		IDLE
		⟵	Mute	
REQB (Unmatched AFI)	REQB(1,nAFI)	⟶		IDLE
		⟵	Mute	
WUPB (Unmatched AFI)	WUPB(1,nAFI)	⟶		IDLE
		⟵	Mute	
HLTB (Unmatched PUPI)	HLTB(~PUPI)	⟶		IDLE
		⟵	Mute	
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	⟶		IDLE
		⟵	Mute	
REQB [d]	REQB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	
WUPB [d]	WUPB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	

[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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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)	⟶		READY-DECLARED
		⟵	ATQB	
WUPB	WUPB(1)	⟶		READY-DECLARED
		⟵	ATQB	
REQB (wrong CRC)	('05 00 00' ~CRC_B)	⟶		READY-REQUESTED
		⟵	Mute	
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	⟶		READY-REQUESTED
		⟵	Mute	
HLTB [b]	HLTB	⟶		READY-REQUESTED
		⟵	Mute	
ATTRIB [b]	ATTRIB(0,0)	⟶		READY-REQUESTED
		⟵	Mute	
Slot-MARKER	SLOTMARKER(n) [a]	⟶		READY-DECLARED
		⟵	ATQB or Mute	
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶		READY-REQUESTED
		⟵	Mute	
DESELECT	S(DESELECT)	⟶		READY-REQUESTED
		⟵	Mute	
REQB	REQB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	
WUPB	WUPB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	
REQB (Unmatched AFI)	REQB(1,nAFI)	⟶		IDLE [d]
		⟵	Mute	

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 14443-3:2010, 7.4.6.

G.4.4.4.2 Procedure

Perform the following steps for every row of Table G.43:

a) Put the PICC into READY-DECLARED 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.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)	⟶		READY-DECLARED
		⟵	ATQB	
REQB (wrong CRC)	('05 00 00' ~CRC_B)	⟶		READY-DECLARED
		⟵	Mute	
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	⟶		READY-DECLARED
		⟵	Mute	
HLTB	HLTB	⟶		HALT
		⟵	('00' CRC_B)	
ATTRIB	ATTRIB(0,0)	⟶		PROTOCOL
		⟵	ATA(0)	
Slot-MARKER	SLOTMARKER(n) [a]	⟶		READY-DECLARED
		⟵	Mute	
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶		READY-DECLARED
		⟵	Mute	
DESELECT	S(DESELECT)	⟶		READY-DECLARED
		⟵	Mute	
REQB [b]	REQB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	
WUPB [b]	WUPB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	
HLTB (Unmatched PUPI)	HLTB(~PUPI)	⟶		READY-DECLARED
		⟵	Mute	
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	⟶		READY-DECLARED
		⟵	Mute	
REQB (Unmatched AFI)	REQB(1,nAFI)	⟶		IDLE [d]
		⟵	Mute	
WUPB (Unmatched AFI)	WUPB(1,nAFI)	⟶		IDLE [d]
		⟵	Mute	

[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 14443-3:2010, 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 14443-3:2010, 7.4.8.

G.4.4.5.2 Procedure

Perform the following steps for every row of Table G.45:

a) Put the PICC into HALT 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.45 — Transitions from HALT state

Transition	PICC-test-apparatus		PICC	TTS
REQB	REQB(1)	⟶		HALT
		⟵	Mute	
WUPB	WUPB(1)	⟶		READY-DECLARED
		⟵	ATQB	
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	⟶		HALT
		⟵	Mute	
HLTB	HLTB	⟶		HALT
		⟵	Mute	
ATTRIB	ATTRIB(0,0)	⟶		HALT
		⟵	Mute	
Slot-MARKER	SLOTMARKER(n) [a]	⟶		HALT
		⟵	Mute	
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶		HALT
		⟵	Mute	
DESELECT	S(DESELECT)	⟶		HALT
		⟵	Mute	

Transition	PICC-test-apparatus		PICC	TTS
WUPB (Unmatched AFI)	WUPB(1,nAFI)	⟶		IDLE
		⟵	Mute	
REQB (Unmatched AFI)	REQB(1,nAFI)	⟶		HALT
		⟵	Mute	
HLTB (Unmatched PUPI)	HLTB(~PUPI)	⟶		HALT
		⟵	Mute	
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	⟶		HALT
		⟵	Mute	
WUPB [b]	WUPB(16) [c]	⟶		READY-REQUESTED
		⟵	Mute	

[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 14443-3:2010, 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)	⟶			PROTOCOL
		⟵	Mute		
WUPB	WUPB(1)	⟶			PROTOCOL
		⟵	Mute		
REQB (wrong CRC)	('05 00 00' ~CRC_B)	⟶			PROTOCOL
		⟵	Mute		
WUPB (wrong CRC)	('05 00 08' ~CRC_B)	⟶			PROTOCOL
		⟵	Mute		
HLTB	HLTB	⟶			PROTOCOL
		⟵	Mute		
ATTRIB	ATTRIB(0,0)	⟶			PROTOCOL
		⟵	Mute		
Slot-MARKER	SLOTMARKER(n) [a]	⟶			PROTOCOL
		⟵	Mute		
ISO/IEC 14443-4 command	$I(0)_0$(TEST_COMMAND1(1))	⟶			PROTOCOL
		⟵	$I(0)_0$(TEST_RESPONSE1(1))	< FWT	
DESELECT	S(DESELECT)	⟶			HALT
		⟵	S(DESELECT)		
WUPB (Unmatched AFI)	WUPB(1,nAFI)	⟶			PROTOCOL
		⟵	Mute		
REQB (Unmatched AFI)	REQB(1,nAFI)	⟶			PROTOCOL
		⟵	Mute		
HLTB (Unmatched PUPI)	HLTB(~PUPI)	⟶			PROTOCOL
		⟵	Mute		
ATTRIB (Unmatched PUPI)	ATTRIB(0, 0, ~PUPI)	⟶			PROTOCOL
		⟵	Mute		

[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 14443-3:2010, 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 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 14443-3:2010, 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 14443-3:2010, 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 $\leq$ 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	$R(NAK, \sim CRC)_0$	⟶	
		⟵	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	$R(NAK)_0$	⟶	
		⟵	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	$R(NAK, \sim CRC)_0$	⟶	
		⟵	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	$R(NAK, \sim CRC)_0$	⟶	
		⟵	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	$R(NAK, \sim CRC)_0$	⟶	
		⟵	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 14443-3:2010, 6.2.1.1	Last bit (1)b -> 1236/*fc* Last bit (0)b -> 1172/*fc*	Last bit (1)b -> 1236/*fc* + 0,4 µs 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 14443-3:2010, 6.2.1.1 and ISO/IEC 14443-4:2008, 8.1	Last bit (1)b -> 1236/*fc* Last bit (0)b -> 1172/*fc*	(256/*fs*) × 2^{FWI} (~302,06 µs × 2^{FWI})	FWI = Max FDT =

NOTE All timing values are calculated for carrier frequency *fc* = 13,56 MHz and bit rate = *fc*/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 14443-3:2010, 7.1.4	10 etu (~94,40 µs)	11 etu (~103,83 µs)	
2	SOF high	ISO/IEC 14443-3:2010, 7.1.4	2 etu (~18,88 µs)	3 etu (~28,32 µs)	
3	EOF low	ISO/IEC 14443-3:2010, 7.1.5	10 etu (~94,40 µs)	11 etu (~103,83 µs)	
4	Bit boundaries	ISO/IEC 14443-3:2010, 7.1.1	(n – 1/8) etu	(n + 1/8) etu	
5	EGT PICC to PCD	ISO/IEC 14443-3:2010, 7.1.2	0 µs	19 µs	
6	TR0 for ATQB	ISO/IEC 14443-3:2010, 7.1.6	64/*fs* (~75,52 µs)	256/*fs* (~302,06 µs)	
7	TR1 for ATQB	ISO/IEC 14443-3:2010, 7.1.6	80/*fs* (~94,40 µs)	200/*fs* (~235,99 µs)	
8	TR0 Not ATQB	ISO/IEC 14443-3:2010, 7.1.6 ISO/IEC 14443-3:2010, 7.10.3	Value as defined in ISO/IEC 14443-3:2010, Table 30	(256/*fs*) × 2^{FWI} (~302,06 µs × 2^{FWI})	FWI = Max TR0 =
9	TR1 Not ATQB	ISO/IEC 14443-3:2010, 7.1.6 ISO/IEC 14443-3:2010, 7.10.3	Value as defined in ISO/IEC 14443-3:2010, Table 31	200/*fs* (~235,99 µs)	
10	TR2	ISO/IEC 14443-3:2010, 7.9.4.4	Value as defined in ISO/IEC 14443-3:2010, Table 27	No maximum	
11	Delay from the end of EOF and Subcarrier off	ISO/IEC 14443-3:2010, 7.1.7	0	2 etu	
12	Deactivation frame waiting time	ISO/IEC 14443-4:2008, 8.1	64/*fs* + 80/*fs* (~169,92 µs)	65536/*fc* (~4,8 ms)	

NOTE All timing values are calculated for carrier frequency *fc* = 13,56 MHz and bit rate = *fc*/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	
		Scenario G.51	
		Scenario G.52	
		Scenario G.53	
		Scenario G.54	
G.5.3	Handling of PICC error detection	Scenario G.55	
		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 14443-3:2010 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.

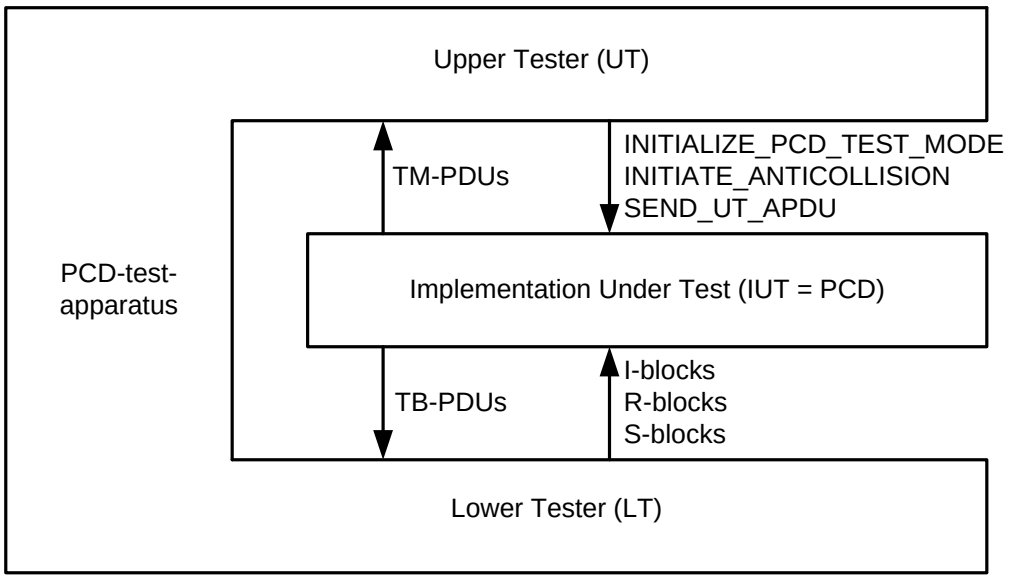


Figure H.1 — Conceptual tester architecture

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 14443-3:2010. 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 14443-3:2010 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 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010	6.2.1.2
H.2.2	Request Guard Time	ISO/IEC 14443-3:2010	6.2.2
H.2.3	Handling of bit collision during ATQA	ISO/IEC 14443-3:2010	6.5.2
H.2.4	Handling of anticollision loop	ISO/IEC 14443-3:2010	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 14443-3:2010	7.1
H.3.2	Frame size selection mechanism	ISO/IEC 14443-3:2010	7.9
H.3.3	Handling of the CID during activation by the PCD	ISO/IEC 14443-3:2010	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 14443-3:2010	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 14443-3:2010.

e) Measure the time between the last modulation transmitted by the LT and the first pause transmitted by the PCD (see ISO/IEC 14443-3:2010, 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 14443-3:2010, 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 14443-3:2010, 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	⟶		
		⟵	ATQA (single size UID)	1
ANTICOLLISION Level 1	ANTICOLLISION command Level 1 ('93 20')	⟶		
		⟵	UID CL1 (uid0 uid1 uid2 uid3 BCC)	2
SELECT	SELECT command ('93 70' uid0 uid1 uid2 uid3 BCC CRC_A)	⟶		
		⟵	SAK(complete)	3

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	→		
		←	ATQA (double 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 (uid3 uid4 uid5 uid6 BCC)	4
SELECT	SELECT command ('95 70' uid3 uid4 uid5 uid6 BCC CRC_A)	→		
		←	SAK(complete)	5

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 $UIDTX_1$[[1..k]], where $UIDTX_1$[[1..k-1]] is either empty (i.e. k = 1) or the value already known by the PCD and $UIDTX_1$[[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' $UIDTX_1$[[1..32]]. In this case the LT answers with BCC.

NOTE This optional ANTICOLLISON command does not change the Stage number.

j) The PCD shall send SELECT command '93 70' $UIDTX_1$[[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 $UID_{PARTIAL}$) $1 \leq k \leq 31$	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
	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
Mute		⟵	Mute
	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	Mute
start deactivation	DESELECT	⟶	
	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	erroneous ATS frame [a]
erroneous ATS frame	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	erroneous ATS frame [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.

d) The PCD shall return the result code, in accordance with Table H.1, of its action to the UT.

e) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

f) The PCD is expected to send any I-block (including empty) to the LT, possibly after PICC presence check sequences.

Scenario H.6 — Handling of RATS and ATS, Procedure 2

Test	PCD		LT
correct ATS	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	ATS
continue operation	Any I-block (including empty)	⟶	

H.2.5.2.3 Procedure 3

In case the PCD does not use the optional retransmission of RATS according to ISO/IEC 14443-4:2008, 5.6.1.1, skip this procedure.

In case the PCD uses the optional retransmission of RATS according to ISO/IEC 14443-4:2008, 5.6.1.1 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) When the PCD has transmitted the RATS, the LT does not respond to the RATS (Mute).

d) When the PCD has retransmitted the RATS, the LT answers with a valid ATS.

e) The PCD is expected to send any I-block (including empty) to the LT, possibly after PICC presence check sequences, or a PPS request.

f) Repeat the procedure with an erroneous ATS frame (use a wrong CRC_A) instead of Mute.

Scenario H.7 — Handling of RATS and ATS, Procedure 3

Test	PCD		LT
Mute or erroneous ATS	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	Mute / erroneous ATS frame [a]
PCD retransmits RATS	RATS command frame (e.g.'E0 01' CRC_A)	⟶	
		⟵	ATS
Continue operation	Any I-block (including empty) or PPS request	⟶	

[a] Determined in step f).

H.2.5.3 Test report

Fill the appropriate row in Table H.23 according to Table H.7.

Table H.7 — Result criteria for handling of RATS and ATS

Explanation	Test result
Only when the PCD's behavior matches each procedure expected Scenario	PASS
Any other case	FAIL

H.2.6 Handling of PPS response

The purpose of this test is to determine the handling of the PPS request according to ISO/IEC 14443-4:2008, 5.6.2.1. This test is applicable only for the PCD, which uses Protocol and Parameter Selection mechanism as a part of the PICC activation sequence. In case the PCD does not use the PPS mechanism the test report column shall indicate N/A.

H.2.6.1 Apparatus

See H.1.

H.2.6.2 Preliminary Procedure

Use the following sequence to put the PCD into the state required by this test:

a) Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

b) The UT performs the activation procedure according to H.1.8.1.

c) The LT answers relevant anticollision messages and waits until the PCD sends the RATS.

d) The LT answers with ATS (with valid TA<>'00' indicating, that higher bit rates are supported and therefore PPS is supported by this PICC and thus the PCD may perform the PPS sequence).

H.2.6.2.1 Procedure 1

Use the following sequence immediately after procedure H.2.6.2:

a) The LT waits until the PCD sends a valid PPS request. Ensure, that PPSS start byte, Parameter 0 and Parameter 1 do not include RFU values.

NOTE It is not mandatory to send a PPS request.

b) The LT answers with PPS response.

c) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

d) The PCD shall transmit I-block using the selected parameters, possibly after PICC presence check sequences.

Scenario H.8 — Handling of PPS request and response, Procedure 1

Test	PCD		LT
correct PPS response	PPS request	⟶	
		⟵	PPS response
Set parameters and continue operation	The I-block provided by the UT	⟶	

H.2.6.2.2 Procedure 2

Use the following sequence immediately after procedure H.2.6.2:

a) The LT waits until the PCD sends a valid PPS request.

b) The LT answers with erroneous PPS response (use a wrong CRC_A).

c) The PCD may retransmit a valid PPS request or continue operation (e.g. send an I-block), both using the default bit rate.

d) Repeat the procedure with no response to the PPS request (Mute).

Scenario H.9 — Handling of PPS request and response, Procedure 2

Test	PCD		LT
erroneous PPS response or Mute	PPS request	⟶	
		⟵	Erroneous PPS response or Mute [a]
	Optional PPS request or an I-block	⟶	
[a] Determined in step d).			

H.2.6.3 Test report

Fill the appropriate row in Table H.23 according to Table H.8.

Table H.8 — Result criteria for handling of PPS request and response

Explanation	Test result
Only when the PCD's behavior matches each procedure expected Scenario	PASS
Any other case	FAIL

H.2.7 Frame size selection mechanism

The purpose of this test is to verify the correct handling of transmitted frame size. The transmitted frames shall not be longer than FSCI indication. This test shall be executed for at least FSCI set to 0, 1 and 8.

H.2.7.1 Apparatus

See H.1.

H.2.7.2 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 answers relevant anticollision messages and waits until the PCD sends a valid RATS command frame.

c) The LT answers with a valid ATS. For the purpose of this test, the LT returns format byte T0 equal '70' (see ISO/IEC 14443-4:2008, 5.2.3). In case there is a PPS request the LT will answer it before continuing with the next step:

   Maximum size of a frame accepted by the LT is in accordance with FSCI.

d) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND2) to the PCD where the data length shall be more than the maximum size of a frame accepted by the LT.

e) The PCD shall send the following $I(1)_0$ block with maximum length of in accordance with FSCI.

Scenario H.10 — Frame size selection mechanism

PCD		LT
RATS command frame (e.g. 'E0 01' CRC_A)	⟶	
	⟵	ATS, T0 = '70'
Optional PPS request is possible and would be processed by the LT before the I-block	⟶	
	⟵	PPS response in case of PPS request
$I(1)_0$ (maximum length in accordance with FSCI)	⟶	

H.2.7.3 Test report

Fill the appropriate row in Table H.23 according to Table H.9.

Table H.9 — Result criteria for frame size selection mechanism

Explanation	Test result
Only when the PCD's behavior matches the expected Scenario for all tested FSCI values	PASS
Any other case	FAIL

H.2.8 Handling of Start-up Frame Guard Time

The purpose of this test is to determine the PCD transmission timing according to ISO/IEC 14443-4:2008, 5.2.5.

This test shall be executed for at least SFGI set to 0, 1 and 14.

H.2.8.1 Apparatus

See H.1.

H.2.8.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

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 answers relevant anticollision messages and waits until the PCD sends a valid RATS command frame.

c) The LT answers with a valid ATS. For the purpose of this test the LT returns interface byte TB (1) equal '0E' (see ISO/IEC 14443-4:2008, 5.2.5). In case there is a PPS request the LT will answer it before continuing with the next step.

   Value '0E' = (00001110)b means:

   Minimum value of the frame delay accepted by the LT is $(256 \times 16/fc) \times 2^{14}$ (~4949 ms).

d) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

e) The PCD shall send the following $I(0 \text{ or } 1)_0$ block after a minimum delay of ~4949 ms.

Scenario H.11 — Start-up Frame Guard Time mechanism

PCD		LT
RATS command frame (e.g. 'E0 01' CRC_A)	⟶	
	⟵	ATS, TB (1) = '0E'
Optional PPS request is possible and would be processed by the LT before the I-block	⟶	
	⟵	PPS response in case of PPS request
$I(0 \text{ or } 1)_0$ (Sent after SFGT = $(256 \times 16/fc) \times 2^{14}$ (~4949 ms))	⟶	

H.2.8.3 Test report

Fill the appropriate row in Table H.23 according to Table H.10.

Table H.10 — Result criteria for handling of Start-up Frame Guard Time

Explanation	Test result
Only when the PCD's behavior matches the expected Scenario for all tested SFGI values	PASS
Any other case	FAIL

H.2.9 Handling of the CID during activation by the PCD

The purpose of this test is to determine the handling of the CID according to ISO/IEC 14443-4:2008, 5.6.3. This test shall be executed for at least CID set to 0, 1 and 14 if the CID can be chosen by the UT. Else, only the CID chosen by the PCD shall be used.

H.2.9.1 Apparatus

See H.1.

H.2.9.2 Procedure

Use the sequence a) to c) to put the PCD into the state required by this test:

a) Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

b) The UT performs the activation procedure according to H.1.8.1.

c) The LT answers relevant anticollision messages and waits until the PCD sends the RATS. The LT answers with ATS.

For each test from Scenario H.12, when supported by the PCD, use the sequence d) to h):

d) Put the PCD into the state required by this test.

e) The LT waits until the PCD applies the command as described in PCD column.

f) The LT answers as described in the LT column.

g) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

h) The PCD is expected to send I-block to the LT applying the condition as described in the PCD column.

Scenario H.12 — Handling of the CID

Test	PCD		LT
CID = n not equal to 0 and receive CID is supported	RATS (CID not equal 0)	⟶	
		⟵	ATS (CID supported)
	any valid command using CID	⟶	

Test	PCD		LT
CID = n not equal to 0 and receive CID is not supported	RATS (CID not equal 0)	⟶	
		⟵	ATS (CID not supported)
	any valid command without CID	⟶	
CID = n equal to 0 and receive CID is supported	RATS (CID equal to 0)	⟶	
		⟵	ATS (CID supported)
	any valid command using CID = 0 or without CID	⟶	
CID = n equal to 0 and receive CID is not supported	RATS (CID equal to 0)	⟶	
		⟵	ATS (CID not supported)
	any valid command without CID	⟶	

H.2.9.3 Test report

Fill the appropriate row in Table H.23 according to Table H.11.

Table H.11 — Result criteria for handling of the CID during activation by the PCD

Explanation	Test result
Only when the PCD's behavior matches the expected Scenario	PASS
Any other case	FAIL

H.3 Type B specific test methods

H.3.1 I/O transmission timing

The purpose of this test is to determine the PCD transmission timing according to ISO/IEC 14443-3:2010, 7.1.

H.3.1.1 Apparatus

See H.1.

H.3.1.2 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.

a) The UT performs the activation procedure according to H.1.8.1.

b) Analyze the bit boundaries timing within a character sent by the PCD (see ISO/IEC 14443-3:2010, 7.1.1).

c) Analyze the extra guard time (EGT) between 2 consecutive characters sent by the PCD (see ISO/IEC 14443-3:2010, 7.1.2).

d) Analyze the timing of SOF sent by the PCD (see ISO/IEC 14443-3:2010, 7.1.4).

e) Analyze the timing of EOF sent by the PCD (see ISO/IEC 14443-3:2010, 7.1.5).

f) Analyze the timing before the PCD SOF (see ISO/IEC 14443-3:2010, 7.1.7).

H.3.1.3 Test report

Report the signal recording. Fill Table H.22 with measured values from b) up to f) and the appropriate row of Table H.24.

H.3.2 Frame size selection mechanism

The purpose of this test is to analyze the frame size selection mechanism according to ISO/IEC 14443-3:2010, 7.9.

This test shall be executed for at least Maximum Frame Size Code set to 0, 1 and 8.

H.3.2.1 Apparatus

See H.1.

H.3.2.2 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 REQB/WUPB command frame.

c) The LT answers with ATQB. Assume, that PUPI of the LT is '12 23 34 45' and the LT supports CID. For the purpose of this test the LT returns the second protocol info byte equal '01' (see ISO/IEC 14443-3:2010, 7.9.4), which means that maximum frame size supported by the LT is 16 bytes and the LT is compliant with ISO/IEC 14443-4:2008.

d) The PCD shall send a valid ATTRIB command frame.

e) The LT sends Answer to ATTRIB command.

f) The PCD shall return the result code, in accordance with Table H.1, of its action to the UT.

g) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND2) to the PCD.

h) The PCD shall send an $I(1)_0$ block with maximum length of 16 bytes as shown in the PCD column of Scenario H.13.

Scenario H.13 — Frame size selection mechanism

PCD		LT
REQB/WUPB	⟶	
	⟵	ATQB
ATTRIB command frame	⟶	
	⟵	Answer to ATTRIB command

PCD		LT
$I(1)_0$ (INF = the first chain includes the first block of UT_TEST_COMMAND2), with maximum length 16 bytes.	⟶	

H.3.2.3 Test report

Fill the appropriate row in Table H.24 according to Table H.12.

Table H.12 — Result criteria for frame size selection mechanism

Explanation	Test result
Only when the PCD's behavior matches the expected Scenario for at least Maximum Frame Size Code set to 0, 1 and 8	PASS
Any other case	FAIL

H.3.3 Handling of the CID during activation by the PCD

The purpose of this test is to determine the handling of the CID according to ISO/IEC 14443-3:2010.

This test shall be executed for at least CID set to 0, 1 and 14 if the CID can be chosen by the UT. Else, only the CID chosen by the PCD shall be used.

H.3.3.1 Apparatus

See H.1.

H.3.3.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

H.3.3.2.1 Procedure 1

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 the REQB/WUPB command.

c) The LT sends ATQB with Frame Option bits (b2,b1) equal (00)b. It means: CID and NAD are not supported.

d) The LT waits until the PCD sends the ATTRIB command. The PCD shall send a valid ATTRIB command frame with Param 4 byte equals 0.

e) The LT sends Answer to ATTRIB command with CID value equals 0.

f) The PCD shall return the result code, in accordance with Table H.1, of its action to the UT.

g) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

h) The PCD shall send the following $I(0 \text{ or } 1)_0$ block without NAD and CID.

Scenario H.14 — Handling of the CID, Procedure 1

PCD		LT
REQB/WUPB	⟶	
	⟵	ATQB Frame Option Bits (b2,b1) = (00)b
ATTRIB command frame: (e.g.'1D 12 23 34 45 00 05 01 00' CRC_B)	⟶	
	⟵	Answer to ATTRIB command
$I(0 \text{ or } 1)_0$ without NAD and CID	⟶	

H.3.3.2.2 Procedure 2

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 the REQB/WUPB command.

c) The LT sends ATQB with Frame Option bits (b2,b1) equal (01)b. It means: CID is supported and NAD is not supported.

d) The LT waits until the PCD sends the ATTRIB command. The PCD shall send a valid ATTRIB command frame, using Param 4 byte equals '0X' (CID = X in the range from 0 to 14).

e) The LT sends Answer to ATTRIB command with CID value assigned by the PCD on step d) in Param 4 byte.

f) The PCD shall return the result code, in accordance with Table H.1, of its action to the UT.

g) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

h) The PCD shall send the following $I(0 \text{ or } 1)_0$ block using CID value assigned by the PCD on step d) or, optionally, using no CID if CID = 0. The PCD shall not use NAD in this $I(0 \text{ or } 1)_0$ block.

Scenario H.15 — Handling of the CID, Procedure 2

PCD		LT
REQB/WUPB	⟶	
	⟵	ATQB Frame Option Bits (b2,b1) = (01)b
ATTRIB command frame with, (e.g., Param 4 = '0X'): (e.g.'1D 12 23 34 45 00 05 01 0X' CRC_B), X in the range from 0 to 14.	⟶	
	⟵	Answer to ATTRIB command
$I(0 \text{ or } 1)_0$ with CID value equals '0X' and without NAD	⟶	

H.3.3.3 Test report

Fill the appropriate row in Table H.24 according to Table H.13.

Table H.13 — Report criteria for handling of the CID during activation by the PCD

Explanation	Test result
Only when the PCD's behavior matches each procedure expected Scenario	PASS
Any other case	FAIL

H.4 Test method for logical operations of the PCD

All test methods described in this clause, except H.4.1, shall be applied twice, once for Type A signal interface and once for Type B signal interface.

H.4.1 Handling of the polling loop

The purpose of this test is to determine the behavior of the PCD during polling. The conditions defined in ISO/IEC 14443-3:2010, 5.1 shall apply.

H.4.1.1 Apparatus

See H.1.

H.4.1.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

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 protocol activation procedure according to H.1.8.1.

b) The LT waits until the PCD sends a valid REQA/WUPA command frame and a valid REQB/WUPB command frame, in any order and repetition.

H.4.1.3 Test report

Fill the appropriate row in Table H.25 considering results for both for Type A and Type B according to Table H.14.

Table H.14 — Result criteria for handling of the polling loop

Explanation	Test result
Only when the PCD sends at least once REQA/WUPA and at least once REQB/WUPB command frames (at least 5 ms between each type). The conditions defined in ISO/IEC 14443-3:2010, 5.1 shall apply.	PASS
Any other case	FAIL

H.4.2 Reaction of the PCD to request for waiting time extension

The purpose of this test is to determine the behavior of the PCD when the PICC uses a request for a waiting time extension (see ISO/IEC 14443-4:2008, 7.3). The mechanism of maintenance of WTX by the PCD is tested too.

This test shall be executed for at least FWI set to 0, 1 and 14 with TR0 and TR1 set to designate the default value in the LT, if it emulates a Type B PICC.

This test combination shall be executed for at least WTXM set to 1, 3 and 59 according to Table H.15.

Table H.15 — Minimum combinations

Combination	FWI	WTXM
1	0	1
2	0	3
3	0	59
4	1	1
5	1	3
6	1	59
7	14	1
8	14	3
9	14	59

H.4.2.1 Apparatus

See H.1.

H.4.2.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

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.

The UT performs the protocol activation procedure according to H.1.8.2 for Type A or H.1.8.3 for Type B.

H.4.2.2.1 Procedure 1 (ISO/IEC 14443-4:2008, 7.3)

Use the following sequence immediately after procedure H.4.2.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends block $I(0)_0$ to the LT, with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends S(WTX) request for a waiting time extension.

d) The PCD shall send S(WTX) response with INF(b6 to b1) = WTXM used.

Scenario H.16 — The PCD reaction to the LT waiting time extension request, Procedure 1

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	S(WTX) request
S(WTX) response	⟶	

H.4.2.2.1.1 Expected result

The PCD's behavior shall match the expected Scenario H.16 for all FWI and WTXM values defined in H.4.2 in all combinations tested. The test shall only pass when every performed test passed.

H.4.2.2.1.2 Test report

Fill the appropriate row in Table H.25 considering the test results both for Type A and Type B according to Table H.16.

Table H.16 — Result criteria for reaction of the PCD to request for waiting time extension, Procedure 1

Explanation	Test result
Only when the PCD sends S(WTX) response with INF(b6 to b1) = WTXM for all FWI and WTXM values defined in H.4.2 (the minimum combinations to test are given in Table H.15)	PASS
Any other case	FAIL

H.4.2.2.2 Procedure 2 (ISO/IEC 14443-4:2008, 7.3)

Use the following sequence immediately after procedure H.4.2.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends S(WTX) request.

d) The PCD shall send S(WTX) response with INF(b6 to b1) = WTXM. If it does not meet the expected response, end the test at this point.

e) Set the following bit-timing-parameters at the LT:

Parameter	Value	Reference
Temporary frame waiting time FWT_{TEMP}	WTXM × (256 × 16/*fc*) × 2^{FWI}	ISO/IEC 14443-3:2010, 7.9.4.3 ISO/IEC 14443-4:2008, 7.2 and 7.3
EGT as defined in ISO/IEC 14443-3:2010	Maximum (19 µs)	ISO/IEC 14443-3:2010, 7.1.2
NOTE The Frame response time is defined as the time between the trailing edge of the EOF of the frame received and the leading edge of the SOF of the next frame sent.		

f) The LT sends the answer to the command UT_TEST_COMMAND1, sent in b).

g) The PCD is expected to transfer the response UT_APDU (answer to the command UT_TEST_COMMAND1) back to the UT.

h) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

i) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1. The PCD shall reset the FWT at this point.

j) The LT remains Mute for at least the expected FWT. This fact means FWT timeout for the PCD.

k) Record the presence, the content and timing of the PCD-response. The PCD shall send an R(NAK) block only after FWT expires (ISO/IEC 14443-4:2008, 7.5.4.2).

l) Measure and record the time between the end of the PCD frame from step i) and start of PCD frame from step k).

Scenario H.17 — PCD reaction to the LT waiting time extension request, Procedure 2

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	S(WTX) request (WTXM belongs to the test set)
S(WTX) response	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1)
$I(0)_1$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	Mute
R(NAK)	⟶	

H.4.2.2.2.1 Expected result

The PCD's behavior shall match the expected Scenario H.17 for all FWI and WTXM values defined in H.4.2 in all combinations tested. The test shall pass only when every test with these different values passed.

H.4.2.2.2.2 Test report

Fill the appropriate row in Table H.25 for both Type A and Type B according to Table H.17.

Table H.17 — Result criteria for reaction of the PCD to request for waiting time extension, Procedure 2

Explanation	Test result
Only when the answer to TEST_COMMAND_1 was not sent back to the UT or when the PCD sent the R(NAK) before FWT expired for at least one FWI one WTXM values defined in H.4.2 (the minimum combinations to test are given in Table H.15)	FAIL
Any other case	PASS

H.4.3 Error detection and recovery

The purpose of this test is to determine the behavior of PCD when an error occurs according to ISO/IEC 14443-4:2008, 7.5.6.

NOTE In this section, "Erroneous block" is a frame with a wrong CRC.

H.4.3.1 Apparatus

See H.1.

H.4.3.2 Procedure

Place the LT into the PCD operating volume and record the presence and the content of the PCD commands.

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.

The UT performs the protocol activation procedure according to H.1.8.2 for Type A or H.1.8.3 for Type B.

H.4.3.2.1 Procedure 1 (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 12)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends an erroneous I-block to the PCD.

d) The PCD shall send $R(NAK)_0$.

e) The LT sends I-block (containing some response UT_APDU with answer to the UT_TEST_COMMAND1) to the PCD.

f) The PCD is expected to transfer the response UT_APDU back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

Scenario H.18 — Error detection and recovery of a transmission error by the PCD (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 12), Procedure 1

PCD		LT
$I(0)_0$(INF = UT_TEST_COMMAND1)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1, wrong CRC)
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1)

H.4.3.2.2 Procedure 2 (ISO/IEC 14443-4:2008, 7.5.4.2 rule 4)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends an erroneous block to the PCD.

d) The PCD shall send $R(NAK)_0$.

e) The LT sends a second invalid block to the PCD.

f) The PCD shall send either $R(NAK)_0$ or S(DESELECT) request.

Scenario H.19 — Error detection and recovery of a transmission error by the PCD, Procedure 2

PCD		LT
$I(0)_0$(INF = UT_TEST_COMMAND1)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1, wrong CRC)
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1, wrong CRC)
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC) or S(DESELECT)	⟶	

H.4.3.2.3 Procedure 3 (ISO/IEC 14443-4:2008, 7.5.4.2 rule 4)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) Maintain the LT Mute.

d) Record all responses from the PCD. The PCD shall send $R(NAK)_0$ at least once.

Scenario H.20 — Error detection and recovery of a transmission error by the PCD, Procedure 3

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	Mute
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	Mute
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC) or S(DESELECT)	⟶	

H.4.3.2.4 Procedure 4 (ISO/IEC 14443-4:2008, 7.5.4.2 rule 7)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends $R(ACK)_0$ (because of infringement of protocol rules by the LT).

d) The PCD shall send a S(DESELECT) request.

Scenario H.21 — Error detection and recovery of a transmission error by the PCD, Procedure 4

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	$R(ACK)_0$ ('AA' CID CRC or 'A2' CRC) [a]
S(DESELECT) request	⟶	

[a] If the PCD block contains a CID, the left option shall be used, else the right option shall be used.

H.4.3.2.5 Procedure 5 (with chaining) (ISO/IEC 14443-4:2008, 7.5.4.2 rule 5, ISO/IEC 14443-4:2008, Informative Annex B, Scenario 23)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT, with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends the first block $I(1)_0$ of the chain and waits for the PCD response.

d) The PCD shall send $R(ACK)_1$.

e) The LT sends an erroneous block $I(0)_1$ to the PCD.

f) The PCD shall send $R(ACK)_1$.

Scenario H.22 — Error detection and recovery of a transmission error by the PCD, Procedure 5 (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 19)

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	$I(1)_0$ (INF = the first chain of the answer to UT_TEST_COMMAND1)
$R(ACK)_1$ ('AB' CID CRC or 'A3' CRC)	⟶	
	⟵	$I(0)_1$ (INF = the last chain of the answer to UT_TEST_COMMAND1, wrong CRC)
$R(ACK)_1$	⟶	

H.4.3.2.6 Procedure 6 (ISO/IEC 14443-4:2008, 7.5.4.2)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends an erroneous block to the PCD.

d) The PCD shall send $R(NAK)_0$.

e) The LT remains Mute.

f) Record all responses from the PCD. The PCD shall send either $R(NAK)_0$ or S(DESELECT) request.

Scenario H.23 — Error detection and recovery of a transmission error by the PCD, Procedure 6

PCD		LT
$I(0)_0$(INF = UT_TEST_COMMAND1)	⟶	
	⟵	$I(0)_0$(INF = answer to UT_TEST_COMMAND1, wrong CRC)
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	Mute
$R(NAK)_0$ or S(DESELECT) request	⟶	

H.4.3.2.7 Procedure 7 (ISO/IEC 14443-4:2008, 7.5.6, Informative Annex B, Scenario 14)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends an erroneous S(WTX) request.

d) The PCD shall send $R(NAK)_0$.

e) The LT sends a valid S(WTX) request.

f) The PCD shall answer with S(WTX) response.

g) The LT sends I-block (containing some response UT_APDU with answer to the UT_TEST_COMMAND1) to the PCD.

h) The PCD is expected to transfer this response UT_APDU back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

Scenario H.24 — Error detection and recovery of a transmission error by the PCD, Procedure 7 (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 14)

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	S(WTX) request, wrong CRC
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	S(WTX) request
S(WTX) response	⟶	
	⟵	$I(0)_0$ (answer to UT_TEST_COMMAND1)

H.4.3.2.8 Procedure 8 (ISO/IEC 14443-4:2008, 7.5.6, Informative Annex B, Scenario 17)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends an S(WTX) request.

d) The PCD shall answer with S(WTX) response.

e) The LT sends an erroneous $I(0)_0$.

f) The PCD shall send $R(NAK)_0$.

g) The LT sends a valid $I(0)_0$ with maximum timing between $R(NAK)_0$ and $I(0)_0$ to check that the PCD FWT is still extended.

h) The PCD is expected to transfer this response UT_APDU back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

Scenario H.25 — Error detection and recovery of a transmission error by the PCD, Procedure 8 (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 17)

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	S(WTX) request
S(WTX) response	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1, wrong CRC)
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND1)

H.4.3.2.9 Procedure 9 (with chaining) (see ISO/IEC 14443-4:2008, Informative Annex B, Scenario 20)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND2(causing 3 chains)) to the PCD.

b) The LT waits until the PCD sends an I-block $I(1)_0$ to the LT, with the INF field the first chain of the UT_TEST_COMMAND2.

c) The LT sends an erroneous $R(ACK)_0$.

d) The PCD shall send $R(NAK)_0$.

e) The LT sends $R(ACK)_0$.

f) The PCD shall send the next block $I(1)_1$ of the chain.

g) The LT sends $R(ACK)_1$.

h) The PCD shall send the last block $I(0)_0$ of the chain .

i) The LT sends I-block (containing some response UT_APDU with answer to the UT_TEST_COMMAND2) to the PCD.

j) The PCD is expected to transfer this response UT_APDU back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

NOTE In case the UT cannot force the PCD to use 3 chains to send the TEST_COMMAND2, modify the expected procedure to reflect the test purpose which is to make sure that the block numbering and the chaining are properly executed after transmission error at step c).

Scenario H.26 — Error detection and recovery of a transmission error by the PCD, Procedure 9 (with chaining) (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 20)

PCD		LT
$I(1)_0$ (INF = the first chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$R(ACK)_0$, wrong CRC ('AA' CID '00 00' or 'A2 00 00') [a]
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	$R(ACK)_0$ ('AA' CID CRC or 'A2' CRC) [a]
$I(1)_1$ (INF = the second chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$R(ACK)_1$ ('AB' CID CRC or 'A3' CRC) [a]
$I(0)_0$ (INF = the last chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND2)

[a] If the PCD frame contains a CID, the left option shall be used, else the right option shall be used.

H.4.3.2.10 Procedure 10 (with chaining) (see ISO/IEC 14443-4:2008, Informative Annex B, Scenario 21)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND2(causing 3 chains)) to the PCD.

b) The LT waits until the PCD sends an I-block $I(1)_0$ to the LT, with the INF field the first chain of the UT_TEST_COMMAND2.

c) The LT remains Mute.

d) The PCD shall send $R(NAK)_0$.

e) The LT sends a R-block with a not synchronized sequential block number $R(ACK)_1$.

f) The PCD shall repeat the previous I-block $I(1)_0$ to the LT .

g) The LT sends $R(ACK)_0$.

h) The PCD shall send the next block $I(1)_1$ of the chain.

i) The LT sends $R(ACK)_1$.

j) The PCD shall send the last block $I(0)_0$ of the chain.

k) The LT sends I-block (containing some response UT_APDU with answer to the UT_TEST_COMMAND2) to the PCD.

l) The PCD is expected to transfer this response UT_APDU back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

NOTE In case the UT cannot force the PCD to use 3 chains to send the TEST_COMMAND2, modify the expected procedure to reflect the test purpose which is to make sure that the block numbering and the chaining are properly executed after Mute at step c).

Scenario H.27 — Error detection and recovery of a transmission error by the PCD, Procedure 10 (with chaining) (see ISO/IEC 14443-4:2008, Informative Annex B, Scenario 21)

PCD		LT
$I(1)_0$ (INF = the first chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$R(ACK)_0$ ('AA' CID CRC or 'A2' CRC) [a]
$I(1)_1$ (INF = the second chain of the UT_TEST_COMMAND2)	⟶	
	⟵	Mute
$R(NAK)_1$ ('BB' CID CRC or 'B3' CRC)	⟶	
	⟵	$R(ACK)_0$ ('AA' CID CRC or 'A2' CRC) [a]
$I(1)_1$ (INF = the second chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$R(ACK)_1$ ('AB' CID CRC or 'A3' CRC) [a]
$I(0)_0$ (INF = the last chain of the UT_TEST_COMMAND2)	⟶	
	⟵	$I(0)_0$ (INF = answer to UT_TEST_COMMAND2)

[a] If the PCD block contains a CID, the left option shall be used, else the right option shall be used.

H.4.3.2.11 Procedure 11 (with chaining) (see ISO/IEC 14443-4:2008, Informative Annex B, Scenario 24)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block $I(0)_0$ to the LT, with the INF field containing the UT_TEST_COMMAND1.

c) The LT sends I-block $I(1)_0$ indicating chaining.

d) The PCD shall send $R(ACK)_1$.

e) The LT sends erroneous I-block $I(1)_1$ to the PCD.

f) The PCD shall send $R(ACK)_1$.

g) The LT retransmits an I-block $I(1)_1$ without error.

h) The PCD shall send $R(ACK)_0$.

i) The LT sends the last block of the chain in I-block $I(0)_0$ (of its response UT_APDU with answer to the UT_TEST_COMMAND1) to the PCD.

j) The PCD is expected to transfer the response UT_APDU, containing all chaining segments, back to the UT. Check at the UT that this response UT_APDU block is correctly accepted.

Scenario H.28 — Error detection and recovery of a transmission error by the PCD, Procedure 11 (with chaining) (ISO/IEC 14443-4:2008, Informative Annex B, Scenario 24)

PCD		LT
$I(0)_0$ (INF = UT_TEST_COMMAND1)	⟶	
	⟵	$I(1)_0$ (INF = the first chain of the answer to UT_TEST_COMMAND1)
$R(ACK)_1$ ('AB' CID CRC or 'A3' CRC)	⟶	
	⟵	$I(1)_1$ (INF = the second chain of the answer to UT_TEST_COMMAND1, wrong CRC)
$R(ACK)_1$ ('AB' CID CRC or 'A3' CRC)	⟶	
	⟵	$I(1)_1$ (INF = the second chain of the answer to UT_TEST_COMMAND1)
$R(ACK)_0$ ('AA' CID CRC or 'A2' CRC)	⟶	
	⟵	$I(0)_0$ (INF = the last chain of the answer to UT_TEST_COMMAND1)

H.4.3.2.12 Procedure 12 (ISO/IEC 14443-4:2008, 7.5.4.2 rule 8)

Use the following sequence immediately after procedure H.4.3.2:

a) The UT sends the SEND_UT_APDU(UT_TEST_COMMAND1) to the PCD.

b) The LT waits until the PCD sends an I-block to the LT with the INF field containing the UT_TEST_COMMAND1.

c) The LT remains Mute.

d) The LT waits until the PCD sends $R(NAK)_0$. (R-block may be sent more than once.)

e) The LT remains Mute.

f) The LT waits until the PCD sends a S(DESELECT) request to the LT.

g) The LT remains Mute.

h) Record the response from the PCD. The PCD shall either retransmit the S(DESELECT) request or ignore the LT.

Scenario H.29 — Error detection and recovery of a transmission error by the PCD, Procedure 12

PCD		LT
$I(0)_0$(INF = UT_TEST_COMMAND1)	⟶	
	⟵	Mute
$R(NAK)_0$ ('BA' CID CRC or 'B2' CRC)	⟶	
	⟵	Mute
S(DESELECT) request 'CA' CID CRC or 'C2' CRC	⟶	
	⟵	Mute
PCD response	⟶	

H.4.3.3 Test report

Fill the appropriate row in Table H.25 for both Type A and Type B according to Table H.18.

Table H.18 — Result criteria for error detection and recovery

Explanation	Test result
Only when the PCD's behavior matches each procedure expected Scenario	PASS
Any other case	FAIL

H.4.4 Handling of NAD during chaining

The purpose of this test is to ensure that the PCD maintains NAD in the proper way.

H.4.4.1 Apparatus

See H.1.

H.4.4.2 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) Configure the LT as one supporting NAD.

b) Repeat procedure from Scenario H.26.

H.4.4.3 Test report

Fill the appropriate row in Table H.25 for both Type A and Type B according to Table H.19.

Table H.19 — Result criteria for handling of NAD during chaining

Explanation	Test result
Only when the PCD use the NAD only in the first packet of chaining or does not use NAD.	PASS
Any other case	FAIL

H.5 Continuous monitoring of packets sent by the PCD

The purpose of this test is to ensure that the PCD does not set any RFU bits in any sent frame to any value other than the default value documented for such RFU bit. Further, the test shall also ensure that no field is ever set to RFU value. The test shall also ensure that the R-block and the S-block match the protocol definitions and that rules given regarding the first byte of the packet are not violated.

H.5.1 RFU fields

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

H.5.2 RFU values

Functional fields shall be continuously monitored during the testing and shall always be verified to contain only functional values documented in the standard or proprietary values documented as such in the standard. A test shall fail and the tested PCD be declared non-compliant in case a functional field is not set to said values at any time.

H.5.3 R-block

R-block shall never contain an INF field (see ISO/IEC 14443-4:2008, 7.1.1.1).

H.5.4 S-block

S-block shall have an INF field of one byte only when it is a WTX block and no INF field otherwise (see ISO/IEC 14443-4:2008, 7.1.1.1).

H.5.5 PCB

PCB byte shall contain allowed values (see ISO/IEC 14443-4:2008, 7.1.1.1 and ISO/IEC 14443-4:2008, Annex C).

H.5.6 Type A initialization frames

Type A initialization frames shall contain allowed values (see ISO/IEC 14443-3:2010, 6.4.1 and 6.5.3.2).

H.5.7 Apparatus

See H.1.

H.5.8 Procedure

During all test procedures and Scenarios the logical content of the communication shall always be recorded.

H.5.9 Test report

Fill the appropriate rows in Table H.25 and Table H.27 for both Type A and Type B according to Table H.20.

Table H.20 — Result criteria for continuous monitoring of packets sent by the PCD

Explanation	Test result
When the PCD satisfies all the following conditions; - The PCD sets default values to RFU bits in all sent frames. - The PCD does not set RFU value to any field. - The PCD does not violate the length rules of R-block and S-block. - The PCD does not violate the first byte coding rules of Block and Frame (as summarized in ISO/IEC 14443-4:2008, Annex C).	PASS
When the PCD satisfies at least one of the following conditions; - The PCD sets other than the default value to at least one RFU bit in any sent frame. - The PCD sets any field to a RFU value. - The PCD violates the length rules of R-block or S-block. - The PCD violates the first byte coding rules of Block and Frame (as summarized in ISO/IEC 14443-4:2008, Annex C).	FAIL

H.6 Reported results

Table H.21 — Type A specific timing table

No	Parameter	ISO Reference	Reference value	Measured value
1	Frame delay time PICC to PCD	ISO/IEC 14443-3:2010, 6.2.1.2	at least 1172/*fc* (~ 86 µs)	
2	Request Guard Time	ISO/IEC 14443-3:2010, 6.2.2	at least 7000/*fc* (~ 512 µs)	

NOTE All timing values are calculated for carrier frequency *fc* = 13,56 MHz and bit rate *fc*/128 (~106 kbit/s).

Table H.22 — Type B specific timing table

No	Parameter	ISO Reference	Minimum value	Maximum value	Measured value
1	SOF low	ISO/IEC 14443-3:2010, 7.1.4	10 etu (~94,40 µs)	11 etu (~103,83 µs)	
2	EOF low	ISO/IEC 14443-3:2010, 7.1.5	10 etu (~94,40 µs)	11 etu (~103,83 µs)	
3	Bit boundaries	ISO/IEC 14443-3:2010, 7.1.1	(n – 0,125) etu	(n + 0,125) etu	
4	EGT PCD to PICC	ISO/IEC 14443-3:2010, 7.1.2	0 µs	57 µs	
5	Minimum delay between the PICC EOF start and PCD SOF start	ISO/IEC 14443-3:2010, 7.1.7	10 etu + 32/*fs*	No maximum	

NOTE All timing values are calculated for carrier frequency *fc* = 13,56 MHz and bit rate *fc*/128 (~106 kbit/s).

Table H.23 — 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 [a]**
H.2.1	Frame delay time PICC to PCD		
H.2.2	Request Guard Time		
H.2.3	Handling of bit collision during ATQA		
H.2.4	Handling of anticollision loop	Scenario H.1	
		Scenario H.2	
		Scenario H.3	
		Scenario H.4	
H.2.5	Handling of RATS and ATS	Scenario H.5	
		Scenario H.6	
		Scenario H.7	
H.2.6	Handling of PPS response	Scenario H.8	
		Scenario H.9	
H.2.7	Frame size selection mechanism	Scenario H.10	
H.2.8	Handling of Start-up Frame Guard Time	Scenario H.11	
H.2.9	Handling of the CID during activation by the PCD	Scenario H.12	

[a] In case a test has several procedures indicate PASS only in case every individual procedure is PASS.

Table H.24 — 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 [a]**
H.3.1	I/O transmission timing		
H.3.2	Frame size selection mechanism	Scenario H.13	
H.3.3	Handling of the CID during activation by the PCD	Scenario H.14	
		Scenario H.15	

[a] In case a test has several procedures indicate PASS only in case every individual procedure is PASS.

Table H.25 — Reported results for test method for logical operations of the PCD

Test method from ISO/IEC 10373-6		Scenario numbers		Test result	
Clause	**Parameter**	**ISO/IEC 10373-6**	**ISO/IEC 14443-4:2008, Informative Annex B**	**Type A [a]**	**Type B [a]**
H.4.1	Handling of the polling loop				
H.4.2	Reaction of the PCD to request for waiting time extension	Scenario H.16			
		Scenario H.17			
H.4.3	Error detection and recovery	Scenario H.18	Scenario 8 "Exchange of I-blocks"		
		Scenario H.19			
		Scenario H.20			
		Scenario H.21			
		Scenario H.22	Scenario 19 "PICC uses chaining"		
		Scenario H.23			
		Scenario H.24	Scenario 14 "Request for waiting time extension"		
		Scenario H.25	Scenario 17 "Request for waiting time extension"		
		Scenario H.26	Scenario 20 "PCD uses chaining"		
		Scenario H.27**Error! Reference source not found.**	Scenario 21 "PCD uses chaining"		
		Scenario H.28	Scenario 24 "PICC uses chaining"		
		Scenario H.29			
H.4.4	Handling of NAD during chaining				
H.5	Continuous monitoring of packets sent by the PCD				

[a] In case a test has several procedures, indicate PASS only when every individual procedure is PASS.

Table H.26 — Test coverage report

No	Parameter	Description	Information
1	Chaining	Tested only if there is a command that supports more than 16 bytes	
2	NAD handling		

Table H.27 — PCD RFU table report

Name	PCD command	RFU field/value	Value		Test result
			Default	Not allowed	PASS or FAIL or Not Done
Short frame Type A	REQA/WUPA	RFU values		All values specified as RFU in ISO/IEC 14443-3: 2010, Table 3	
SEL coding	SEL	RFU values		All values specified as RFU in ISO/IEC 14443-3: 2010, Table 7	
AFI	REQB/WUPB	RFU values		All values specified as RFU in ISO/IEC 14443-3: 2010, Table 22	
PARAM	REQB/WUPB	RFU field (b8 to b6)	(000)b	All other values	
		RFU values in number of slots (b3 to b1)		(101)b (110)b (111)b	
Param 1	ATTRIB	RFU field (b2 to b1)	(00)b	All other values	
Minimum TR0	ATTRIB	RFU values (b8 to b7)		(11)b	
Minimum TR1	ATTRIB	RFU values (b6 to b5)		(11)b	
Param 2	ATTRIB	RFU values (b4 to b1)		All values from '9'((1001)b) up to 'F'((1111)b)	
Param 3	ATTRIB	RFU field (b8 to b4)	(00000)b	All other values	
Param 4	ATTRIB	RFU field (b8 to b5)	(0000)b	All other values	
		RFU value (b4 to b1)		'F'((1111)b)	

Annex I
(normative)

Removed

Annex J
(normative)

High bit rate selection test methods for PCD

J.1 Apparatus

In this test the PCD-test-apparatus shall be configurable to change the bit rate during the test procedure. Tester shall be able to measure the bit rate used by the PCD on each stage of this test procedure.

J.2 Procedure

Place the PCD-test-apparatus into the field of the PCD.

J.2.1 Procedure for Type A

The following procedure shall be repeated for all values of interface byte TA(1) defined in Table J.1:

a) Run through activation sequence as defined in ISO/IEC 14443-3:2010.

b) The PCD shall send a RATS command as defined in ISO/IEC 14443-4:2008.

c) The PCD-test-apparatus answers with a valid ATS including TA(1) according to Table J.1.

d) The PCD may optionally send a PPS with a valid parameter setting for PPS1 byte according to Table J.1.

e) If the PCD has sent a PPS then the PCD-test-apparatus acknowledges the received PPS with a valid PPS response.

f) The PCD shall send $I(0)_0$ block using the bit rate selected.

NOTE 1 This block may also be $I(1)_0$, or R(NAK) in case of PICC presence check method 2a as described in ISO/IEC 14443-4:2008, 7.5.5.2.

g) The PCD-test-apparatus sends a valid response using the bit rate selected. Check, if the answer from the PCD-test apparatus is accepted by the PCD.

NOTE 2 The following steps may not be applicable when the PCD is embedded in a product:

h) The PCD shall send a S(DESELECT) request using the bit rate selected.

i) The PCD-test-apparatus sends a valid S(DESELECT) response using the bit rate selected. Check, if the answer from the PCD-test apparatus is accepted by the PCD.

j) The PCD shall send a valid REQA command frame using the bit rate *fc*/128.

k) The PCD-test-apparatus answers with a valid ATQA.

Table J.1 — Correct behavior of PCD after ATS with TA(1)

TA(1)	Valid parameter setting for PPS1
(10000000)b	(00000000)b [a]
(10010001)b	(00000101)b, (00000000)b
(10100010)b	(00001010)b, (00000000)b
(10110011)b	(00000101)b, (00001010)b, (00000000)b
(11000100)b	(00001111)b, (00000000)b
(11010101)b	(00000101)b, (00001111)b, (00000000)b
(11100110)b	(00001010)b, (00001111)b, (00000000)b
(11110111)b	(00000101)b, (00001010)b, (00001111)b, (00000000)b
(00000000)b	(00000000)b [a]
(00000001)b	(00000001)b, (00000000)b
(00000010)b	(00000010)b, (00000000)b
(00000011)b	(00000001)b, (00000010)b, (00000000)b
(00000100)b	(00000011)b, (00000000)b
(00000101)b	(00000001)b, (00000011)b, (00000000)b
(00000110)b	(00000010)b, (00000011)b, (00000000)b
(00000111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b
(00010000)b	(00000000)b (00000100)b
(00010001)b	(00000001)b, (00000000)b (00000101)b, (00000100)b
(00010010)b	(00000010)b, (00000000)b (00000110)b, (00000100)b
(00010011)b	(00000001)b, (00000010)b, (00000000)b (00000101)b, (00000110)b, (00000100)b
(00010100)b	(00000011)b, (00000000)b (00000111)b, (00000100)b
(00010101)b	(00000001)b, (00000011)b, (00000000)b (00000101)b, (00000111)b, (00000100)b
(00010110)b	(00000010)b, (00000011)b, (00000000)b (00000110)b, (00000111)b, (00000100)b
(00010111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00000101)b, (00000110)b, (00000111)b, (00000100)b
(00100000)b	(00000000)b (00001000)b
(00100001)b	(00000001)b, (00000000)b (00001001)b, (00001000)b
(00100010)b	(00000010)b, (00000000)b (00001010)b, (00001000)b
(00100011)b	(00000001)b, (00000010)b, (00000000)b (00001001)b, (00001010)b, (00001000)b
(00100100)b	(00000011)b, (00000000)b (00001011)b, (00001000)b

TA(1)	Valid parameter setting for PPS1
(00100101)b	(00000001)b, (00000011)b, (00000000)b (00001001)b, (00001011)b, (00001000)b
(00100110)b	(00000010)b, (00000011)b, (00000000)b (00001010)b, (00001011)b, (00001000)b
(00100111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00001001)b, (00001010)b, (00001011)b, (00001000)b
(00110000)b	(00000000)b (00000100)b (00001000)b
(00110001)b	(00000001)b, (00000000)b (00000101)b, (00000100)b (00001001)b, (00001000)b
(00110010)b	(00000010)b, (00000000)b (00000110)b, (00000100)b (00001010)b, (00001000)b
(00110011)b	(00000001)b, (00000010)b, (00000000)b (00000101)b, (00000110)b, (00000100)b (00001001)b, (00001010)b, (00001000)b
(00110100)b	(00000011)b, (00000000)b (00000111)b, (00000100)b (00001011)b, (00001000)b
(00110101)b	(00000001)b, (00000011)b, (00000000)b (00000101)b, (00000111)b, (00000100)b (00001001)b, (00001011)b, (00001000)b
(00110110)b	(00000010)b, (00000011)b, (00000000)b (00000110)b, (00000111)b, (00000100)b (00001010)b, (00001011)b, (00001000)b
(00110111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00000101)b, (00000110)b, (00000111)b, (00000100)b (00001001)b, (00001010)b, (00001011)b, (00001000)b
(01000000)b	(00000000)b (00001100)b
(01000001)b	(00000001)b, (00000000)b (00001101)b, (00001100)b
(01000010)b	(00000010)b, (00000000)b (00001110)b, (00001100)b
(01000011)b	(00000001)b, (00000010)b, (00000000)b (00001101)b, (00001110)b, (00001100)b
(01000100)b	(00000011)b, (00000000)b (00001111)b, (00001100)b
(01000101)b	(00000001)b, (00000011)b, (00000000)b (00001101)b, (00001111)b, (00001100)b
(01000110)b	(00000010)b, (00000011)b, (00000000)b (00001110)b, (00001111)b, (00001100)b
(01000111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00001101)b, (00001110)b, (00001111)b, (00001100)b
(01010000)b	(00000000)b (00000100)b (00001100)b

TA(1)	Valid parameter setting for PPS1
(01010001)b	(00000001)b, (00000000)b (00000101)b, (00000100)b (00001101)b, (00001100)b
(01010010)b	(00000010)b, (00000000)b (00000110)b, (00000100)b (00001110)b, (00001100)b
(01010011)b	(00000001)b, (00000010)b, (00000000)b (00000101)b, (00000110)b, (00000100)b (00001101)b, (00001110)b, (00001100)b
(01010100)b	(00000011)b, (00000000)b (00000111)b, (00000100)b (00001111)b, (00001100)b
(01010101)b	(00000001)b, (00000011)b, (00000000)b (00000101)b, (00000111)b, (00000100)b (00001101)b, (00001111)b, (00001100)b
(01010110)b	(00000010)b, (00000011)b, (00000000)b (00000110)b, (00000111)b, (00000100)b (00001110)b, (00001111)b, (00001100)b
(01010111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00000101)b, (00000110)b, (00000111)b, (00000100)b (00001101)b, (00001110)b, (00001111)b, (00001100)b
(01100000)b	(00000000)b (00001000)b (00001100)b
(01100001)b	(00000001)b, (00000000)b (00001001)b, (00001000)b (00001101)b, (00001100)b
(01100010)b	(00000010)b, (00000000)b (00001010)b, (00001000)b (00001110)b, (00001100)b
(01100011)b	(00000001)b, (00000010)b, (00000000)b (00001001)b, (00001010)b, (00001000)b (00001101)b, (00001110)b, (00001100)b
(01100100)b	(00000011)b, (00000000)b (00001011)b, (00001000)b (00001111)b, (00001100)b
(01100101)b	(00000001)b, (00000011)b, (00000000)b (00001001)b, (00001011)b, (00001000)b (00001101)b, (00001111)b, (00001100)b
(01100110)b	(00000010)b, (00000011)b, (00000000)b (00001010)b, (00001011)b, (00001000)b (00001110)b, (00001111)b, (00001100)b
(01100111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00001001)b, (00001010)b, (00001011)b, (00001000)b (00001101)b, (00001110)b, (00001111)b, (00001100)b
(01110000)b	(00000000)b (00000100)b (00001000)b (00001100)b
(01110001)b	(00000001)b, (00000000)b (00000101)b, (00000100)b (00001001)b, (00001000)b (00001101)b, (00001100)b

TA(1)	Valid parameter setting for PPS1
(01110010)b	(00000010)b, (00000000)b (00000110)b, (00000100)b (00001010)b, (00001000)b (00001110)b, (00001100)b
(01110011)b	(00000001)b, (00000010)b, (00000000)b (00000101)b, (00000110)b, (00000100)b (00001001)b, (00001010)b, (00001000)b (00001101)b, (00001110)b, (00001100)b
(01110100)b	(00000011)b, (00000000)b (00000111)b, (00000100)b (00001011)b, (00001000)b (00001111)b, (00001100)b
(01110101)b	(00000001)b, (00000011)b, (00000000)b (00000101)b, (00000111)b, (00000100)b (00001001)b, (00001011)b, (00001000)b (00001101)b, (00001111)b, (00001100)b
(01110110)b	(00000010)b, (00000011)b, (00000000)b (00000110)b, (00000111)b, (00000100)b (00001010)b, (00001011)b, (00001000)b (00001110)b, (00001111)b, (00001100)b
(01110111)b	(00000001)b, (00000010)b, (00000011)b, (00000000)b (00000101)b, (00000110)b, (00000111)b, (00000100)b (00001001)b, (00001010)b, (00001011)b, (00001000)b (00001101)b, (00001110)b, (00001111)b, (00001100)b
[a] PPS command is useless in this case and may not be supported by the PICC.	

Scenario J.1 — High bit rate selection, Type A, Procedure 1

PCD		PCD-test-apparatus
RATS command frame ('E0 01' CRC_A)	⟶	
	⟵	ATS, TA(1) according to Table J.1
Optional PPS request according to Table J.1 — Correct behavior of PCD after ATS with TA(1)	⟶	
	⟵	PPS response (using bit rate *fc*/128)
$I(0)_0$ (using selected bit rate)	⟶	
	⟵	$I(0)_0$ (using selected bit rate)
S(DESELECT) request	⟶	
	⟵	S(DESELECT) response (using selected bit rate)
WUPA (using bit rate *fc*/128)	⟶	
	⟵	ATQA (using bit rate *fc*/128)

J.2.1.1 Expected result

The PCD shall behave as described in Scenario J.1 in each of the 72 test cases.

J.2.1.2 Test report

If the PCD behaves valid according to Scenario J.1 in each of the 72 test cases, then this test passed. The test report should document the bit rates chosen by the PCD in each of the 72 test cases.

J.2.2 Procedure for Type B

The following procedure shall be repeated for all values of the protocol info byte Bit_Rate_capability defined in Table J.2:

a) The PCD shall send a valid REQB command frame.

b) The PCD-test-apparatus answers with a valid ATQB including Bit_Rate_capability byte according to Table J.2.

c) The PCD shall send an ATTRIB command with a valid parameter setting for Param 2 byte according to Table J.2.

d) The PCD-test-apparatus acknowledges the received ATTRIB with a valid Answer to ATTRIB command.

e) PCD shall send $I(0)_0$ block using the bit rate selected with Param 2.

NOTE 1 This block may also be $I(1)_0$, or R(NAK) in case of PICC presence check method 2a as described in ISO/IEC 14443-4:2008, 7.5.5.2.

f) The PCD-test-apparatus sends a valid response using the bit rate selected with Param 2. Check, if the answer from the PCD-test apparatus is accepted by the PCD.

NOTE 2 The following steps may not be applicable when the PCD is embedded in a product:

g) The PCD shall send a S(DESELECT) request using the bit rate selected.

h) The PCD-test-apparatus sends a valid S(DESELECT) response using the bit rate selected. Check, if the answer from the PCD-test apparatus is accepted by the PCD.

i) The PCD shall send a valid REQB command frame using the bit rate *fc*/128.

j) PCD-test-apparatus answers with a valid ATQB including Bit_Rate_capability byte according to Table J.2.

Table J.2 — Correct behavior of PCD after ATQB

Bit_Rate_capability	Valid parameter setting for Param 2 [a]
(10000000)b	(0000xxxx)b
(10010001)b	(0101xxxx)b, (0000xxxx)b
(10100010)b	(1010xxxx)b, (0000xxxx)b
(10110011)b	(0101xxxx)b, (1010xxxx)b, (0000xxxx)b
(11000100)b	(1111xxxx)b, (0000xxxx)b
(11010101)b	(0101xxxx)b, (1111xxxx)b, (0000xxxx)b
(11100110)b	(1010xxxx)b, (1111xxxx)b, (0000xxxx)b
(11110111)b	(0101xxxx)b, (1010xxxx)b, (1111xxxx)b, (0000xxxx)b
(00000000)b	(0000xxxx)b
(00000001)b	(0001xxxx)b, (0000xxxx)b
(00000010)b	(0010xxxx)b, (0000xxxx)b
(00000011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b
(00000100)b	(0011xxxx)b, (0000xxxx)b
(00000101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b
(00000110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b
(00000111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b
(00010000)b	(0000xxxx)b (0100xxxx)b
(00010001)b	(0001xxxx)b, (0000xxxx)b (0101xxxx)b, (0100xxxx)b
(00010010)b	(0010xxxx)b, (0000xxxx)b (0110xxxx)b, (0100xxxx)b
(00010011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0100xxxx)b
(00010100)b	(0011xxxx)b, (0000xxxx)b (0111xxxx)b, (0100xxxx)b
(00010101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0111xxxx)b, (0100xxxx)b

Bit_Rate_capability	Valid parameter setting for Param 2 [a]
(00010110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0110xxxx)b, (0111xxxx)b, (0100xxxx)b
(00010111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0111xxxx)b, (0100xxxx)b
(00100000)b	(0000xxxx)b (1000xxxx)b
(00100001)b	(0001xxxx)b, (0000xxxx)b (1001xxxx)b, (1000xxxx)b
(00100010)b	(0010xxxx)b, (0000xxxx)b (1010xxxx)b, (1000xxxx)b
(00100011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (1001xxxx)b, (1010xxxx)b, (1000xxxx)b
(00100100)b	(0011xxxx)b, (0000xxxx)b (1011xxxx)b, (1000xxxx)b
(00100101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (1001xxxx)b, (1011xxxx)b, (1000xxxx)b
(00100110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1010xxxx)b, (1011xxxx)b, (1000xxxx)b
(00100111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1001xxxx)b, (1010xxxx)b, (1011xxxx)b, (1000xxxx)b
(00110000)b	(0000xxxx)b (0100xxxx)b (1000xxxx)b
(00110001)b	(0001xxxx)b, (0000xxxx)b (0101xxxx)b, (0100xxxx)b (1001xxxx)b, (1000xxxx)b
(00110010)b	(0010xxxx)b, (0000xxxx)b (0110xxxx)b, (0100xxxx)b (1010xxxx)b, (1000xxxx)b
(00110011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0100xxxx)b (1001xxxx)b, (1010xxxx)b, (1000xxxx)b
(00110100)b	(0011xxxx)b, (0000xxxx)b (0111xxxx)b, (0100xxxx)b (1011xxxx)b, (1000xxxx)b
(00110101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0111xxxx)b, (0100xxxx)b (1001xxxx)b, (1011xxxx)b, (1000xxxx)b
(00110110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1010xxxx)b, (1011xxxx)b, (1000xxxx)b
(00110111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1001xxxx)b, (1010xxxx)b, (1011xxxx)b, (1000xxxx)b
(01000000)b	(0000xxxx)b (1100xxxx)b
(01000001)b	(0001xxxx)b, (0000xxxx)b (1101xxxx)b, (1100xxxx)b
(01000010)b	(0010xxxx)b, (0000xxxx)b (1110xxxx)b, (1100xxxx)b

Bit_Rate_capability	Valid parameter setting for Param 2 [a]
(01000011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (1101xxxx)b, (1110xxxx)b, (1100xxxx)b
(01000100)b	(0011xxxx)b, (0000xxxx)b (1111xxxx)b, (1100xxxx)b
(01000101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (1101xxxx)b, (1111xxxx)b, (1100xxxx)b
(01000110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01000111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1101xxxx)b, (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01010000)b	(0000xxxx)b (0100xxxx)b (1100xxxx)b
(01010001)b	(0001xxxx)b, (0000xxxx)b (0101xxxx)b, (0100xxxx)b (1101xxxx)b, (1100xxxx)b
(01010010)b	(0010xxxx)b, (0000xxxx)b (0110xxxx)b, (0100xxxx)b (1110xxxx)b, (1100xxxx)b
(01010011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0100xxxx)b (1101xxxx)b, (1110xxxx)b, (1100xxxx)b
(01010100)b	(0011xxxx)b, (0000xxxx)b (0111xxxx)b, (0100xxxx)b (1111xxxx)b, (1100xxxx)b
(01010101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0111xxxx)b, (0100xxxx)b (1101xxxx)b, (1111xxxx)b, (1100xxxx)b
(01010110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01010111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1101xxxx)b, (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01100000)b	(0000xxxx)b (1000xxxx)b (1100xxxx)b
(01100001)b	(0001xxxx)b, (0000xxxx)b (1001xxxx)b, (1000xxxx)b (1101xxxx)b, (1100xxxx)b
(01100010)b	(0010xxxx)b, (0000xxxx)b (1010xxxx)b, (1000xxxx)b (1110xxxx)b, (1100xxxx)b
(01100011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (1001xxxx)b, (1010xxxx)b, (1000xxxx)b (1101xxxx)b, (1110xxxx)b, (1100xxxx)b
(01100100)b	(0011xxxx)b, (0000xxxx)b (1011xxxx)b, (1000xxxx)b (1111xxxx)b, (1100xxxx)b
(01100101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (1001xxxx)b, (1011xxxx)b, (1000xxxx)b (1101xxxx)b, (1111xxxx)b, (1100xxxx)b

Bit_Rate_capability	Valid parameter setting for Param 2 [a]
(01100110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1010xxxx)b, (1011xxxx)b, (1000xxxx)b (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01100111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (1001xxxx)b, (1010xxxx)b, (1011xxxx)b, (1000xxxx)b (1101xxxx)b, (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01110000)b	(0000xxxx)b (0100xxxx)b (1000xxxx)b (1100xxxx)b
(01110001)b	(0001xxxx)b, (0000xxxx)b (0101xxxx)b, (0100xxxx)b (1001xxxx)b, (1000xxxx)b (1101xxxx)b, (1100xxxx)b
(01110010)b	(0010xxxx)b, (0000xxxx)b (0110xxxx)b, (0100xxxx)b (1010xxxx)b, (1000xxxx)b (1110xxxx)b, (1100xxxx)b
(01110011)b	(0001xxxx)b, (0010xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0100xxxx)b (1001xxxx)b, (1010xxxx)b, (1000xxxx)b (1101xxxx)b, (1110xxxx)b, (1100xxxx)b
(01110100)b	(0011xxxx)b, (0000xxxx)b (0111xxxx)b, (0100xxxx)b (1011xxxx)b, (1000xxxx)b (1111xxxx)b, (1100xxxx)b
(01110101)b	(0001xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0111xxxx)b, (0100xxxx)b (1001xxxx)b, (1011xxxx)b, (1000xxxx)b (1101xxxx)b, (1111xxxx)b, (1100xxxx)b
(01110110)b	(0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1010xxxx)b, (1011xxxx)b, (1000xxxx)b (1110xxxx)b, (1111xxxx)b, (1100xxxx)b
(01110111)b	(0001xxxx)b, (0010xxxx)b, (0011xxxx)b, (0000xxxx)b (0101xxxx)b, (0110xxxx)b, (0111xxxx)b, (0100xxxx)b (1001xxxx)b, (1010xxxx)b, (1011xxxx)b, (1000xxxx)b (1101xxxx)b, (1110xxxx)b, (1111xxxx)b, (1100xxxx)b

[a] The least significant half byte of Param 2 is used to code the maximum frame size that can be received by the PCD.

Scenario J.2 — High bit rate selection, Type B, Procedure 2

PCD		PCD-test-apparatus
REQB ('05 00 00' CRC_B)	⟶	
	⟵	ATQB, Bit_Rate_capability according to Table J.2
ATTRIB command, Param 2 according to Table J.2	⟶	
	⟵	Answer to ATTRIB command (using bit rate *fc*/128)
$I(0)_0$ (using selected bit rate)	⟶	
	⟵	$I(0)_0$ (using selected bit rate)
S(DESELECT) request	⟶	
	⟵	S(DESELECT) response (using selected bit rate)
WUPB (using bit rate *fc*/128)	⟶	
	⟵	ATQB (using bit rate *fc*/128)

J.2.2.1 Expected result

The PCD shall behave as described in Scenario J.2 in each of the 72 test cases.

J.2.2.2 Test report

If the PCD behaves valid according to Scenario J.2 in each of the 72 test cases, then this test passed. The test report should document the bit rates chosen by the PCD in each of the 72 test cases.

Bibliography

[1] ISO/IEC 9646-1:1994, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 1: General concepts*

[2] ISO/IEC 9646-2:1994, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 2: Abstract Test Suite specification*

[3] ISO/IEC 9646-3:1998, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 3: The Tree and Tabular Combined Notation (TTCN)*

[4] ISO/IEC 9646-4:1994, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 4: Test realization*

[5] ISO/IEC 9646-5:1994, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 5: Requirements on test laboratories and clients for the conformance assessment process*

[6] ISO/IEC 9646-6:1994, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 6: Protocol profile test specification*

[7] ISO/IEC 9646-7:1995, *Information technology − Open Systems Interconnection − Conformance testing methodology and framework − Part 7: Implementation Conformance Statements*