ISO/IEC JTC 1/SC 17 N 1654

Date: 1999-11-17

ISO/IEC CD 15693-3

ISO/IEC JTC 1/SC 17/WG 8

Secretariat: DIN

Identification cards — Contactless integrated circuit(s) cards - Vicinity cards — Part 3: Anticollision and transmission protocol

Cartes d'identification — Cartes à circuit(s) intégré(s) sans contact - Cartes de Vicinité — Partie 3 : Anticollision et protocole de transmission

Document type: International Standard
Document subtype:
Document stage: (30) Commitee
Document language: E
D:\1Standards\ISO-TF3\15693-3\ISO-IEC 15693-3 (E).doc STD Version 1.0

Contents

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 15693 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 15693-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 17, *Cards and personal identification*.

This second/third/... edition cancels and replaces the first/second/... edition (), [clause(s) / subclause(s) / table(s) / figure(s) / annex(es)] of which [has / have] been technically revised.

ISO/IEC 15693 consists of the following parts, under the general title *Identification cards — Contactless integrated circuit(s) cards - Vicinity cards*:

— *Part 1: Physical characteristics*

— *Part 2: Radio frequency power and signal interface*

— *Part 3: Anticollision and transmission protocol*

— *Part 4: Extended command set and security features*

Introduction

ISO/IEC 15693 is one of a series of International Standards describing the parameters for identification cards as defined in ISO/IEC 7810 and the use of such cards for international interchange.

This part of ISO/IEC 15693 describes the anticollision and transmission protocols

This International Standard does not preclude the incorporation of other standard technologies on the card.

Contactless card standards cover a variety of types as embodied in ISO/IEC 10536 (Close-coupled cards), ISO/IEC 14443 (Proximity cards), ISO/IEC 15693 (Vicinity cards). These are intended for operation when very near, nearby and at a longer distance from associated coupling devices respectively.

Identification Cards — Contactless integrated circuit(s) cards - Vicinity cards — Part 3: Anticollision and transmission protocol

1 Scope

This part of ISO/IEC 15693 describes:

- protocol and commands,
- other parameters required to initialize communications between a VICC and a VCD,
- methods to detect and communicate with one card among several cards ("anticollision"),
- optional means to ease and speed up the selection of one among several cards based on application criteria.

2 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of this part of ISO/IEC 15693. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 15693 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 15693, *Identification cards - Contactless integrated circuit(s) cards – Vicinity cards - Part 1: Physical characteristics. Part 2: Radio frequency power and signal interface*

ISO/IEC 10373, *Identification cards - Test methods – Part 7: Vicinity cards*

ISO/IEC 13239, *Information technology – Telecommunications and information exchange between systems – High level data link control (HDLC) procedures*

ISO/IEC 7816-5, *Identification cards – Integrated circuit(s) card with contacts – Part 5: Numbering system and registration procedure for application identifiers*

3 Definitions, abbreviations and symbols

3.1 Definitions

3.1.1 Anticollision loop

Algorithm used to prepare for and handle a dialogue between a VCD and one or more VICCs from several in its energizing field.

3.1.2 Byte

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

3.2 Abbreviations

AFI	Application Family Identifier
CRC	Cyclic Redundancy Check
DSFID	Data Storage Format IDentifier
EOF	End Of Frame
LSB	Least Significant Bit
MSB	Most Significant Bit
RFU	Reserved for Future Use
SOF	Start Of Frame
UID	Unique IDentifier
VCD	Vicinity Coupling Device
VICC	Vicinity Integrated Circuit Card

3.3 Symbols

f_c Frequency of operating field (carrier frequency)

4 Definition of data elements

4.1 Unique identifier (UID)

The VICCs are uniquely identified by a 64 bits unique identifier (UID). This is used for addressing each VICC uniquely and individually, during the anticollision loop and for one-to-one exchange between a VCD and a VICC.

The UID shall be set permanently by the IC manufacturer in accordance with figure 1.

MSB LSB

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

Figure 1 — UID format

The UID comprises

- The 8 MSB bits shall be 0xE0,

- The IC manufacturer code, on 8 bits according to ISO/IEC 7816-6/AM1,
- A unique serial number on 48 bits assigned by the IC manufacturer.

4.2 Application family identifier (AFI)

AFI (Application family identifier) represents the type of application targeted by the VCD and is used to extract from all the VICCs present only the VICCs meeting the required application criteria.

It may be programmed and locked by the respective commands.

The most significant nibble of AFI is used to code one specific or all application families, as defined in table 1.

The least significant nibble of AFI is used to code one specific or all application sub-families. Sub-family codes different from 0 are proprietary.

AFI most significant nibble	**AFI Least significant nibble**	**Meaning** **VICCs respond from**	**Examples / note**
‘0’	‘0’	All families and sub-families	No applicative preselection
‘X’	'0'	All sub-families of family X	Wide applicative preselection
'X'	‘Y’	Only the Yth sub-family of family X	
‘0’	‘Y’	Proprietary sub-family Y only	
‘1'	‘0’, ‘Y’	Transport	Mass transit, Bus, Airline,...
'2'	‘0’, ‘Y’	Financial	IEP, Banking, Retail,...
'3'	‘0’, ‘Y’	Identification	Access control,...
'4'	‘0’, ‘Y’	Telecomunication	Public telephony, GSM,...
‘5’	‘0’, ‘Y’	Medical	
'6'	‘0’, ‘Y’	Multimedia	Internet services....
'7'	‘0’, ‘Y’	Gaming	
'8'	‘0’, ‘Y’	Data storage	Portable files, ...
'9'	‘0’, ‘Y’	Item management	
'A'	‘0’, ‘Y’	Express parcels	
'B'	‘0’, ‘Y’	Postal services	
'C'	‘0’, ‘Y’	Airline bags	
'D'	‘0’, ‘Y’		
'E'	‘0’, ‘Y’		
‘F’	‘0’, ‘Y’		

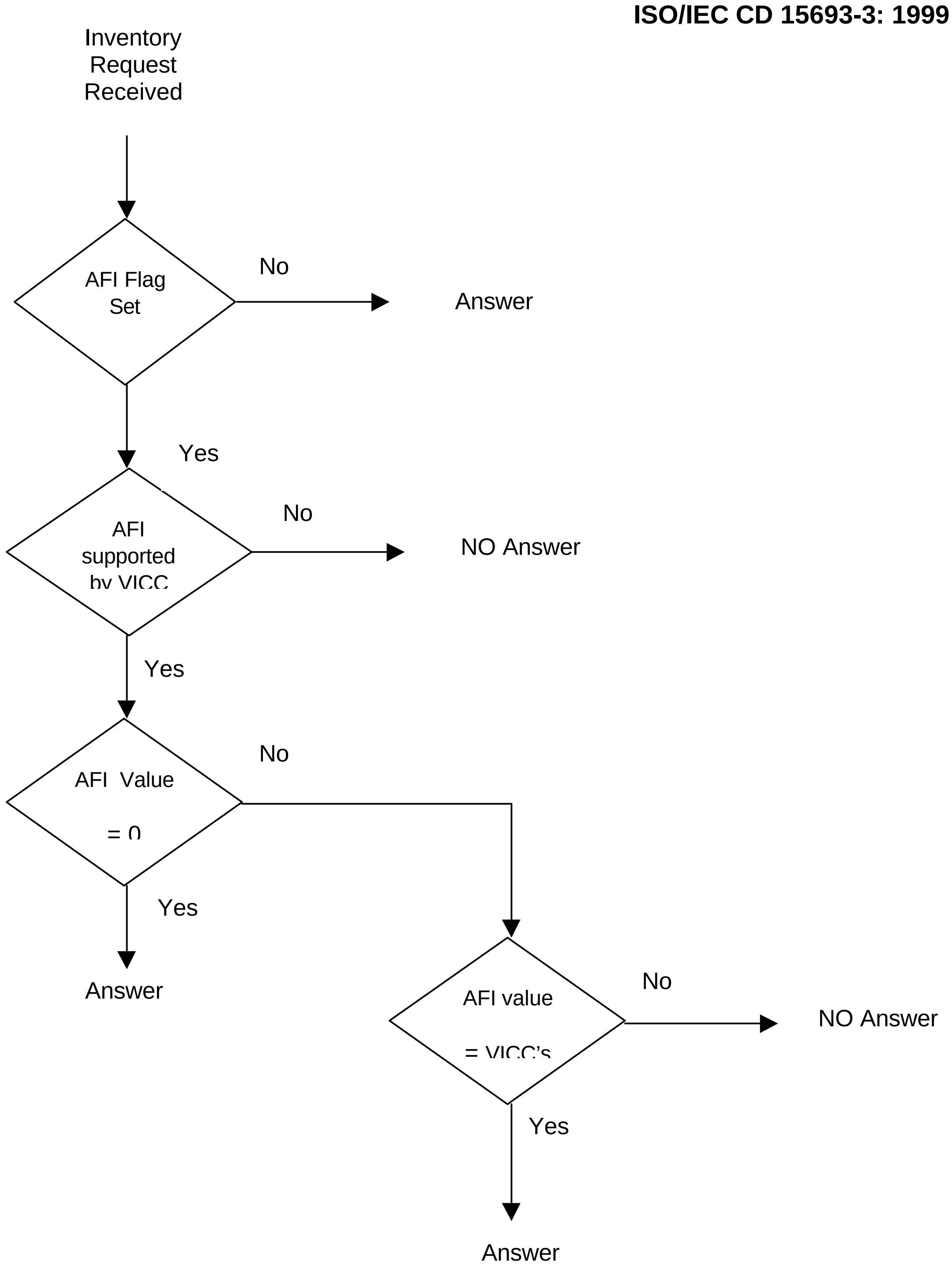
NOTE X = ‘1’ to ‘F’, Y = ‘1’ to ‘F’

Table 1 — AFI coding

The support of AFI by the VICC is optional.

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

If AFI is supported by the VICC, it shall answer according to the matching rules described in table 1.



Note: “Answer” means that the VICC shall answer to the Inventory request.

Figure 2 — VICC decision tree for AFI

4.3 Data storage format identifier (DSFID)

The Data storage format identifier indicates how the data is structured in the VICC memory.

It may be programmed and locked by the respective commands. It is coded on one byte. It allows for instant knowledge on the logical organization of the data.

If its programming is not supported by the VICC, the VICC shall respond with the value zero (0).

4.4 CRC

The CRC shall be calculated as per the definition in ISO/IEC 13239.

The initial register content shall be all ones : "FFFF".

The two bytes CRC are appended to each request and each response, within each frame, before the EOF. The CRC is calculated on all the bytes after the SOF up to but not including the CRC field.

Upon reception of a request from the VCD, the VICC shall verify that the CRC value is valid. If it is invalid, it shall discard the frame and shall not answer (modulate).

Upon reception of a response from the VICC, it is recommended that the VCD verify that the CRC value is valid. If it is invalid, actions to be performed are left to the responsibility of the VCD designer.

The CRC is transmitted least significant byte first.

Each byte is transmitted least significant bit first.

LSByte		MSByte	
LSBit	MSBit	LSBit	MSBit
CRC 16 (8bits)		CRC 16 (8 bits)	

Figure 3 — CRC bits and bytes transmission rules

5 VICC memory organization

The commands specified in this standard assume that the physical memory is organized in blocks (or pages) of fixed size.

- Up to 256 blocks can be addressed.
- Block size can be of up to 256 bits.
- This leads to a maximum memory capacity of up to 8 KBytes (64 KBits).

Note: The structure allows for future extension of the maximum memory capacity.

The commands described in this standard allow the access (read and write) by block(s). There is no implicit or explicit restriction regarding other access method (e.g. by byte or by logical object in future revision(s) of the standard or in custom commands.

6 Block security status

The block security status is coded on one byte. It is an element of the protocol. There is no implicit or explicit assumption that the 8 bits are actually implemented in the physical memory structure of the VICC.

When it is present in a request or a response, the VICC shall interpret (request) or set (response) the supported bits according to the description below:

Table 2 — Block security status

Bit Nb	Flag name	State	Description
Bit 1	Locked	0	Not locked
		1	Locked
Bit 2 to 8	RFU		Shall be set to 0

7 Overall protocol description

Protocol concept

The transmission protocol (or protocol) defines the mechanism to exchange instructions and data between the VCD and the VICC, in both directions.

It is based on the concept of "VCD talks first".

This means that any VICC does not start transmitting (i.e. modulating according to 15693-2) unless it has received and properly decoded an instruction sent by the VCD.

- The protocol is based on an exchange of
  - a request from the VCD to the VICC
  - a response from the VICC(s) to the VCD
- Each request and each response are contained in a frame. The frame delimiters (SOF, EOF) are specified in ISO/IEC 15693-2.
- Each request consists of the following fields:
  - Flags
  - Command code
  - Mandatory and optional parameters fields, depending on the command
  - Application data fields
  - CRC
- Each response consists of the following fields:
  - Flags
  - Mandatory and optional parameters fields, depending on the command

- Application data fields
- CRC

- The protocol is bit-oriented. The number of bits transmitted in a frame is a multiple of eight (8), i.e. an integer number of bytes.
- A single-byte field is transmitted least significant bit (LSBit) first.
- A multiple-byte field is transmitted least significant byte (LSByte) first, each byte is transmitted least significant bit (LSBit) first.
- The setting of the flags indicates the presence of the optional fields. When the flag is set (to one), the field is present. When the flag is reset (to zero), the field is absent.
- RFU flags shall be set to zero (0).

7.1 Modes

The term mode refers to the mechanism to specify in a request the set of VICC's that shall answer to the request.

7.1.1 Addressed mode

When the Address_flag is set to 1 (addressed mode), the request shall contain the unique ID (UID) of the addressed VICC.

Any VICC receiving a request with the Address_flag set to 1 shall compare the received unique ID (address) to its own ID.

It it matches, it shall execute it (if possible) and return a response to the VCD as specified by the command description.

If it does not match, it shall remain silent.

7.1.2 Non-addressed mode

When the Address_flag is set to 0 (non-addressed mode), the request shall not contain a unique ID.

Any VICC receiving a request with the Address_flag set to 0 shall execute it (if possible) and shall return a response to the VCD as specified by the command description. 1)

7.1.3 Select mode

When the Select_flag is set to 1 (select mode), the request shall not contain a VICC unique ID.

The VICC in the selected state receiving a request with the Select_flag set to 1 shall execute it (if possible) and shall return a response to the VCD as specified by the command description.

Only the VICC in the selected state shall answer to a request having the select flag set to 1.

---

1) Sub-carrier_flag refers to the VICC-to-VCD communication as specified in ISO/IEC 15693-2.

7.2 Request format

The request consists of the following fields:

- Flags
- Command code (see clause 9)
- Parameters and data fields
- CRC (see clause 4.4)

SOF	Flags	Command code	Parameters	Data	CRC	EOF

Figure 4 — General request format

7.2.1 Request flags

In a request, the field "flags" specifies the actions to be performed by the VICC and whether corresponding fields are present or not.

It consists of eight bits.

Table 3 — Request flags 1 to 4 definition

Bit Nb	Flag name	State	Description
Bit 1	Sub-carrier_flag	0	A single sub-carrier frequency shall be used by the VICC
		1	Two sub-carriers shall be used by the VICC
Bit 2	Data_rate_flag	0	Low data rate shall be used
		1	High data rate shall be used
Bit 3	Inventory_flag	0	Flags 5 to 8 meaning is according to table 4
		1	Flags 5 to 8 meaning is according to table 5
Bit 4	Protocol Extension_flag	0	No protocol format extension
		1	Protocol format is extended. Reserved for future use.

Note: 1. Sub-carrier_flag refers to the VICC-to-VCD communication as specified in ISO/IEC 15693-2.

2. Data_rate_flag refers to the VICC-to-VCD communication as specified in ISO/IEC 15693-2.

Table 4 — Request flags 5 to 8 definition when inventory flag is NOT set

Bit Nb	Flag name	State	Description
Bit 5	Select_flag	0	Request shall be executed by any VICC according to the setting of Address_flag
		1	Request shall be executed only by VICC in selected state
Bit 6	Address_flag	0	Request is not addressed. UID field is not present. It shall be executed by any VICC.
		1	Request is addressed. UID field is present. It shall be executed only by the VICC whose UID matches the UID specified in the request.
Bit 7	Custom_flag	0	Meaning is defined by the Custom command
		1	Meaning is defined by the Custom command
Bit 8	RFU	0	
		1	

Note: if the Select_flag is set to 1, the Address_flag shall be set to 0 and the UID field shall not be present in the request.

Table 5 — Request flags 5 to 8 definition when inventory flag is set

Bit Nb	Flag name	State	Description
Bit 5	AFI flag	0	AFI field is not present
		1	AFI field is present
Bit 6	Nb_slots_flag	0	16 slots
		1	1 slot
Bit 7	Custom_flag	0	Meaning is defined by the Custom command
		1	Meaning is defined by the Custom command
Bit 8	RFU	0	
		1	

7.3 Response format

The response consists of the following fields:

- Flags
- one or more parameter fields
- Data
- CRC (see clause 4.4)

SOF	Flags	Parameters	Data	CRC	EOF

Figure 5 — General response format

7.3.1 Response flags

In a response, it indicates how actions have been performed by the VICC and whether corresponding fields are present or not.

It consists of eight bits.

Table 6 — Response flags 1 to 8 definition

Bit Nb	Flag name	State	Description
Bit 1	Error_flag	0	No error
		1	Error detected. Error code is in the "Error" field.
Bit 2	RFU		
Bit 3	RFU		
Bit 4	Extension_flag	0	No protocol format extension.
		1	Protocol format is extended. Reserved for future use.
Bit 5	RFU		
Bit 6	RFU		
Bit 7	RFU		
Bit 8	RFU		

7.3.2 Response error code

If the error flag is set by the VICC in the response, the error code field is present and provides information about the error that occurred.

The following error codes are specified. Other codes are reserved for future use.

Table 7 — Response error code definition

Error code	Meaning
0x01	The command is not supported, i.e. the request code is not recognized.
0x02	The command is not recognized, for example: a format error occurred.
0x0F	Unknown error.
0x10	The specified block is not available (doesn’t exist).
0x11	The specified block is already -locked and thus cannot be locked again
0x12	The specified block is locked and its content cannot be changed.
0x13	The specified block was not successfully programmed.
0x14	The specified block was not successfully locked.
0xA0 - 0xDF	Custom command error codes
all others	RFU

Note: If the VICC does not support error codes listed in table 7, it shall answer with the error code 0x0F (unknown error).

7.4 VICC states

A VICC can be in 4 states:

- Power-off
- Ready
- Quiet
- Selected

The transition between these states is specified in figure 4.

The support of power-off, ready and quiet states is mandatory.

The support of selected state is optional.

7.4.1 Power-off state

The VICC is in the power-off state when it cannot be activated by the VCD.

7.4.2 Ready state

The VICC is in the Ready state when it is activated by the VCD. It shall process any request where the select flag is not set.

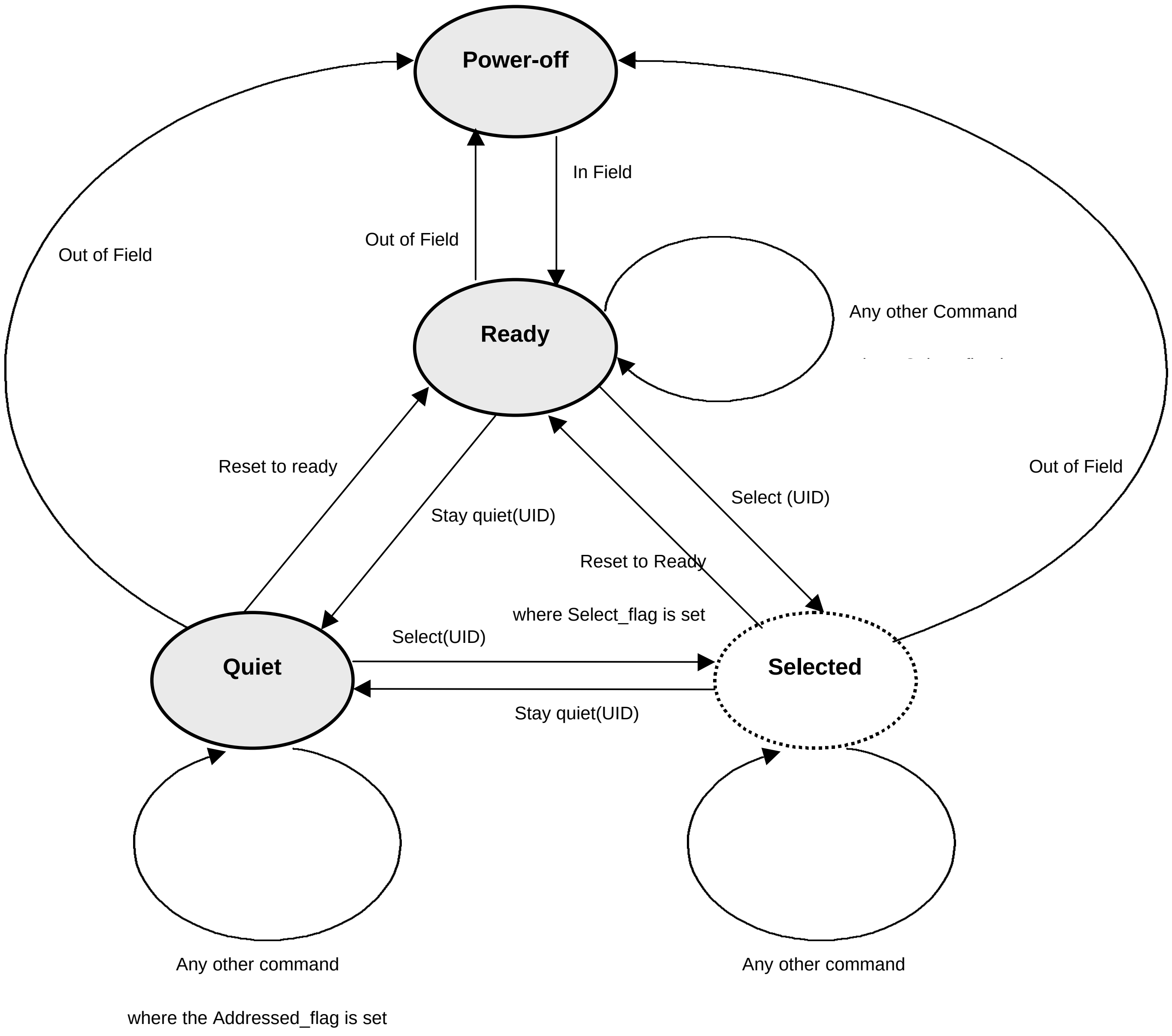
7.4.3 Quiet state

When in the quiet state, the VICC shall process any request where the Inventory_flag is not set AND where the Addressed_flag is set.

7.4.4 Selected state

When in the selected state, the VICC shall answer only to request having the select flag set.

Figure 5: VICC state transition diagram



Note: The intention of the state transition method is that only one VICC should be in the selected state at a time.

8 Anticollision

The purpose of the anticollision sequence is to make an inventory of the VICCs present in the VCD field by their unique ID (UID).

The VCD is the master of the communication with one or multiple VICCs. It initiates card communication by issuing the inventory request.

The VICC shall send its response in the slot determined or shall not respond, according to the algorithm described in clause 8.2.

8.1 Request parameters

When issuing the inventory command, the VCD shall:

- set the Nb_slots_flag to the desired setting,
- add after the command field the mask length and the mask value.

SOF	Flags	Command	Mask length	Mask value	EOF
	8 bits	8 bits	8 bits	0 to 8 bytes	

Figure 6 — Inventory request format

- The mask length is the number of significant bits of the mask value.
- The mask value is contained in an integer number of bytes. The mask length indicates the number of significant bits. LSB shall be transmitted first.
- If the mask length is not a multiple of 8 (bits), the mask value MSB shall be padded with the required number of null (set to 0) bits so that the mask value is contained in an integer number of bytes.
- The next field starts on the next byte boundary.

MSB	LSB
0000	0100 1100 1111
Pad	Value

Figure 7 — Example of the padding of the mask

In the example of the figure 6, the mask length is 12 bits. The mask value MSB is padded with four bits set to 0

The AFI field shall be present if the AFI_flag is set.

The pulse shall be generated according to the definition of the EOF in ISO/IEC 15693-2.

The first slot starts immediately after the reception of the request EOF.

To switch to the next slot, the VCD sends an EOF.

The following rules and restrictions apply:

if no VICC answer is detected, the VCD may switch to the next slot by sending an EOF,

if one or more VICC answers are received, the VCD shall wait until the VICC frames are completely received before sending an EOF for switching to the next slot.

This is illustrated in figure 7 and the timing is described in table 8.

8.2 Request processing by the VICC

Upon reception of a valid request, the VICC shall perform the following algorithm or its functional equivalent (implementation dependent).

NbS is the total number of slots (1 or 16)

SN is the current slot number (0 to 15)

LSB (value, n) function returns the n less significant bits of value

"&" is the concatenation operator

Slot_Frame is either a SOF or an EOF

```
    SN= 0
    if Nb_slots_flag    then NbS =1  SN_length=0
                        else    NbS = 16    SN_length=4
    endif
label1:  if LSB(UID, SN_length + Mask_length) = LSB(SN, SN_length)&LSB(Mask, Mask_length) then
        transmit response to inventory request
    endif
```

```
wait (Slot_Frame)
if Slot_Frame= SOF then
    Stop anticollision and decode/process request
    exit
endif
if SN<NbS-1 then
    SN = SN +1
    goto label1
    exit
endif
exit
```

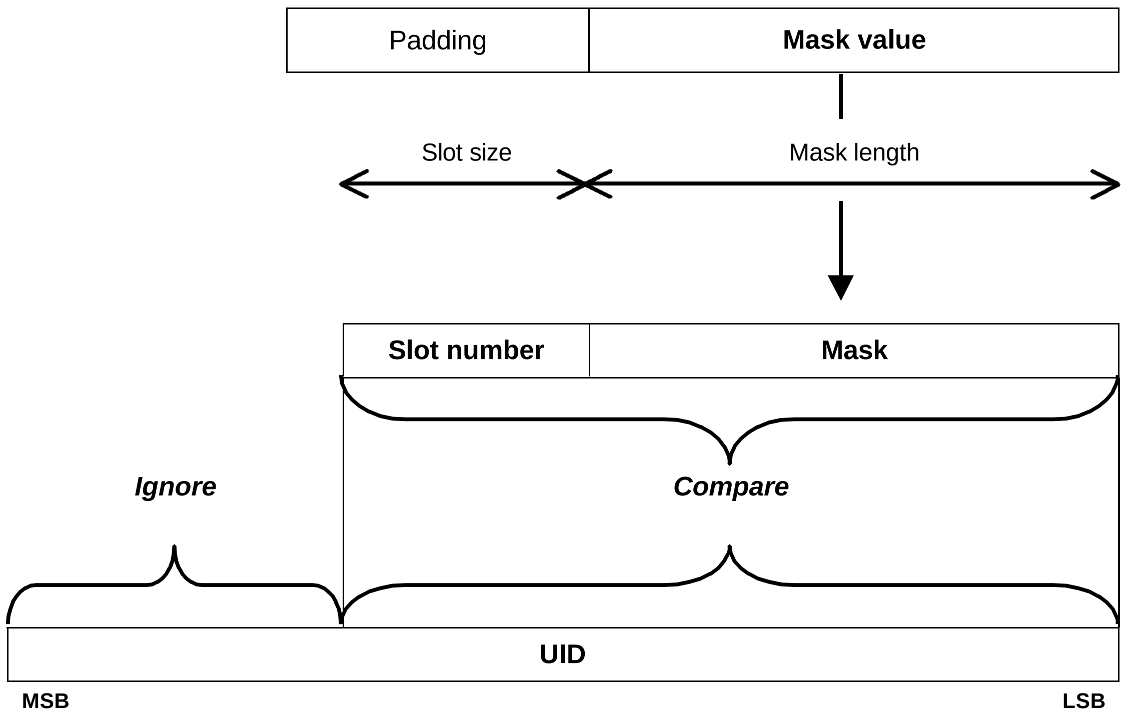


Figure 8 — Principle of comparison between the mask, slot number and UID

8.3 Explanation of an anticollision sequence

Figure 8 summarizes the main variations that can occur during an anticollision sequence for slot number of 16.

The different steps are:

- The VCD sends an inventory request, in a frame, terminated by a EOF. The number of slots is 16.

- VICC 1 transmits its response in slot 0. It is the only one to do so, therefore no collision occurs and its UID is received and registered by the VCD;

- The VCD sends an EOF, meaning to switch to the next slot.

- In slot 1, two VICCs 2 and 3 transmits their response, this generates a collision. The VCD detects it and remembers that a collision was detected in slot 1.

- The VCD sends an EOF, meaning to switch to the next slot.

- In slot 2, no VICC transmits a response. Therefore the VCD does not detect a VICC SOF and decides to switch to the next slot by sending a EOF.

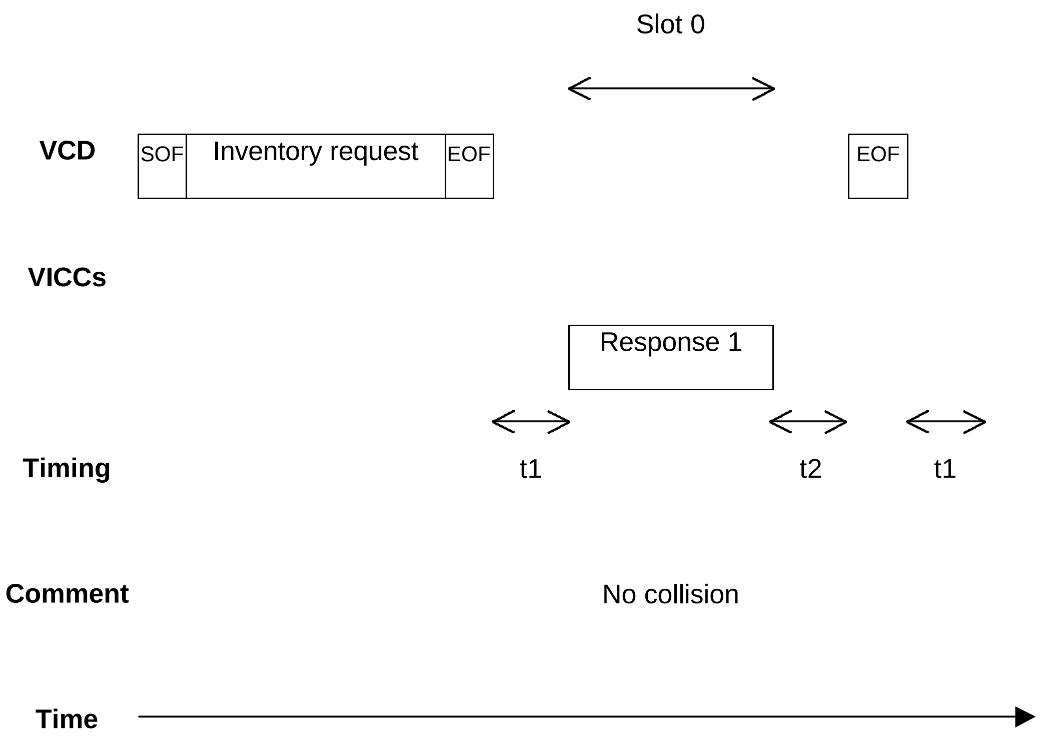
- In slot 3, there is another collision caused by responses from VICC 4 and 5

- The VCD then decides to send a request (for instance a Read Block) to VICC 1, which UID was already correctly received.

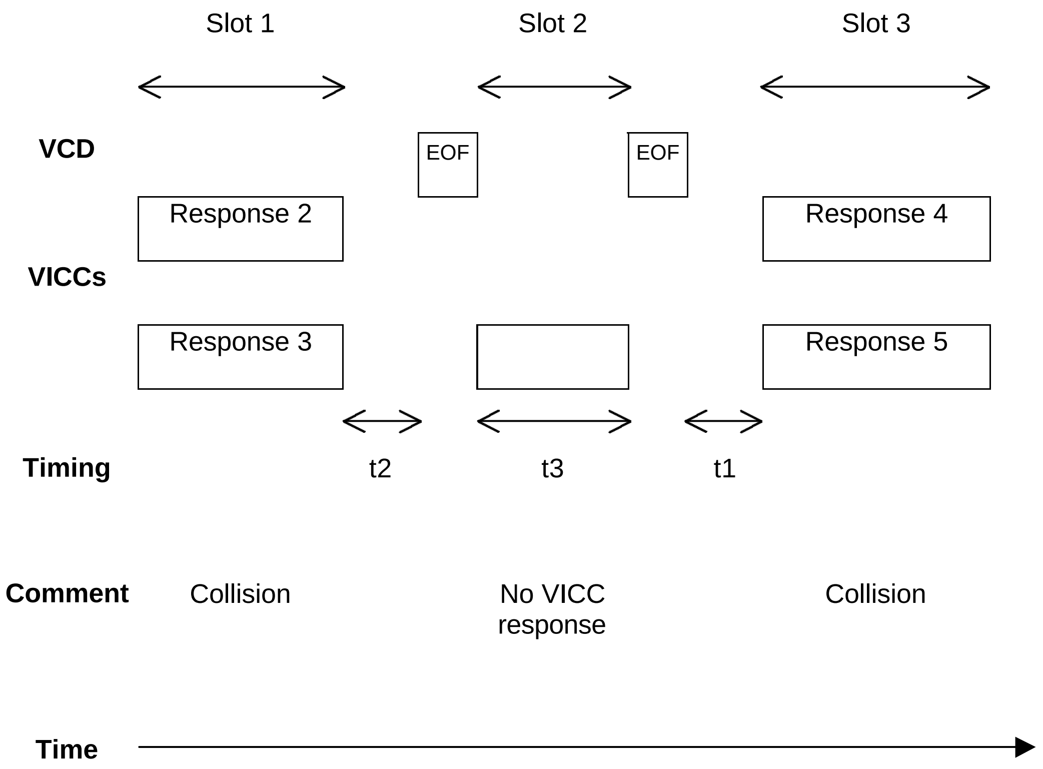
- All VICCs detect a SOF and exit the anticollision sequence. They process this request and since the request is addressed to VICC 1, only VICC1 transmit its response.

- All VICCs are ready to receive another request. If it is an inventory command, the slot numbering sequence restarts from 0.

Note: the decision to interrupt the anticollision sequence is up to the VCD. It could have continued to send EOF's till slot 15 and then send the request to VICC 1.


Slot 0
VCD
SOF
Inventory request
EOF
EOF
VICCs
Response 1
Timing
t1
t2
t1
Comment
No collision
Time

Continued…


Slot 1
Slot 2
Slot 3
VCD
EOF
EOF
Response 2
Response 4
VICCs
Response 3
Response 5
Timing
t2
t3
t1
Comment
Collision
No VICC response
Collision
Time

Continued…

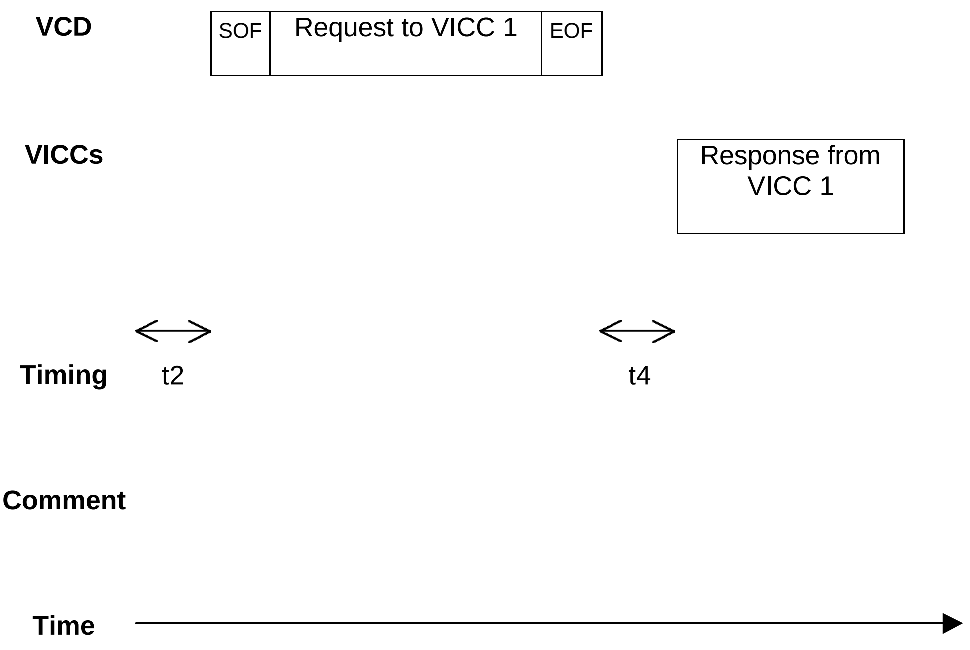


Figure 9 — Description of a possible anticollision sequence

8.4 Timing definition

- t1 is the time after which the VICC shall start transmitting its response. It starts from the raising edge of the EOF received from the VCD.
- t2 is the time after which the VCD shall send an EOF to switch to the next slot when one or more VICC responses have been received. It starts from the reception of the EOF received from the VICCs.
- t3 is the time after which the VCD shall send an EOF to switch to the next slot when no VICC response has been received. It starts from the raising edge of the EOF previously sent by the VCD.
- t4 is the time after which the VICC shall start transmitting its response to a VCD request. It starts from the raising edge of the EOF received from the VCD.

All timings shall be a multiple of $4096/f_c$, with a tolerance of $\pm 32/f_c$.

Table 8 — Timing for high VICC-to-VCD data rate

	minimum	typical	maximum
t1	4064/f_c (300 µs)	4096/f_c (302 µs)	4128/f_c (304 µs)
t2	4064/f_c (300 µs)	-	-
t3	8160/f_c (602 µs)	-	-
t4	4064/f_c (300µs)	4096/f_c (302 µs) for read alike requests 10 ms for write alike requests	4128/f_c (304 µs) for read alike requests 20 ms for write alike requests

Table 9 — Timing for low VICC-to-VCD data rate

	minimum	typical	Maximum
t1	16256/f_c (1200 µs)	16384/f_c (1208 µs)	16512/f_c (1216 µs)
t2	16256/f_c (1200 µs)	-	-
t3	20332/f_c (1502 µs)	-	-
t4	16256/f_c (1200µs)	16384/f_c (1208 µs) for read alike requests 10 ms for write alike requests	16512/f_c (1216 µs) for read alike requests 20 ms for write alike requests

9 Commands

Four sets of commands are defined:

- **Mandatory** 0x01 to 0x1F all VICCs shall support them

- **Optional** 0x20 to 0x9F VICCs may support them, at their option. If supported, request and response formats shall comply with the definition given in this standard.

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

- **Custom** 0xA0 to 0xDF VICCs support them, at their option, to implement manufacturer specific functions. The function of flags (including reserved bits) shall not be modified except the custom flag. The only fields that can be customized are the parameters and the data fields.

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

  Any custom command contains as its first parameter the IC manufacturer code. This allows IC manufacturers to implement custom commands without risking duplication of command codes and thus misinterpretation.

- **Proprietary** 0xE0 to 0xFF these commands are used by IC and VICC manufacturers for various purposes such as tests, programming of system information, etc... They are not specified in this standard. The IC manufacturer may at its option document them or not. It is allowed that these commands are disabled after IC and/or VICC manufacturing.

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

9.1 Command codes

Table 10 — Command codes

Command code	Type	Function
0x01	Mandatory	Inventory
0x02	Mandatory	Stay quiet
0x03 – 0x1F	Mandatory	RFU
0x20	Optional	Read single block
0x21	Optional	Write single block
0x22	Optional	Lock block
0x23	Optional	Read multiple blocks
0x24	Optional	Write multiple blocks
0x25	Optional	Select
0x26	Optional	Reset to ready
0x27	Optional	Write AFI
0x28	Optional	Lock AFI
0x29	Optional	Write DSFID
0x2A	Optional	Lock DSFID
0x2B	Optional	Get system information
0x2C	Optional	Get multiple block security status
0x2D – 0x9F	Optional	RFU
0xA0 – 0xDF	Custom	IC Mfg dependent
0xE0 – 0xFF	Proprietary	IC Mfg dependent

9.2 Mandatory commands

9.2.1 Inventory

Command code = 0x01

When receiving the Inventory request, the VICC shall perform the anticollision sequence.

The request contains:

- The flags,
- The Inventory command code
- The AFI if the AFI flag is set
- The mask length

- The mask value
- The CRC

The Inventory_flag shall be set to 1.

The meaning of flags 5 to 8 is according to table 4.

SOF	Flags	Inventory	Optional AFI	Mask length	Mask value	CRC16	EOF
	8 bits	8 bits	8 bits	8 bits	0 – 64 bits	16 bits	

Figure 10 — Inventory request format

The response shall contain:

- the DSFID
- the unique ID

SOF	Flags	DSFID	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 11 — Inventory response format

9.2.2 Stay quiet

Command code = 0x02

When receiving the Stay quiet command, the VICC shall enter the quiet state and shall NOT send back a response. There is NO response to the Stay quiet command.

When in quiet state:

- the VICC shall not process any request which Inventory_flag is set,
- the VICC shall process any addressed request

The VICC shall exit the quiet state when:

- reset (power off),
- receiving a Select request. It shall then go to the selected state if supported or return an error if not supported,
- receiving a Reset to ready request. It shall then go to the Ready state.

SOF	Flags	Stay quiet	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 12 — Stay quiet request format

The Stay quiet command shall always be executed in Addressed mode.

9.3 Optional commands

9.3.1 Read single block

Command code = 0x20

When receiving the Read single block command, the VICC shall read the requested block and send back its value in the response.

SOF	Flags	Read single block	UID	Block number	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	16 bits	

Figure 13 — Read single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 14 — Read single block response format when error flag is set

SOF	Flags	Data	CRC16	EOF
	8 bits	Block length	16 bits	

Figure 15 — Read single block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

if error flag is not set

Block data

9.3.2 Write single block

Command code = 0x21

When receiving the Write single block command, the VICC shall write the requested block with the data contained in the request and report the success of the operation in the response.

SOF	Flags	Write single block	UID	Block number	Data	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	Block length	16bits	

Figure 16 — Write single block request format

Request parameter:

(Optional) UID

Block number

Data

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 17 — Write single block response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 18 — Write single block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

9.3.3 Lock block

Command code = 0x22

When receiving the Lock block command, the VICC shall lock permanently the requested block.

SOF	Flags	Lock block	UID	Block number	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	16 bits	

Figure 19 — Lock single block request format

Request parameter:

(Optional) UID

Block number

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 20 — Lock block response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 21 — Lock block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

9.3.4 Read multiple blocks

Command code = 0x23

When receiving the Read multiple block command, the VICC shall read the requested block(s) and send back their value in the response.

SOF	Flags	Read multiple block	UID	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	8 bits	16 bits	

Figure 22 — Read multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 23 — Read multiple blocks response format when error flag is set

SOF	Flags	Data	CRC16	EOF
	8 bits	Block length	16 bits	
		Repeated as needed		

Figure 24 — Read multiple block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

if error flag is not set

Block data (repeated as per figure 23)

9.3.5 Write multiple blocks

Command code = 0x24

When receiving the Write multiple single block command, the VICC shall write the requested block(s) with the data contained in the request and report the success of the operation in the response.

SOF	Flags	Write multiple block	UID	First block number	Number of blocks	Data	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	8 bits	Block length	16 bits	
						Repeated as needed		

Figure 25 — Write multiple blocks request format

Request parameter:

(Optional) UID

First block number

Number of blocks

Block data (repeated as per figure 24)

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 26 — Write multiple blocks response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 27 — Write multiple block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

9.3.6 Select

Command code = 0x25

When receiving the Select command:

- if the UID is equal to its own UID, the VICC shall enter the selected state and shall send a response.
- if it is different, the VICC shall return to the Ready state and shall not send a response.

SOF	Flags	Select	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 28 — Select request format

Request parameter:

UID (mandatory)

SOF	Flags	Error Code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 29 — Select response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 30 — Select block response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

9.3.7 Reset to ready

Command code = 0x26

When receiving a Reset to ready command, the VICC shall return to the Ready state.

SOF	Flags	Reset to ready	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 31 — Reset to ready request format

Request parameter:

(Optional) UID

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 32 — Reset to ready response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 33 — Reset to ready response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

9.3.8 Write AFI

Command code = 0x27

When receiving the Write AFI request, the VICC shall write the AFI value into its memory.

SOF	Flags	Write AFI	UID	AFI	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	16 bits	

Figure 34: Write AFI request format

Request parameter:

(Optional) UID

AFI

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 35 — Write AFI response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 36 — Write AFI response format when error flag is NOT set

9.3.9 Lock AFI

Command code = 0x28

When receiving the Lock AFI request, the VICC shall lock the AFI value permanently into its memory.

SOF	Flags	Lock AFI	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 37 — Lock AFI request format

SOF	Flags	Error Code	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 38 — Lock AFI response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 39 — Lock AFI response format when error flag is NOT set

9.3.10 Write DSFID command

Command code = 0x29

When receiving the Write DSFID request, the VICC shall write the DSFID value into its memory.

SOF	Flags	Write DSFID	UID	DSFID	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	16 bits	

Figure 40 — Write DSFID request format

Request parameter:

(Optional) UID

DSFID

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 41 — Write DSFID response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 42 — Write DSFID response format when error flag is NOT set

9.3.11 Lock DSFID

Command code = 0x2A

When receiving the Lock DSFID request, the VICC shall lock the DSFID value permanently into its memory.

SOF	Flags	Lock DSFID	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 43 — Lock DSFID request format

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 44 — Lock DSFID response format when error flag is set

SOF	Flags	CRC16	EOF
	8 bits	16 bits	

Figure 45 — Lock DSFID response format when error flag is NOT set

9.3.12 Get system information

Command code = 0x2B

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

SOF	Flags	Get system info	UID	CRC16	EOF
	8 bits	8 bits	64 bits	16 bits	

Figure 46 — Get system information request format

Request parameter:

(Optional) UID

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 47 — Get system information response when error flag is set

SOF	Flags	Info flags	UID	DSFID	AFI	Other fields	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	8 bits	See below	16 bits	

Figure 48 — Get system information response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

if error flag is not set

Information flag

UID (mandatory)

Information fields, in the order of their corresponding flag, as defined in figure 47, if their corresponding flag is set.

Table 11 — Information flags definition

Bit Nb	Flag name	State	Description
Bit 1	DSFID	0	DSFID is not supported. DSFID field is not present
		1	DSFID is supported. DSFID field is present
Bit 2	AFI	0	AFI is not supported. AFI field is not present
		1	AFI is supported. AFI field is present
Bit 3	VICC memory size	0	Information on VICC memory size is not supported. Memory size field is present.
		1	Information on VICC memory size is supported. Memory size field is present.
Bit 4	IC reference	0	Information on IC reference is not supported. IC reference field is not present.
		1	Information on IC reference is supported. IC reference field is present.
Bit 5	RFU		
Bit 6	RFU		
Bit 7	RFU		
Bit 8	RFU		

Table 12 — VICC memory size information

MSB		LSB
16 … 14	13 … 9	8 … 1
RFU	Block size in bytes	Number of blocks

Block size is expressed in number of bytes, on 5 bits, allowing to specify up to 32 bytes i.e. 256 bits.

Number of blocks is on 8 bits, allowing to specify up to 256 blocks.

The three most significant bits are reserved for future use and shall be set to zero.

The IC reference is on 8 bits and its meaning is defined by the IC manufacturer.

9.3.13 Get multiple block security status

Command code = 0x2C

When receiving the Get multiple block security status command, the VICC shall send back the block security status.

SOF	Flags	Get multiple block security status	UID	First block number	Number of blocks	CRC16	EOF
	8 bits	8 bits	64 bits	8 bits	8 bits	16 bits	

Figure 49 — Get multiple block security status request format

Request parameter:

(Optional) UID

First block number

Number of blocks

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 50 — Get multiple block security status response format when error flag is set

SOF	Flags	Block security status	CRC16	EOF
	8 bits	8 bits Repeated as needed	16 bits	

Figure 51 — Get multiple block security status response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

if error flag is not set

Block security status (repeated as per figure 50)

9.4 Custom commands

The format of custom command is generic and allows unambiguous attribution of custom command codes to each VICC manufacturer.

The custom command code is the combination of a custom command code and of the VICC manufacturer code.

The custom request parameters definition is the responsibility of the VICC manufacturer.

SOF	Flags	Custom	IC Mfg **c**ode	Custom request parameters	CRC16	EOF
	8 bits	8 bits	8 bits	Custom defined	16 bits	

Figure 52 — Custom request format

Request parameter:

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

SOF	Flags	Error **c**ode	CRC16	EOF
	8 bits	8 bits	16 bits	

Figure 53 — Custom response format when error flag is set

SOF	Flags	Custom response parameters	CRC16	EOF
	8 bits	Custom defined	16 bits	

Figure 54 — Custom response format when error flag is NOT set

Response parameter:

Error flag (and code if error flag is set)

if error flag is not set

Custom parameters

9.5 Proprietary commands

The format of these commands is not defined in this standard

Annex A
(informative)

Compatibility with other card standards

This standard does not preclude the addition of other existing card standards on the VICC, such as ISO/IEC 7816 or others listed in annex B.

Annex B
(informative)

Bibliography of other ISO/IEC card standards.

ISO/IEC 7811, *Identification cards - Recording technique -*

ISO/IEC 7812-1:1993, *Identification cards - Identification of issuers - Part 1: Numbering system.*

ISO/IEC 7812-2:1993, *Identification cards - Identification of issuers - Part 2: Application and registration procedures.*

ISO/IEC 7813:1995, *Identification cards - Financial transaction cards.*

ISO/IEC 7816-1:1997, *Identification cards - Integrated circuit(s) cards with contacts - Part 1: Physical characteristics.*

ISO/IEC 7816-2:1998, *Identification cards - Integrated circuit(s) cards with contacts - Part 2: Dimensions and location of the contacts.*

Annex C
(informative)

VCD pseudo-code for anticollision

The following pseudo-code describes how the anticollision could be implemented on the VCD, using recursivity.

It does not describe the collision detection mechanism.

Algorithm for 16 slots

```
function push (mask, address)   ; pushes on private stack

function pop (mask, address)    ; pops from private stack

function pulse_next_pause    ; generates a power pulse

function store(VICC_UID)     ; stores VICC_UID

function poll_loop (sub_address_size as integer)

; address length must be four (4) bits.

pop (mask, address)

mask = address & mask        ; generates new mask

; send the Request

mode = anticollision

send_Request(Request_cmd, mode, mask length, mask value)

for address = 0 to (2^sub_address_size - 1)



if no_collision_is_detected then   ; VICC is inventoried

     store (VICC_UID)

  else            ; remember a collision was detected

     push(mask,address)

  endif

pulse_next_pause

next sub_address

; if some collisions have been detected and not yet processed,
; the function calls itself recursively to process the last
; stored collision
if stack_not_empty then poll_loop (sub_address_size)

end poll_loop
main_cycle:
mask = null
address = null
push (mask, address)
poll_loop(sub_address_size)
end_main_cycle
```

Annex D
(informative)

Cyclic Redundancy Check (CRC)

D.1 The CRC error detection method

The Cyclic Redundancy Check (CRC) is calculated on all data contained in a message, from the start of the flags through to the end of data. This CRC is used from VCD to VICC and from VICC to VCD.

CRC Definition					
CRC type	**Length**	**Polynomial**	**Direction**	**Preset**	**Residue**
ISO/IEC 13239	16 bits	$X^{16} + X^{12} + X^5 + 1$ = Ox8408	Backward	0xFFFF	0xF0B8

Table D.1 — CRC Definition

To add extra protection against shifting errors, a further transformation on the calculated CRC is made. The One's complement of the calculated CRC is the value attached to the message for transmission. This transformation is included in the example below.

For checking of received messages the 2 CRC bytes are often also included in the re-calculation, for ease of use.

In this case, given the expected value for the generated CRC is the residue of 0xF0B8.

D.2 CRC calculation example

This example in C language illustrates one method of calculating the CRC on a given set of bytes comprising a message.

```
#define  POLYNOMIAL     0x8408          //  x^16 + x^12 + x^5 + 1

#define  PRESET_VALUE   0xFFFF

#define  CHECK_VALUE    0xF0B8

#define  NUMBER_OF_BYTES     4         // Example: 4 data bytes

#define  CALC_CRC            1

#define  CHECK_CRC           0

void main()

{

  unsigned int  current_crc_value;

  unsigned char array_of_databytes[NUMBER_OF_BYTES + 2] = {1, 2, 3, 4, 0x91, 0x39};
```

```
int           number_of_databytes = NUMBER_OF_BYTES;

int           calculate_or_check_crc;

int           i, j;

calculate_or_check_crc = CALC_CRC;

//  calculate_or_check_crc = CHECK_CRC;  // This could be an other example

if (calculate_or_check_crc == CALC_CRC)

{

    number_of_databytes = NUMBER_OF_BYTES;

}

else    // check CRC

{

    number_of_databytes = NUMBER_OF_BYTES + 2;

}

current_crc_value = PRESET_VALUE;

for (i = 0; i < number_of_databytes; i++)

{

    current_crc_value = current_crc_value ^ ((unsigned int)array_of_databytes[i]);

    for (j = 0; j < 8; j++)

    {

        if (current_crc_value & 0x0001)

        {

            current_crc_value = (current_crc_value >> 1) ^ POLYNOMIAL;

        }

        else

        {

            current_crc_value = (current_crc_value >> 1);

        }

    }

}

if (calculate_or_check_crc == CALC_CRC)
```

```
  {
      current_crc_value = ~current_crc_value;
      printf ("Generated CRC is 0x%04X\n", current_crc_value);
      // current_crc_value is now ready to be appended to the data stream
      // (first LSByte, then MSByte)
  }
  else   // check CRC
  {
      if (current_crc_value == CHECK_VALUE)
      {
          printf ("Checked CRC is ok (0x%04X)\n", current_crc_value);
      }
      else
      {
          printf ("Checked CRC is NOT ok (0x%04X)\n", current_crc_value);
      }
  }
}
```

Figure D.1 — C-example to calculate or check the CRC16 according to ISO/IEC 13239