Reference Manual

STARCOS S 1.2

Edition 08/96
Order No. 146579031

Giesecke & Devrient GmbH
Prinzregentenstr. 159
Postfach 80 07 29
D-81607 München
Deutschland
Tel. +49 89 4119-0
Fax +49 89 4119-535

Giesecke & Devrient GmbH
Prinzregentenstr. 159
Postfach 80 07 29
81607 München
Germany

Contents

About the Product

Characteristics

STARCOS® S (Smart Card Chip Operating System/Standard Version) developed by G&D constitutes a complete operating system for smartcards. Great emphasis has been put on compliance with both existing and forthcoming ISO standards.
Smartcard operating systems control the data transfer, the storage areas and process information; they administrate the resources and supply all necessary functions for operation and administration of a random number of applications.

The main features of STARCOS S include:

- the support for several applications in the card, which may be installed independently of each other (multi-functionality)
- the implementation of several hierarchical file structures (file organisation)
- the implementation of various access controls (authentication)

The number of loadable applications is only limited by the amount of EEPROM memory available. The registration, creation and loading of data for an application can be done independently with defined security levels. The application designer is responsible for the definition of the security level and structure of his own application.

About the Manual

The present 'Reference Manual STARCOS S' should serve the G&D customers both as introduction and reference manual for the operating system STARCOS S.

Structure

The information contained is structured as follows:

- Files – Structures and Definitions
  describes both the file hierarchy and the file attributes used by STARCOS S.
- Functions – Sequence and Interaction
  describes the sequence and type of STARCOS S functions performed during command processing.
- Commands
  describes the STARCOS S commands and their responses in alphabetical order.
- Completion
  describes the completion procedure and the respective commands.
- Appendix
  contains a description of the transmission protocols T=1 and T=0, a list of the figures, a glossary of terms and abbreviations used as well as an index.

Target Group

This reference manual addresses developers and specialists for smartcard applications. The user should be familiar both with smartcard hardware/software and the appropriate technical terms.

❍ The explanations in this manual refer to the standards ISO/IEC 7816-1 through 7816-5.
We strongly recommend reading this reference work.

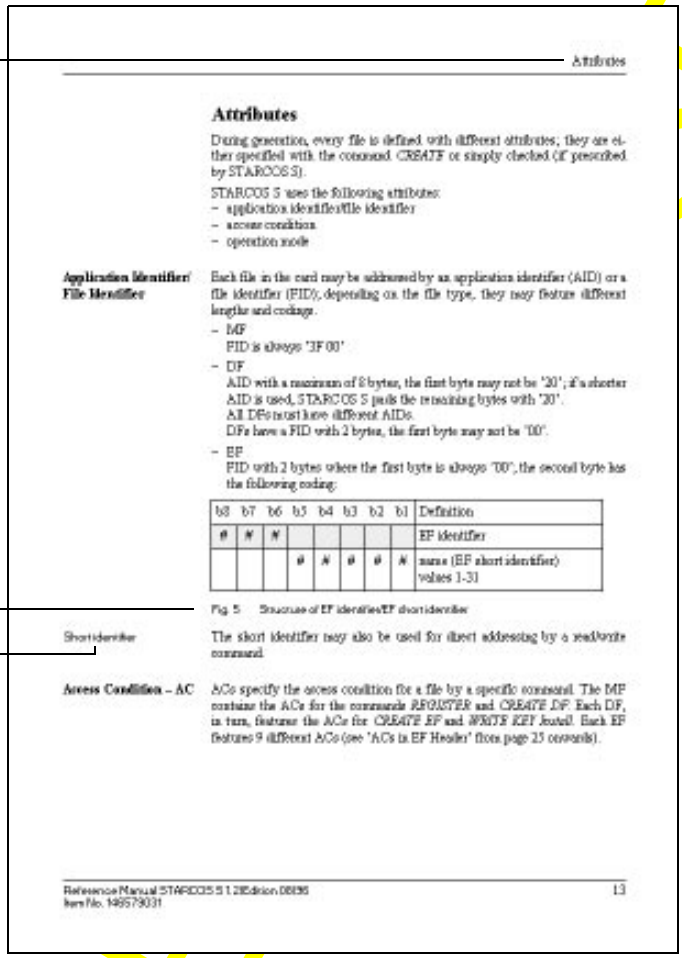
Guidance/Notation

In order to facilitate access to required information and to provide quick orientation, the following graphical aids are used:

chapter/subchapter

figure/table title

catchwords



Attributes

Attributes

During generation, every file is defined with different attributes; they are either specified with the command *CREATE* or simply checked (if prescribed by STARCOS S).

STARCOS S uses the following attributes:
- application identifier/file identifier
- access condition
- operation mode

Application Identifier/ File Identifier

Each file in the card may be addressed by an application identifier (AID) or a file identifier (FID), depending on the file type, they may feature different lengths and codings.

- MF
  FID is always '3F 00'
- DF
  AID with a maximum of 8 bytes, the first byte may not be '20'; if a shorter AID is used, STARCOS S pads the remaining bytes with '20'.
  All DFs must have different AIDs.
  DFs have a FID with 2 bytes, the first byte may not be '00'.
- EF
  FID with 2 bytes where the first byte is always '00', the second byte has the following coding:

b8	b7	b6	b5	b4	b3	b2	b1	Definition
0	x	x						EF identifier
			0	x	0	0	x	none (EF short identifier) values 1-31

Fig. 5 Structure of EF identifier/EF short identifier

Short identifier

The short identifier may also be used for direct addressing by a read/write command.

Access Condition – AC

ACs specify the access condition for a file by a specific command. The MF contains the ACs for the commands *REGISTER* and *CREATE DF*. Each DF, in turn, features the ACs for *CREATE EF* and *WRITE KEY*. Install. Each EF features 9 different ACs (see 'ACs in EF Header' from page 25 onwards).

Reference Manual STARCOS S 1.2/Edition 08/96 13
Order No. 146579031

The following notation is used:

catch words in the margin which are also part of the index

program elements such as commands

❍ Notes comprise of hints and recommendations useful when working with STARCOS S.

❒ ***Please read warnings carefully - they are given to prevent severe malfunctions and loss of data!***

The header page of each chapter features an overview of the topics covered in the chapter. All technical terms and abbreviations used are explained in a glossary at the end of the manual.

Files – Structures and Definitions

The STARCOS S functions provide secure storage for one or more applications and their data. This requires previous definition of data hierarchy and format.

The operating system STARCOS S organises data into files with different functions and administrates them in two levels. This scheme allows applications to be installed on any desired level.

Hierarchy

According to the standard ISO/IEC 7816-4, the STARCOS S file hierarchy is depicted as follows:

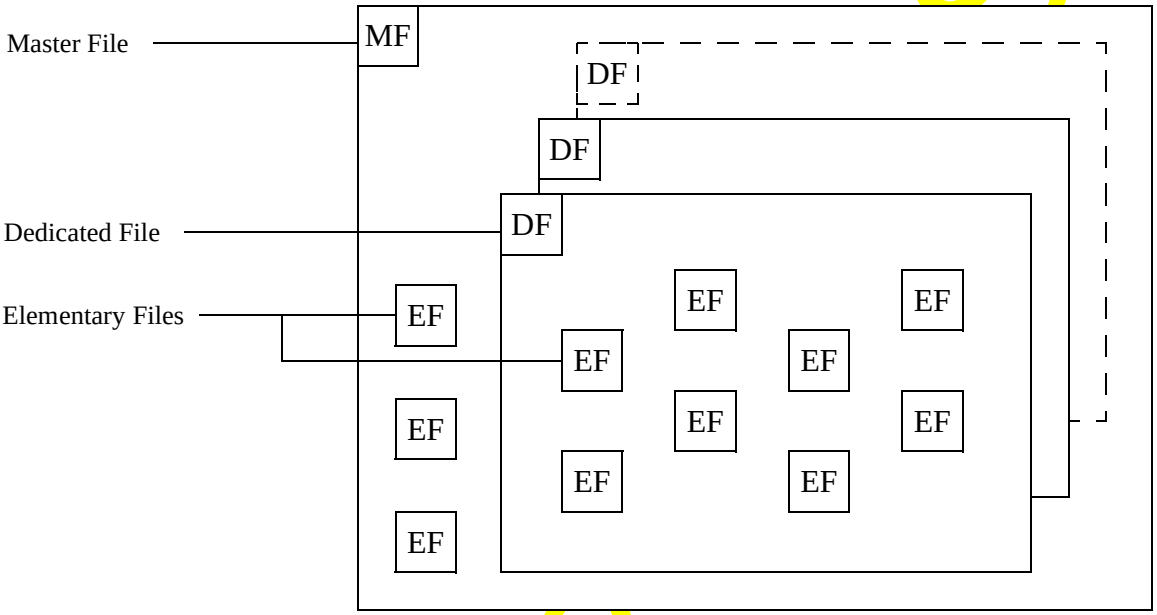


Fig. 1 File hierarchy according to ISO

The file hierarchy consists of the following levels:

– Master File (MF)
  constitutes the file system root (comparable to the root directory)
– Dedicated File (DF)
  a structure underlying the MF (comparable to a directory), which may feature EFs

The Elementary Files (EF) store the actual data of the applications and administrative information; these EFs are available at both levels (comparable to files).

Although a smartcard operating system such as STARCOS S features tasks and structures different from usual operating systems, its file hierarchy may still be depicted as a directory tree similar to the following example:

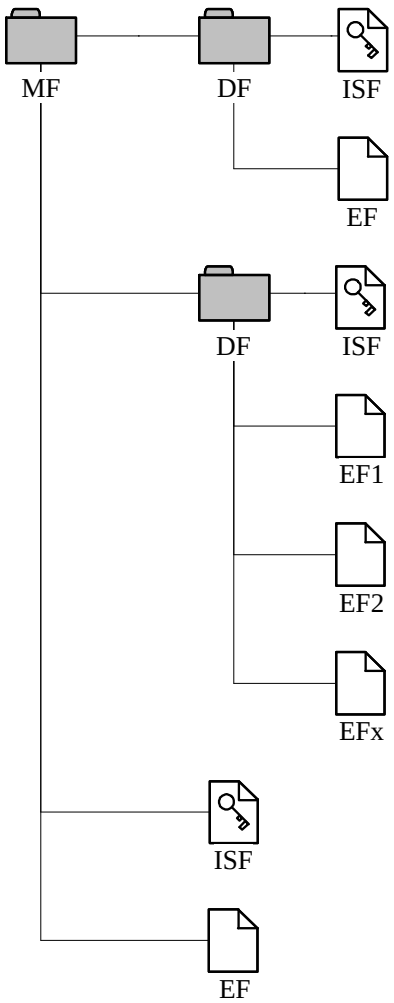


Fig. 2 File hierarchy depicted as a directory tree

Limitations

The limitations of STARCOS S, as compared to other operating systems, result from the card memory capacity and the operating system security concept:

- There is only the root level (MF) and one sub-level (DF).
- File access from a lower to a higher level is only possible for keys.
- The size of a level is specified during generation and is not dynamically determined by the number of files in this level.

Master File – MF

The Master File (MF) constitutes the file system root. Although the system allows applications in the MF, it is sensible for multi-functional cards to install only functions relating to card organisation. The MF then serves all applications as a common pool of card-specific functions such as serial numbers or card owner data.
The MF also defines whether or not new applications may be installed with the commands *REGISTER DF* and/or *CREATE*.

The MF will be activated as soon as a completed card has been contacted; deactivation is only possible if the card is no longer contacted. The MF cannot be deleted during the life of a card.

Dedicated File – DF

A Dedicated File (DF) stores all data of a single application. The application sequence is object-oriented thus enabling a separate security concept for every DF.

Physically, a DF is a static memory block, the size of which may not be changed after generation.

Every DF is separated both physically and logically from all other DFs; this scheme excludes any mutual influence of different applications. Several DFs may share common resources via the MF.

The installation of a DF is performed in two stages. Before the actual installation is performed with the command *CREATE*, the memory space must be allocated with the command *REGISTER DF*; this allows installation of applications independent of time or location (see 'Sample Generation of a STARCOS S Card' from page 117 onwards).

Elementary File – EF

Elementary Files (EF) constitute the actual data storage. STARCOS S distinguishes between the common elementary files (EF) for applications and special internal secret files (ISF) used by the operating system.

Elementary File (EF)

- contains application data
- can be read, modified, or overwritten by the terminal (depending on the defined security attributes)
- maximum of 31 per MF or DF
- no prescribed structure and attributes

❍ All FCBs and key records are secured with an error detection code (EDC); if this EDC is not correct, file access will be denied.

Internal Secret File (ISF)

- contains secret data (keys, PINs, etc.) for user authentication and cryptographic methods within the card

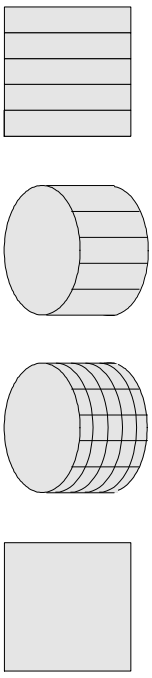
– cannot be read by the terminal
– created along with MF or DF (no file identifier required)
– keys may be entered, modified or overwritten
– only one per MF and DF
– prescribed structure linear fixed

Structures

STARCOS S supports the following four file structures:
– linear fixed
– cyclic
– compute
– transparent

The file structures differ in their access and storage modes. The decision which structure should be used for a given type of data depends on their variability and updating rate.

❍ The file structure for EFs may be chosen at random during generation; for ISFs, however, STARCOS S automatically generates the file structure linear fixed.



linear fixed	records with fixed length
cyclic	cyclic structure
compute	cyclic structure with fixed record structure
transparent	amorphous or transparent structure

Fig. 3 File structures

linear fixed

This file structure handles records of identical (fixed) length; it enables random access reading and writing, since every record is stored in a position specified by a unique record number. The operating system requires this number for any read or write operation.
According to ISO, this record number has a range from 1 through 254.

Reading/writing:

- random write sequence
- writing requires parameter information whether the record will be created new or will be overwritten
- data may be read partially (prefixes)

When a linear fixed file is created, STARCOS S calculates the allocated memory space and the record address with the following formulae:

allocated memory = records x × record length

❍ If the value is not a multiple of 4, it is automatically rounded off to a multiple of 4 bytes.

record address = record number × record length

cyclic

This file structure handles elements of identical length; it does not allow random access writing. The elements are stored in sequential order. However, only a limited number x of elements may be stored; when this number x is exceeded, the most obsolete element will be overwritten by a new one.

When a cyclic file is created, STARCOS S calculates the allocated memory space with the following formula:

allocated memory = (elements x × element length) + 4 bytes

The additional 4 bytes are required for the cyclic pointer.

Addressing for read operations uses a relative offset, called element offset. This means that the element written last has the number 1, the preceding element the number 2 and so on. A number exceeding the value x is rejected as invalid addressing.

compute

This file structure is based on the file structure cyclic. The first 5 bytes of such a record have a fixed purpose whereas possible further bytes cannot be addressed.

Write operations are only possible with the following commands:
– *INCREASE*
– *SECURE INCREASE*
– *DECREASE*
– *SECURE DECREASE*

❍ The command *UPDATE RECORD* is not possible for compute files.

3 bytes			2 bytes	
MSB		LSB	CS2	CS1
counter value			checksum	

Fig. 4 Record structure compute

Addressing for read operations uses a relative offset, called element offset. This means that the element written last has the number 1, the preceding element the number 2 and so on. A number exceeding the value x is rejected as invalid addressing.

transparent

This file structure has an amorphous or binary structure for the operating system. The terminal is responsible for addressing and managing data; STARCOS S only provides an allocated memory space.

Addressing uses an absolute offset with 2 bytes within the file with the first data byte of the file having the value ’00 00’.

Attributes

During generation, every file is defined with different attributes; they are either specified with the command *CREATE* or simply checked (if prescribed by STARCOS S).

STARCOS S uses the following attributes:

– application identifier/file identifier
– access condition
– operation mode

Application Identifier/ File Identifier

Each file in the card may be addressed by an application identifier (AID) or a file identifier (FID); depending on the file type, they may feature different lengths and codings.

– MF
  FID is always ’3F 00’
– DF
  AID with a maximum of 8 bytes, the first byte may not be ’20’; if a shorter AID is used, STARCOS S pads the remaining bytes with ’20’.
  All DFs must have different AIDs.
  DFs have a FID with 2 bytes, the first byte may not be ’00’.
– EF
  FID with 2 bytes where the first byte is always ’00’; the second byte has the following coding:

b8	b7	b6	b5	b4	b3	b2	b1	Definition
#	#	#						EF identifier
			#	#	#	#	#	name (EF short identifier) values 1-31

Fig. 5 Structure of EF identifier/EF short identifier

Short identifier

The short identifier may also be used for direct addressing by a read/write command.

Access Condition – AC

ACs specify the access condition for a file by a specific command. The MF contains the ACs for the commands *REGISTER* and *CREATE DF*. Each DF, in turn, features the ACs for *CREATE EF* and *WRITE KEY Install*. Each EF features 9 different ACs (see ’ACs in EF Header’ from page 25 onwards).

Operation Mode

The file operation mode specifies access attributes overriding the attributes in the ACs.

These attributes control file access with bits (flags) or bit combinations which may be categorised as follows.

Attributes for EFs:

– EF Locked
  This bit specifies whether access to this EF is permitted or not.

Attributes for ISFs:

– Write (WR)
  This bit specifies whether an existing key may be overwritten by the command *WRITE KEY.*
– Write Once (WO)
  This bit specifies that the key record may be overwritten only once by the command *WRITE KEY.*
– Virgin
  This bit indicates whether the key record has already been written or not.

Selection

STARCOS S offers two different modes for file selection; both assume that the file has been installed completely.

Prior to the actual selection of the new file, each file selection process causes a deactivation of the previously selected file on this level and of all levels below.

Explicit selection

Explicit selection is carried out with the command *SELECT* which uses the complete AID, the FID and the file level.

❍ All files may be selected explicitly, except for ISFs.

Implicit selection

Implicit selection is carried out with commands which perform EF operations and use the short identifier (e.g. *READ RECORD*).

❍ All EFs of the currently selected level may be selected implicitly.

Functions – Sequence and Interaction

The STARCOS S functions provide secure storage for one or more applications and their data. STARCOS S controls data access by using managers.

The manager functions range from data transmission to object-oriented administration and key management.

The basic principle of any STARCOS S activity is the reception of a command from the terminal and the transmission of a respective response from the card. Thus every STARCOS S function is a command processing activity which happens in the following functional units:

– Transmission Manager
  supervises correct data transmission.
– Command interpreter
  identifies the command with the class and instruction bytes; checks the parameters P1, P2 and P3 for their upper and lower limits (see 'Command' on page 43).
– Object-oriented security scheme
  controls the actual sequence of the application using defined initial and consecutive states.
– File Manager
  checks the required access rights before data operations are permitted.

Every command processing must be correct and complete, i.e. all commands must pass through the functional units or managers without any errors. If an error occurs, the respective error message will be issued.

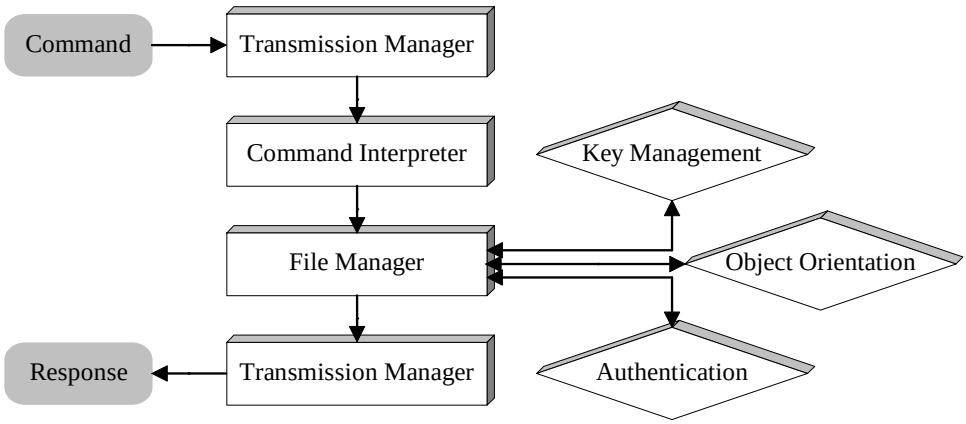


Fig. 6 Command processing scheme

Data Transmission – Transmission Manager

Purpose

The prerequisite for any STARCOS S activity is communication with a terminal; the Transmission Manager supervises the correct reception of a command and transmission of an appropriate response. Depending on the configuration, STARCOS S uses the block transmission protocol T=1 or the byte protocol T=0 for communication (both according to ISO/IEC 7816-3).

Transmission Method

The communication between terminal and card is always initiated by the terminal, which also supplies power to the card. As only one line is available for communication, the transmission is half duplex; i.e. only one device may transmit at a time.

The protocol T=1 specifies that the right to transmit always passes to the receiver of the last block (see ’Transmission Protocol T=1’ on page 114).

Answer To Reset – ATR

After the card has been contacted, the Transmission Manager transmits the Answer To Reset (ATR); the ATR contains card-specific data such as I/O buffer size and the conversion factor. In general, the ATR is transmitted after every physical reset, signalling the operational status of the card.

Protocol Type Select – PTS

As an option, a Protocol Type Select (PTS) according to ISO/IEC 7816-3 may be conducted after the ATR to select the protocol to be used (see ’Select transmission protocol’ on page 111).

Asynchronous transmission

STARCOS S uses an asynchronous transmission protocol procedure, i.e. every character transmitted is accompanied by additional synchronisation bits. The start bit at the beginning of a character signals a new character to the receiver; the bit duration gives enough time to prepare for reception. The stop bits at the end of a character give enough time to process the character internally without missing the next start bit.

Transmission control

The block transmission protocol T=1 and the byte protocol T=0 specify certain mechanisms for transmission control. In addition, the Transmission Manager conducts a centre-of-bit sampling. This happens three times; the Transmission Manager then makes a majority vote to determine the bit value; this compensates for simple line errors.

If transmission errors occur, they will be handled according to ISO/IEC 7816-3.

In the case of transmission errors with the T=1 protocol, the Transmission Manager reports them to the terminal immediately after a block end. The detection is done either by a parity bit with every byte, by the checksum at the end of the block or by a length check. In these cases, the terminal is requested to repeat the block. The terminal performs the same kind of error detection and handling (by repeating blocks).

In the case of transmission errors with the T=0 protocol, the respective byte will be repeated until the terminal indicates correct reception.

Data transfer

If a complete block has been received correctly, the Transmission Manager passes the command's information component to the next entity. After the command has been processed, the Transmission Manager will receive the information component of the response and transmit it to the terminal.

Sleep mode

After terminating the communication, the card utilises the processor sleep mode to reduce power consumption (see 'Smartcard Hardware' on page 116). Only those components detecting a new byte are supplied with power; this mechanism is controlled by the chip.

T=1 Restrictions

In contrast to the specifications of the block transmission protocol T=1, no waiting time extension (WTX) has been defined for STARCOS S, as there is no STARCOS S function with a time interval requiring WTX. A WTX from the terminal does not exist, since in sleep mode the card has no time-out limit while waiting for the next block.

Security Architecture – Sequence Control

Purpose

STARCOS S controls the application sequence using different states, allowing or forbidding certain actions. Access control for data is performed by state machines. STARCOS S has different state machines for the MF and each DF, which may have up to 16 different states.

State transition

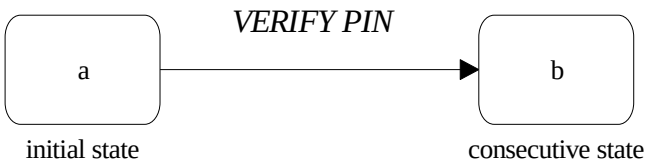


Fig. 7 State transition for the command VERIFY PIN

State transitions only occur with one of the following authentication commands: *VERIFY PIN*, *VERIFY AND CHANGE*, *EXTERNAL AUTHENTICATE* and *MUTUAL AUTHENTICATE*.

Since authentication commands always use a key, the Key Record stores the information on a state transition. The Key Records are stored in the ISF (see page 9) and contain the Access Control Values (ACV) as information for the state machine.

Access Control Value – ACV

The ACV contains:

– the access condition (AC), containing
  the initial state (IS),
  the compare operator (=/</≥/≠) and
  a reference to the current state (current or MF)
  (see ’current state’ on page 25)
– the consecutive state (CS)

Authentication

During the authentication process, the command *EXTERNAL AUTHENTICATE* performs the following three steps:

– compare the key Access Condition (AC) to the current state (current or MF)
– verify authentication data
– switch current state of the selected level (MF or DF) to the consecutive state.

If one of the three steps does not match, the command will be terminated and the current state will not be changed.

State transitions in the DF and MF

The following prerequisites must be fulfilled to make a DF state transition dependent on the current MF state:

– Separate state machines are available for MF and DFs.
– The compare operator allows flexible state verification (=/</≥/≠) with the current state.
– The initial state in the DF must refer to the current state of the MF.

For reasons of security, the current state of the MF may not refer to the current state of a DF. The consecutive state always switches only the state of the current level. It is thus possible to manipulate the MF state machine from the DF and vice versa to manipulate the DF state machine from the MF.

Example for a state transition

A card authenticates itself using a DES key, prior to switching to a DF. The application contained in the DF guarantees that the PIN verification will only be performed in the DF, if the DES authentication has been performed in the MF.

Prerequisites:

– The MF-ISF must contain a DES key, leading from the initial state a to the consecutive state b.
– The DF must contain a PIN requesting the MF state b as initial state, leading to the consecutive state c in the current state of the DF.

Sequence:

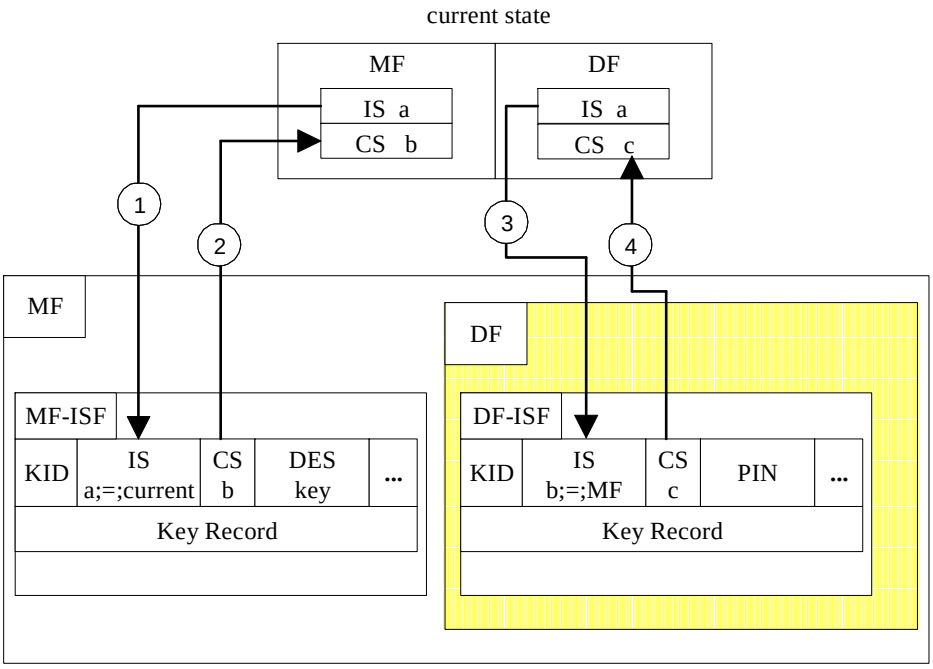


Fig. 8 State transitions

– Contact card:
The MF is selected automatically and the current state of the MF and DF is set to the initial state a.

– DES authentication:
A DES authentication is performed with the command *EXTERNAL AUTHENTICATE*. First, a random number is requested from the card with the command *GET CHALLENGE*. This random number is enciphered by the terminal with the key stored in the MF. The encipherment result and the KID (Key Identifier) are transmitted to the card with the command *EXTERNAL AUTHENTICATE*. The card now selects the key in the ISF and checks if the initial state (current state MF=a) has been fulfilled ①. If this is the case, the enciphered random number will be deciphered and compared to the random number stored in the card. If the numbers match, the current state of the MF will be switched to the consecutive state b ②.

– Select DF:
The command *SELECT* selects the DF. The current state of the DF is automatically re-initalised with the state a.

– Verify PIN:
The PIN and the KID are transmitted to the card with the command *VERIFY*. STARCOS S selects the PIN stored with the KID in the ISF and verifies if the initial state (current state MF=b) has been fulfilled ③. If this is the case, the PIN is compared to the PIN stored in the ISF. If the numbers match, the current state of the DFs will be switched to the consecutive state c ④.

Application Data Protection

The application data is stored in the EFs by the state transitions. EFs feature ACs for different access types (read, update, invalidate, rehabilitate, etc.). In contrast to the ACVs of keys (see page 20), EFs do not have a consecutive state, since only access to them is to be protected; the state machine must not be switched to another state. The ACs are established during EF generation.

Example for application data protection

An application in a DF contains an EF to be read after a DES authentication. Write is allowed after a PIN verification. The PIN verification is only allowed after a DES authentication.

Prerequisites:

– key management:
see prerequisites of the ’Example for a state transition’ on page 21.

– AC *READ* must refer to the current state of the MF and be in state b.

– AC *WRITE* must refer to the current state of the DF and be in state c.

– = is used for the compare operation, since the AC *READ* refers to the current state of the MF which cannot be changed by a PIN verification. This allows reading the EF after the PIN verification.

❍ If the DES authentication occurs in the DF, the AC *READ* must refer to the current state of the DF and contain the compare operator ≥. The compare operator = does not allow reading the EF after PIN verification, since the current state has been switched to c.

Sequence:

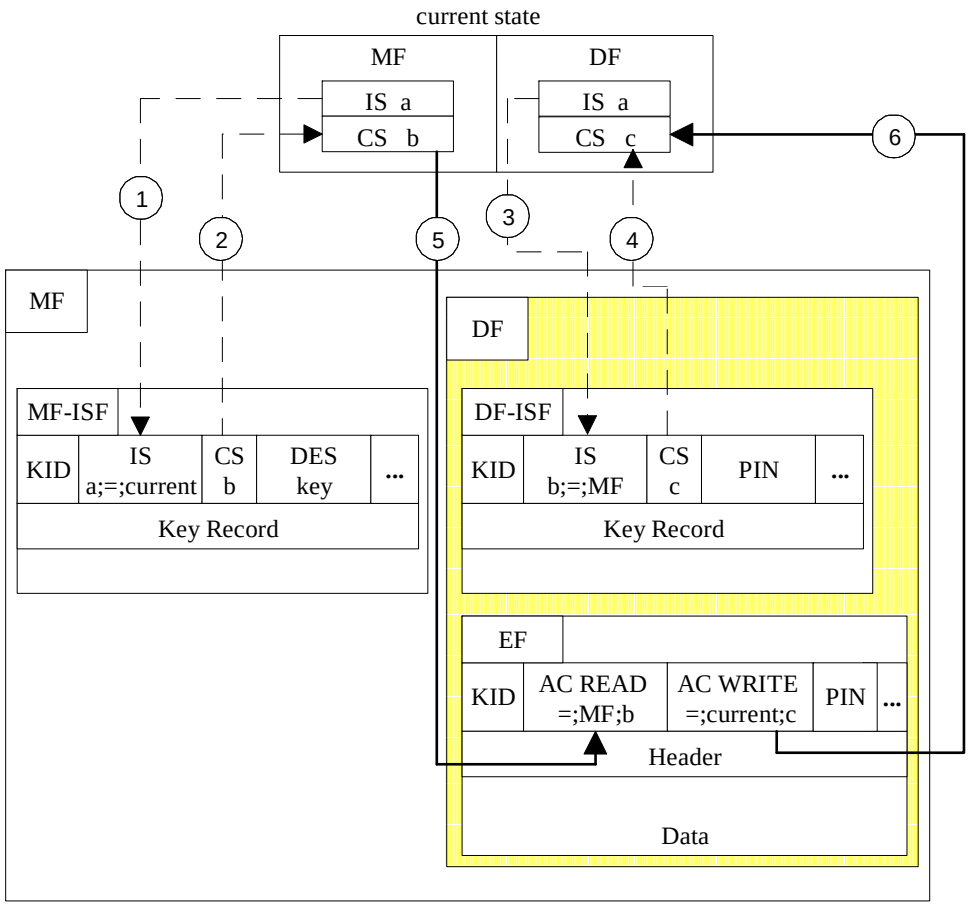


Fig. 9 Access protection

– Access to MF and DF:
see sequence as in the ’Example for a state transition’ on page 21 steps ① through ④.

– Read EF data:
After DF selection, the EF can be accessed with the command *READ*. The command *READ* verifies if the AC *READ* corresponds to the current state MF=b ⑤. Only if this is case, will data be transmitted with the response to the command *READ*.

– Write EF data:
After PIN verification, data will be overwritten in the EF with the command *UPDATE*. The command *UPDATE* verifies if the current state of the DF fulfils the AC *WRITE* ⑥. If this is the case, the data in the EF will be overwritten by the data transmitted with the command *UPDATE*.

Summary of the Examples

The table below illustrates the state transitions after the commands *EXTERNAL AUTHENTICATE*, *SELECT DF* and *VERIFY PIN* have been performed, as well as subsequent possible accesses.

Command	State Machine		Command	
	MF	DF	READ	WRITE
(Initial State)	a	a	no	no
EXTERNAL AUTHENTICATE	b	a	no	no
SELECT DF	b	a	yes	no
VERIFY PIN	b	c	yes	yes

Security Architecture – Structure

Access control for data is controlled by state machines (see ’Security Architecture – Sequence Control’ from page 20 onwards). In the following you will find a description of commands and bytes, working together during the state transitions.

State transitions

State transitions are only effected by the commands *VERIFY, VERIFY AND CHANGE, EXTERNAL AUTHENTICATE* and *MUTUAL AUTHENTICATE*. The state transition always refers to the current MF or DF state.

current state

The current state of the operating system is initialised with the value ’FF’ after a card reset. The byte for the current state is bipartite; the upper nibble contains coded the MF state and the lower nibble the state of the currently selected DF. This DF state is set to the initial state ’F’ after every command *SELECT* for the MF or a DF.

Access Control Value – ACV

The ACVs are the basis of the object-oriented security scheme; an ACV contains the following elements:

– the access condition (AC), containing
  the initial state (IS)
  the compare operator (=/</≥/≠) and
  a reference to the valid state (current or MF).
– the consecutive state (CS)

❍ If an AC is current for the current state, the state of the currently selected level will be used for comparison of the current state. If a DF has been selected, the current state of the DF will be compared; if the MF has been selected, the current state of the MF will be compared.

The ACVs are an essential component of the EF header, of the preheader for EF, ISF and DF and of the key records.

ACs in EF Header

EFs feature 9 different ACs containing the execution conditions for the commands of the same name. ACs in the EF header have the following sequence:

– READ
– WRITE
– RFU
– LOCK (lock/unlock)
– INCREASE
– DECREASE
– SECURE INCREASE
– SECURE DECREASE

Whenever a command accesses an EF, the respective AC is read first. Then the compare operation specified in the AC is performed with the current state (current or MF). If the compare operation is successful, the command will be executed; if it fails, processing will be aborted and an error message will be issued.

❍ If a command is not allowed for an EF, the respective AC will not be accessed.

Commands / AC	READ	WRITE	RFU	LOCK Lock	LOCK Unlock	INCREASE	DECREASE	SECURE INCREASE	SECURE DECREASE
DECREASE							✓		
INCREASE						✓			
LOCK FILE				✓	✓				
READ BINARY	✓								
READ RECORD	✓								
SECURE DECREASE									✓
SECURE INCREASE								✓	
UPDATE BINARY		✓							
UPDATE RECORD		✓							

Fig. 10 ACs in EF header

ACs in Preheader

Generation of a new DF, EF or ISF is controlled by the respective AC byte in the preheader.

– DF preheader
contains 2 ACs for the commands *REGISTER* and *CREATE DF*
– EF preheader
contains 1 AC for the command *CREATE EF*
– ISF preheader
contains 1 AC for the command *WRITE KEY Install*.

Commands \ AC	DF Preheader		EF Preheader	ISF Preheader
	REGISTER	CREATE DF	CREATE EF	CREATE KEY
CREATE		✓	✓	
REGISTER	✓			
WRITE KEY Install				✓

Fig. 11 ACs in preheader

ACV and AC in Key Records

Each key contains

– an ACV (see page 20) as execution condition for the key.
For a PUK, the consecutive state is replaced by the KID assigned to a PIN.

– an AC *WRITE* for the command *WRITE KEY Update*. The key can only be overwritten, if the AC *WRITE* has been fulfilled.

Commands	ACV	AC WRITE
CRYPT	✓	
EXTERNAL AUTHENTICATION	✓	
INTERNAL AUTHENTICATION	✓	
KEY STATE	✓	
MUTUAL AUTHENTICATION	✓	
VERIFY AND CHANGE	✓	
VERIFY PIN	✓	
WRITE KEY Update		✓

Fig. 12 ACV and AC in key record

Global Key

The global key is a special case for the state transition of the current MF or DF state. If a global key is successfully used in a DF (see ’Global Key’ on page 36), both the current state of the MF and the current state of the DF will be changed. This makes it possible to define a security scheme within a DF independent of the current state of the MF resulting from the use of the global key.

❍ It is thus possible to change the MF state from a DF; however, it is not possible to request or change a DF state from the MF level.

Resource Management – File Manager

Purpose

STARCOS S organises data into files of different structure and administrates them in two levels. The management uses an object-oriented approach, i.e. the files feature their own access conditions.

❍ The file attributes, such as access rights and data structure, are determined during generation.

Hierarchy

STARCOS S administrates files in the following levels:
– Master File (MF)
– Dedicated File (DF)

The Elementary Files (EF) store the actual application data and administrative information; these EFs are available at all levels.

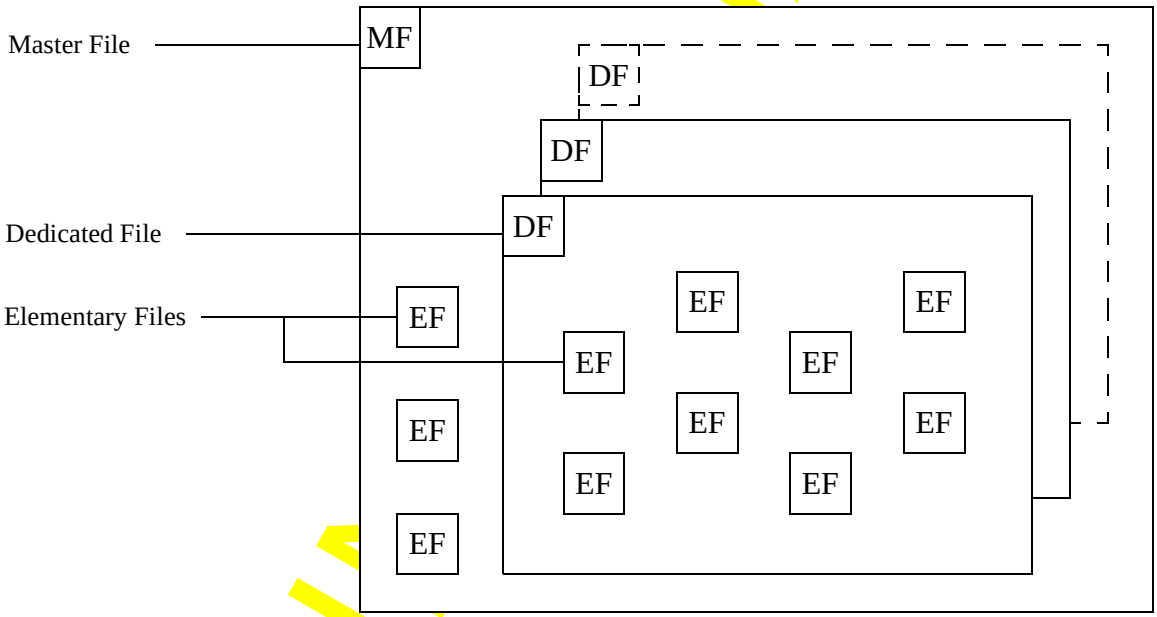


Fig. 13 File hierarchy

Structures

STARCOS S supports the following four file structures:
– linear fixed
– cyclic
– compute
– transparent

The file structures differ in their access and storage modes. The decision which structure should be used for a given type of data depends on their variability and updating rate.

❍ For detailed information see ’Structures’ from page 10 onwards.

Attributes

During generation, each file is defined with certain attributes; they are either specified with the command CREATE or simply checked (if prescribed by STARCOS S).

STARCOS S uses the following attributes:
– application identifier/file identifier
– access condition
– operation mode

❍ For detailed information see ’Attributes’ from page 13 onwards.

Level Incursions

The security concept of STARCOS S is based on the principle that applications do not influence each other. Nevertheless, applications may share common keys; such level incursions, however, are subject to the following rules:

– Only the resources of a level previously activated are accessible.
– If a global key is used, the ACs of both levels must be fulfilled.

Authentication

Purpose

The authentication procedures secure access both to data and to certain functions. Authentication complexity depends on the required security level and whether man or device has to be authenticated.

Personal Identification Number – PIN

STARCOS S allows using a key called PIN; this secret number comprises of 8 digits. Only the legitimate card user should know the PIN; the knowledge of this number authorises access to secured data or functions.
The user may be allowed several attempts to enter his PIN (typing error, mistake, etc.). Only after the preset number of attempts has been exceeded, will STARCOS S block the PIN. A blocked PIN may only be reactivated with a personal unblocking key (PUK). If the PIN is entered correctly, the counter is reset and the full number of attempts is available again for the next authentication.
STARCOS S supports counter limits between 1 and 14 attempts (see 'Presentation Counter' on page 38).

In addition, the PINs are subdivided as follows depending on their installation and validity:

Global PIN

– Global PIN
  Lower levels may use this PIN for authentication; if a global PIN is blocked because its counter has expired, it is also blocked for the other levels using it.
  The global PIN property is defined during generation in the msb (most significant bit) according to ISO coding or in the lower 3 bits according to STARCOS X coding (see page 37).

Current PIN

– Current PIN
  This PIN must be stored in the ISF of the currently selected level. The PIN cannot be accessed from other levels; if a current PIN is blocked because its counter has expired, the PIN check on the respective level is blocked.
  The current PIN property is defined during generation in the msb (most significant bit) according to ISO coding or in the lower 3 bits according to STARCOS X coding (see page 37).

Personal Unblocking Key – PUK

Like the PIN, the PUK is a key; the PUK, however, is secured by the operating system in a way that it may not be modified with the command *VERIFY AND CHANGE*. Only the command *WRITE KEY* may overwrite a PUK and reset its counter.

Administrative function

The PUK makes it possible to reactivate a PIN whose counter has expired. Therefore the PUK user has a certain administrative function. Since this function should be secured, the PUK counter should be set to a certain number of attempts.

As a PIN is often blocked because a user forgot his PIN, the PUK function always delivers a new PIN value.

A PUK is always assigned to a certain PIN; both must be stored in the same ISF and the PIN key identifier must be part of the PUK record.

❍ Every PUK can only be used for one PIN.

Data Encryption Standard – DES

STARCOS S uses the DES as its cryptographic algorithm for device authentication. The prerequisite for authentication is a common DES key for both devices.

Padding

Since DES processes only blocks with a multiple of 8 bytes, shorter data blocks must be extended correspondingly (padding); data blocks which are already a multiple of 8 are nevertheless extended with an additional block. The single blocks are chained using the CBC method (Cipher Block Chaining).

However, it must be visible which bytes have been padded; to ensure this, a block to be enciphered is first padded with a ’80’ byte and then completed with further ’00’ bytes to a multiple of 8.

CBC method

Data encipherment with the CBC method uses an XOR chain for the block Xi to be enciphered and the block Yi-1 previously enciphered and then the entire chain will be enciphered using DES; the result is Yi. Since the first block of a sequence to be enciphered does not yet contain a Yi-1, the initial value (‘00 00 00 00 00 00 00 00’) will be used.

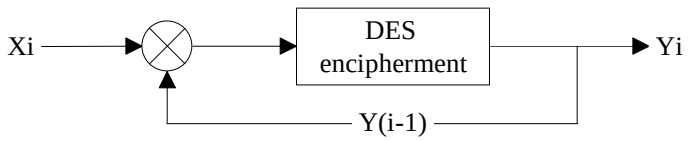


Fig. 14 Encipherment CBC method

Data decipherment with the CBC method uses an XOR chain for the block Xi to be deciphered and the previous block Xi-1; the result is Yi. Since the first block of a sequence to be deciphered does not yet contain a Xi-1, the initial value (‘00 00 00 00 00 00 00 00’) will also be used.

Fig. 15 Decipherment CBC method

Authentication process

The authenticated device receives a random number from the authenticating device, enciphers it and returns the value. The authenticating device deciphers the value and compares the result to the initial random number transmitted. If the numbers match, the authentication has been successful.

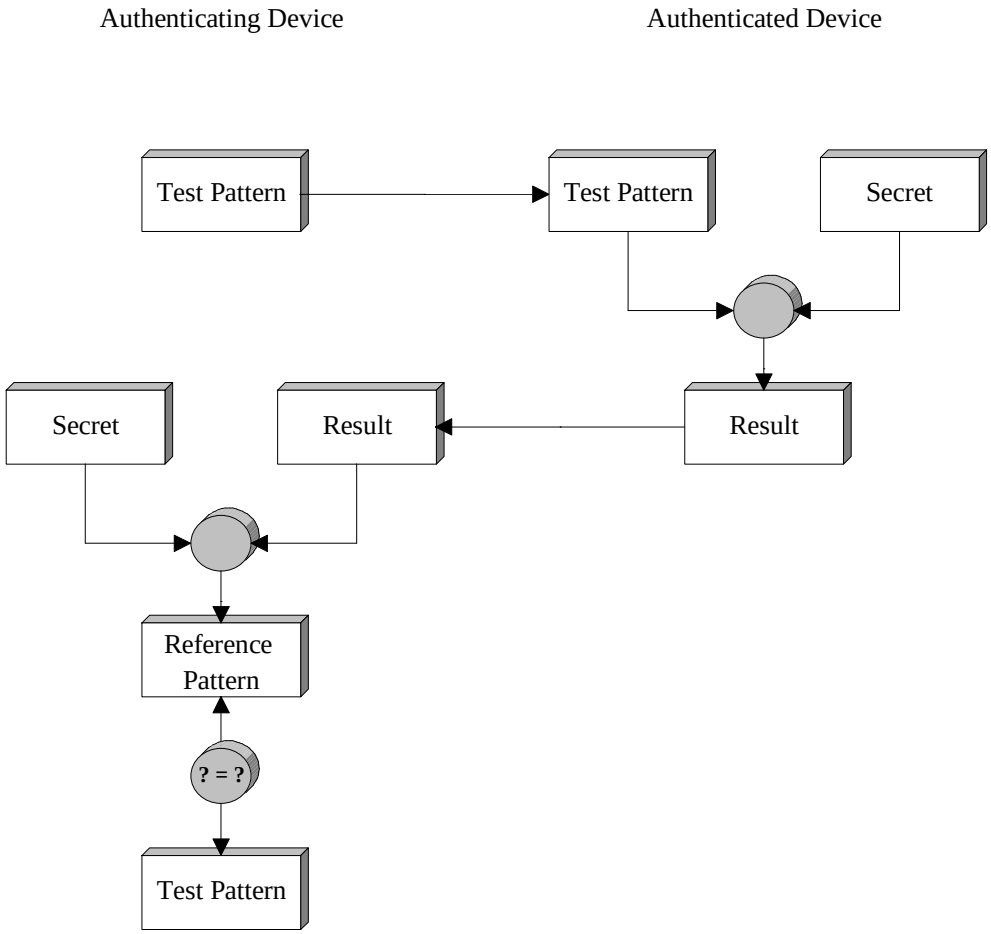


Fig. 16 Authentication Scheme

STARCOS S supports the following device authentication methods according to ISO/IEC 9798-2:

– Internal authentication
  the card authenticates itself to the terminal
– External authentication
  the terminal authenticates itself to the card
– Mutual authentication
  mutual authentication of card and terminal

❍ For security reasons, only keys derived from the global key should be used with the DES algorithm. The key calculation could use the card data entered in the system area during STARCOS S completion; this data can be read with the command *GET CARD DATA* (see page 61).

Internal authentication

The terminal reads the STARCOS S card data with the command *GET CARD DATA* and calculates the authentication key. The terminal generates a random number with 8 bytes and transmits it with the command *INTERNAL AUTHENTICATE* to the STARCOS S card.

The card enciphers the random number with the authentication key addressed in the command; the ciphered value is returned to the terminal. The terminal deciphers the ciphered value and verifies the random number.

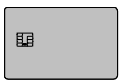
Terminal (IFD) 		Card (ICC) 
GET CARD DATA	→	
	←	carddata
calculate cardkey from carddata generate RND_{IFD}		
AUTH (KID, RND_{IFD})	→	
		$X = encrypt_{KID} (RND_{IFD})$
	←	X
$decrypt_{cardkey}$ (X) ?=? RND_{IFD}		

Fig. 17 Internal authentication

External authentication

The terminal reads the STARCOS S card data with the command *GET CARD DATA* and calculates the authentication key. As the card cannot transmit on its own, this authentication method is also controlled by the terminal. The terminal requests a random number with 8 bytes from the STARCOS S card, enciphers it with the authentication key and returns it with the command *EXTERNAL AUTHENTICATE* to the STARCOS S card. The card is requested to verify the ciphered value with the key.
The STARCOS S card then deciphers the ciphered value with the authentication key specified in the command and verifies the random number.

Terminal (IFD) 		Card (ICC) 
GET CARD DATA	→	
	←	carddata
calculate cardkey from carddata		
GET CHALLENGE	→	
	←	RND_{ICC}
X = $encrypt_{cardkey}$ RND_{ICC}		
AUT_E (KID, X)	→	
		$decrypt_{KID}$ (X) ?=? RND_{ICC}
	←	SW1 SW2

Fig. 18 External authentication

Mutual authentication

The terminal reads the STARCOS S card data with the command *GET CARD DATA* and calculates the authentication key. The terminal requests a random number with 8 bytes from the STARCOS S card and generates another random number with 8 bytes on its own. This random number, the random number from the card and the card data (concatenated in a string and enciphered with the CBC method) are enciphered with the authentication key; the terminal transmits this ciphered value X with the command *MUTUAL AUTHENTICATE* to the STARCOS S card. The card deciphers the ciphered value X with the authentication key addressed in the command and verifies its own random number and the card data. After successful verification, the STARCOS S card enciphers the terminal random number and its own random number (the sequence of the numbers being reversed) with its authentication key and returns the ciphered value Y to the terminal. The terminal deciphers the ciphered value Y and verifies its own random number.

Terminal (IFD) 		Card (ICC) 
GET CARD DATA	→	
	←	carddata
calculate cardkey from carddata		
GET CHALLENGE	→	
	←	RND_{ICC}
generate RND_{IFD} X = $encrypt_{cardkey}$ (RND_{IFD} || RND_{ICC} || carddata)		
AUT_M (KID, X)	→	
		$decrypt_{KID}$ (X) = (RND_{IFD} || RND_{ICC} || carddata) check of RND_{ICC} and carddata
	←	NO: ’63 Cx’ AUT-FAILED YES: Y = encrypt KID (RND_{ICC} || RND_{IFD})
$decrypt_{cardkey}$ (Y) ?=? RND_{ICC} || RND_{IFD}		

Fig. 19 Mutual authentication

Key Management

Purpose

Whenever a command uses keys, the last entity checks whether or not the required key is available and suitable for the intended purpose. The keys for each level are stored – irrespective of the authentication method used – in the internal secret file (ISF) with a unique key identifier (see ’WRITE KEY’ on page 94). After their installation, they never leave the card and are only used by card functions.

Global Key

In order to use a single PIN for several applications, for example, the global key is used. This key type in the MF is used from the DF level.

Before using a global key, a dummy key (key record) with the same KID as the global key must be created in the DF’s ISF; then a specific ACV is defined for the key record.
Whenever the global key should be used, the AC defined in the DF key record is checked first. Then the key with the corresponding KID is searched in the ISF of the MF and this key’s AC is checked.
If the ACs for both keys are fulfilled, the key will be used for the respective command. In this case, the MF state is overwritten with the consecutive state defined in the global key and the DF state is overwritten with the consecutive state defined in the dummy key (see ’Global Key’ on page 27).

Security scheme with global key

Using a global key makes it possible to define a security scheme within a DF independent of the MF state resulting from the use of the global key. As the ACVs of both keys are checked, the security scheme in the MF cannot be circumvented.

Addressing

Since every ISF includes several keys which may also be used globally, every key for a specific function must be addressed implicitly.

For proper addressing, STARCOS S requires the level of the respective ISF. This information is coded in the key identifier with the following structure:

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								level
0 1								global ISF of MF current ISF of current level, i.e. last activated level
	0	0						RFU
			#	#	#	#	#	number (1-31)

Fig. 20 Key addressing structure according to ISO/IEC 7816-4

b8	b7	b6	b5	b4	b3	b2	b1	Definition
#	#	#	#	#				number (1-31)
								level
					1 1	0 0	0 1	global ISF of MF current ISF of current level, i.e. last activated level

Fig. 21 Key addressing structure according to STARCOS X coding

Attributes

The attributes of a key determine its application as every command checks whether or not the key is suitable for the intended purpose.

❒ ***The attributes Authenticate and Crypt should never be used for the same DES key; otherwise the card may be misused for its own authentication. From the cryptographic perspective, the multiple use of keys generally involves a security risk.***

b8	b7	b6	b5	b4	b3	b2	b1	Definition
1								AKD exists
	1							SECURE DECREASE
		1						SECURE INCREASE
								RFU
				1				DES Authenticate
					1			DES Crypt
						1		PUK
							1	PIN

Fig. 22 Key attributes structure

If the bit b8 is set to 1, the Additional Key Description (AKD) provides the following settings for DES Authenticate.

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								RFU
								MUTUAL AUTHENTICATION
					1 0			allowed not allowed
								EXTERNAL AUTHENTICATION
						1 0		allowed not allowed
								INTERNAL AUTHENTICATION
							1 0	allowed not allowed

Fig. 23 Additional Key Description

Presentation Counter

The key fault presentation counter (KFPC) blocks a key, if the preset number of attempts has been exceeded. With every unsuccessful attempt a counter is decremented; if the counter value is 0, the key will be blocked for further use. If an authentication is correct, the counter will be reset and the full number of attempts is available again for the next authentication.

An expired KFPC may only be reset with the command *VERIFY AND CHANGE* (see page 92) using the PUK (see ’Personal Unblocking Key – PUK’ on page 30).

❍ If initial value and error counter are set to ’FF’, the key fault presentation counter is not active.

Key Records

Each key is stored as a record which is secured with a checksum by STARCOS S.

The key record has the following structure:

– Header
  contains information about the purpose, etc.; identical for all key types
– Body
  actual key value
– Checksum
  secures the complete record; calculated by the operating system.

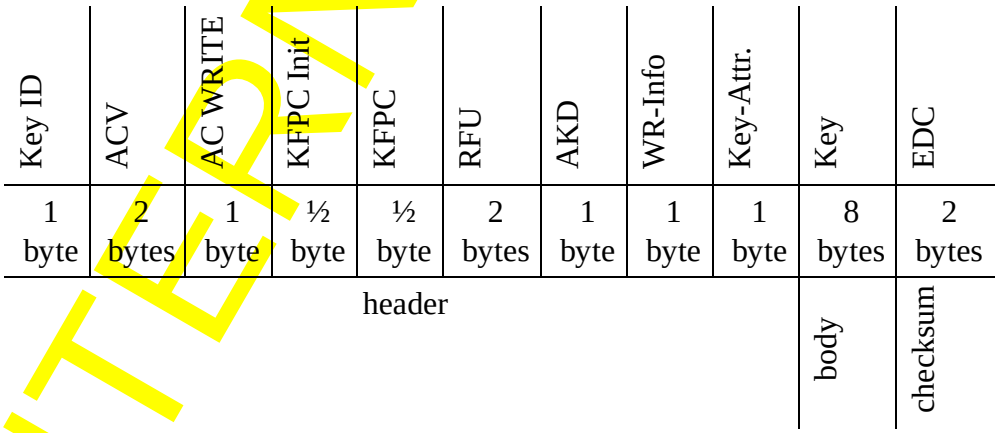


Fig. 24 Key record structure

Commands

The prerequisite for any activity of the STARCOS S card is the communication with a terminal via a command-response pair (CRP).
The terminal always generates the command; thus the card with its response is the reacting party.

The STARCOS S commands and their responses are covered in the following chapter, both with an overview table and with detailed explanations in alphabetical order.

Structures

Command

A command transmitted from the terminal to the card consists of a minimum of 5 bytes (header); whether this header is followed by a data component (body) or not depends on the command.

Header					Body	
CLA	INS	P1	P2	P3	DATA	L_e

Fig. 25 Command structure

CLA – Class Byte

The class byte coding differentiates between commands according to ISO (CLA = ’00’) and private use commands (CLA = PU, value set in the EEPROM with default CLA = ’80’).

INS – Instruction Byte

The instruction byte contains the coding of the command. As a standard, STARCOS S uses the ISO coding; if a command is not defined in ISO, STARCOS S will either use an instruction code from ETSI/CEN or its own code.

P1/P2 – Parameter

The parameters P1 and P2 are used for subdividing a command; their interpretation depends on the respective command.

P3/DATA/L_e

The interpretation of parameter P3 depends on the respective command; according to ISO/IEC 7816-4, four different schemes, called CASES, are defined:

– CASE 1
commands which do not transmit or receive data.
$P3 = L_e = L_c = 0$

– CASE 2
commands which do not transmit data to the card but receive data from the card.
$P3 = L_e$
Example *READ RECORD*

CLA	INS	P1	P2	P3
’00’	’B2’	’00’	’04’	’05’

Response

DATA	SW1	SW2
A B C D E	’90’	’00’
L_e		

$P3 = L_e = 5$
$P3 = 0$ complete record transmitted
$P3 = 1 - 5$ specified number of bytes transmitted

– CASE 3
commands which transmit data to the card but do not receive data from the card.
$P3 = L_c$
Example *UPDATE RECORD*

CLA	INS	P1	P2	P3	DATA
’00’	’D2’	’00’	’04’	’05’	A B C D E
					L_c

– CASE 4
commands which transmit data to the card and receive data from the card.
$P3 = L_c$
L_e = length of expected data

❍ L_e may be omitted in CASE 4. If L_e is not included, the card sets $L_e = 0$ for the command.

Example *INTERNAL AUTHENTICATE*

CLA	INS	P1	P2	P3	DATA	L_e
’00’	’88’	’00’	KID	’08’	RND	’08’
					L_c	

Response

DATA	SW1	SW2
x1 x2 x3 x4 x5 x6 x7 x8	’90’	’00’
L_e		

$P3 = L_c = 8$
$L_e = 0$ all data transmitted
$L_e = 1 - 8$ specified number of bytes transmitted

❍ For protocol T=1, parameter $L_e = 0$ will cause all response data to be transmitted.

Response

The card has to react with a response to a terminal command. The response consists of a minimum of 2 status bytes (trailer) constituting the status message. The additional data component (body) may contain the required data and depends on the command.

Body	Trailer	
DATA	SW1	SW2

Fig. 26 Response structure

The typical status message for correct command execution is SW1 = ’90’ and SW2 = ’00’.

❍ All status bytes described in the following are in hexadecimal format.

Overview

Commands

Command	Code (INS byte)	Usage	defined in ISO/IEC 7816-4
CREATE	’E0’	file management	
CRYPT	’F8’	cryptography	
DECREASE	’30’	counter function	
EXCHANGE CHALLENGE	’80’	authentication	
EXTERNAL AUTHENTICATE	’82’	authentication	✓
GET CARD DATA	’F6’	authentication	
GET CHALLENGE	’84’	authentication	✓
GET RESPONSE	’C0’	data transmission	✓
INCREASE	’32’	counter function	
INTERNAL AUTHENTICATE	’88’	authentication	✓
KEY STATUS	’F2’	key management	
LOCK FILE	’76’	file management	
MUTUAL AUTHENTICATE	’8A’	authentication	
READ BINARY	’B0’	file management	✓
READ RECORD	’B2’	file management	✓
REGISTER DF	’52’	file management	
SECURE DECREASE	’34’	counter function	
SECURE INCREASE	’36’	counter function	
SELECT FILE	’A4’	file management	✓

Command	Code (INS byte)	Usage	defined in ISO/IEC 7816-4
UPDATE BINARY	’D6’	file management	✓
UPDATE RECORD	’DC’	file management	✓
VERIFY	’20’	authentication	✓
VERIFY AND CHANGE	’24’	authentication	
WRITE KEY	’F4’	key management	

Fig. 27 Command overview

Completion commands

STARCOS S features three special commands used for card completion only which are covered in the chapter ’Card Completion’ from page 101 onwards.

Command	Code (INS byte)	Usage	defined in ISO/IEC 7816-4
CHECK KEY	’10’	completion	
LOAD COMPLETION DATA	’12’	completion	
COMPLETION END	’14’	completion	

Fig. 28 Completion command overview

Responses – Status Byte

Code	Description
’61 00’	too much data
’61 xx’	for T=0 only normal processing; xx bytes response data available
’62 82’	end of file reached
’62 84’	file locked
’63 Cx’	incorrect key, x represents the number of retries *or* incorrect PIN, x represents the number of retries
’64 00’	*GET RESPONSE* not expected
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state

Code	Description
’69 81’	incorrect file type *or* PUK record contains pointer to incorrect key type
’69 82’	no file selected
’69 83’	key locked
’69 85’	no valid random number present *or* MF not selected *or* file already exists *or* incorrect key type (no PIN/PUK)
’6A 00’	key not found *or* PIN not found *or* file not found *or* record not found *or* DF-ID or DF name already exists *or* file not registered yet *or* incorrect EF-ID
’6A 80’	underflow *or* incorrect DF-ID *or* incorrect parameter(s) in data component
’6A 84’	insufficient memory
’6A 86’	parameters P1/P2 incorrect
’6B 00’	incorrect offset
’6C xx’	incorrect length; maximum available length xx

Code	Description
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 00’	no data available
’6F 81’	system error
’90 00’	normal processing

Fig. 29 Response overview

The STARCOS S commands and their status bytes are covered in the following with detailed explanations in alphabetical order.

CREATE

The command *CREATE* generates MFs or DFs with the respective EFs; during this process, the command also allocates memory and creates the DF-header as well as the EF and ISF preheaders. All file-specific attributes must be entered; they are also partially checked for plausibility.
The command *CREATE* features the following functions:

– MF
A card in the original delivery state has no MF; therefore, an MF must first be generated using the command *CREATE MF*. After key comparison, the memory for the MF is allocated and the DF header and EF/ISF preheaders are generated. During this process, the access conditions for the commands *REGISTER* and *CREATE* are specified (DF preheader). The allocated memory is specified in bytes. If the value is not a multiple of 4, it is automatically rounded off to a multiple of 4 bytes.

– DF
Before a DF may be generated with *CREATE*, the command *REGISTER* must be used to allocate the required memory.
The memory to be allocated for the ISF must be specified in bytes. If the value is not a multiple of 4, it is automatically rounded off to a multiple of 4 bytes.

– EF
After generating an MF or a DF, an EF may be generated with *CREATE EF*.
For EF generation, the required net memory (excluding header and pointer) is specified in the description byte. If the complete EF memory requirement is not a multiple of 4, it is internally rounded off to a multiple of 4 bytes. However, only the bytes specified in the description byte may be addressed afterwards.

– End
The generation of an MF or DF must be terminated with *CREATE End*; the ACs of the respective file level are only checked after this termination. Thus, the security policy of the file to be generated must not be taken into consideration during initialisation.
If *CREATE End* for the MF has been executed, then the AC *CREATE DF* must be fulfilled; else creation of DFs is free.
If *CREATE End* for a DF has been executed, the AC *CREATE EF* and the AC *WRITE KEY Install* must be fulfilled; else generation of EFs and installation of keys is always possible.
The command *CREATE End* activates the security mechanism for a file level. Until the command *CREATE End* has been executed, STARCOS S will not check the ACs of the respective file level. Thus it is possible to generate a read-only EF file by generating a file with an AC *WRITE* of never and writing the file data, before *CREATE End* for the DF is executed.

The command *CREATE DF* may only be executed if the AC *CREATE* of the DF preheader is fulfilled or if *CREATE End* has not been performed in the MF.

Note

- The key required for MF generation is stored in the EEPROM outside the application area.

Command

CLA	INS	P1	P2	L_c	DATA
PU	'E0'	OP	'00'		

OP operation mode
'00' MF
'01' DF
'02' End
'03' EF

L_c length

DATA structure for *CREATE MF*

DATA for *CREATE MF*

KEY	MF SPACE	ISF SPACE	EF-AC	ISF-AC	DF-CRE-AC	DF-REG-AC
8 bytes	2 bytes	2 bytes	1 byte	1 byte	1 byte	1 byte

KEY
key entered in the card during initialisation (protected by a KFPC)

- For test cards, the original delivery state may be restored with the command *DELETE MF*; the key for the test cards is '01 02 03 04 05 06 07 08'.

MF SPACE
features the following structure:

Use	Number/bytes (decimal)	Availability
MF header	20	
EF preheader	8	per MF
EF header	20	per EF
DATA SPACE	x	per EF
cyclic pointer	4	per cyclic and compute EF
ISF SPACE	x	per MF

ISF SPACE
features the following structure:

Use	Number/bytes (decimal)	Availability
preheader	8	
key	20	per key

EF-AC EF access condition byte
contains AC for command *CREATE EF*

ISF-AC ISF access condition byte
contains AC for command *WRITE KEY Install*

DF-CRE-AC DF Create access condition byte
contains AC for command *CREATE DF*

DF-REG-AC DF Register access condition byte
contains AC for command *REGISTER DF*

All access condition bytes feature the following coding:

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								compare operation
0	0							= < ≥ ≠
1	0							
0	1							
1	1							
								RFU
								reference value
			0					current state MF current state current
			1					
				#	#	#	#	compare value for reference value 0 = highest state 15 = lowest state

DATA structure for *CREATE DF*

DATA for *CREATE DF*

DF-ID	DF-AID	ISF SPACE	EF-AC	ISF-AC
2 bytes	8 bytes	2 bytes	1 byte	1 byte

DF-ID DF identifier
1st byte ≠ '00'
2nd byte random

DF-AID DF application identifier
8 bytes, blanks must be padded with ’20’

ISF-SPACE (see page 52)

EF-AC and ISF-AC (see page 52)

DATA structure for *CREATE End*

DATA for *CREATE End*

DF-ID
2 bytes

DF-ID DF identifier
1st byte ≠ ’00’
2nd byte random

DATA structure for *CREATE EF*

DATA for *CREATE EF*

EF-ID	AC	RFU	EF-INFO	EF-Description
2 bytes	9 bytes	2 bytes	1 byte	2 bytes

EF-ID EF identifier
EF-ID = ’00 xx’;
in order to select with EF short identifier, ’xx’ must be ≤ ’1F’

AC access condition byte
contains the ACs for the respective commands (see ’ACs in EF Header’ on page 25), coding see page 52

EF-INFO
contains information about structure and attributes

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								EF attribute
0 1								locked unlocked
								RFU
								structure
				0 0 0 1	0 0 1 1	0 1 0 0	1 0 0 0	binary linear fixed cyclic compute

EF-Description

for cyclic or linear fixed structure

number of records	record length
1 byte	1 byte

for transparent structure

data length
2 bytes

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’69 85’	MF not selected *or* file already exists
’6A 00’	file not registered yet *or* incorrect EF-ID
’6A 80’	incorrect parameter(s) in data component
’6A 84’	insufficient memory
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

CRYPT

The command *CRYPT* makes it possible to use the STARCOS S card as an encipherment module.

❍ *CRYPT* is only available for STARCOS S versions 1.2-1-1-0, 1.2-1-2-0, and 1.2-1-3-0 (see 'Versions' on page 116).

CRYPT features six different modes:

– encrypt without following data/encrypt with following data
enciphering input text

– decrypt without following data/decrypt with following data
deciphering input text

– MAC without following data/MAC with following data
calculate MAC for input text; MAC length is 4 bytes.

All six modes use the CBC method (Cipher Block Chaining) for the DES (see 'Data Encryption Standard – DES' on page 31).

The command *CRYPT* may only be executed if the AC of the key used is fulfilled.

Notes

❍ The DES key addressed in *CRYPT* must be in the ISF and must have the attribute DES Crypt.

❍ The DES encipherment uses padding according to ISO/IEC 9797, method 2.

❍ The MAC consists of the first 4 bytes of the block enciphered last.

❍ When using *CRYPT* with following data, P1 = '0x' in the last command indicates end of data.

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	'F8'	OP	KID			'00'

OP operation mode
'00' encrypt without following data
'01' decrypt without following data
'02' MAC without following data
'80' encrypt with following data
'81' decrypt with following data
'82' MAC with following data

KID key identifier
identifies key in the ISF
(for coding see 'WRITE KEY' from page 94 onwards)

L_c length

decrypt	maximum buffer size, must be a multiple of 8
encrypt, MAC	maximum buffer size with following data, but only maximum buffer size minus 1 byte in the last block due to padding

DATA input text

L_e length of expected response data

Response

DATA	SW1	SW2
encrypted/decrypted data or MAC	’90’	’00’

Status Bytes

Code	Description
’61 xx’	for T=0 only normal processing; xx response data available
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’69 85’	incorrect key type
’6A 00’	key not found
’6A 86’	parameters P1/P2 incorrect *or* data length for decrypt no multiple of 8
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

DECREASE

The command *DECREASE* decreases a value stored in the card (e.g. for a counter or purse function). The card receives only the modification value and calculates the new current value itself.

The current value is stored and changed in an EF with the structure compute. The file structure compute is based on the EF file structure cyclic, but the records have a fixed structure. This file can be read using the command *READ RECORD*, but cannot be written with the command *UPDATE RECORD*. The current value is read from this file, decreased by the given modification value and the result is stored as the new current value.

Internally, the current value is automatically secured with a checksum (2 bytes, EDC, AT&T); when creating this file, element size must be ≥ 5 (see 'compute' on page 12).

The command *DECREASE* may only be executed if the AC *DECREASE* of the EF is fulfilled.

Notes

- During installation, the starting element of an EF is written automatically with DATA = '00 00 00 00 00'.
- The EF may be selected with:
  - *SELECT FILE*
  - *READ RECORD* (with EF short identifier)
  - *INCREASE*
  - *DECREASE*
  - *SECURE INCREASE*
  - *SECURE DECREASE*
  - *CREATE*

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	'30'	EF	'00'	'03'		'00' - '06'

EF elementary file
file to be used

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0 #	0 #	0 #	0 #	0 #				currently selected EF short identifier (1-31)
					1	1	0	

DATA modification value (3 bytes)

Response

DATA	SW1	SW2
new current value || modification value	’90’	’00’

Status Bytes

Code	Description
’61 06’	for T=0 only normal processing; 6 bytes response data available
’62 84’	file locked
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file not found
’6A 80’	underflow
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

EXTERNAL AUTHENTICATE

The terminal uses the command *EXTERNAL AUTHENTICATE* to authenticate to the STARCOS S card using the cryptographic DES method. For this authentication, the terminal enciphers a random number requested from the card (see 'External authentication' on page 33).

The command *EXTERNAL AUTHENTICATE* may only be executed if the AC of the key used is fulfilled.

Notes

- *EXTERNAL AUTHENTICATE* requires a prior request from the card for a random number; the random number must still be valid.
- The current random number is rendered invalid by the commands *CRYPT*, *INTERNAL AUTHENTICATE*, *EXTERNAL AUTHENTICATE* and *MUTUAL AUTHENTICATE* in order to prevent multiple use of the random number.
- The key must be stored in an ISF having the attribute Authenticate, but the selection is carried out explicitly by the terminal.
- The DES encipherment for authentication uses no padding.
- As the card performs the check, the KFPC of the key used may be altered.
- The key should be allowed explicitly for *EXTERNAL AUTHENTICATE*.
- After successful execution, the consecutive state defined in the key record is attained.

Command

CLA	INS	P1	P2	L_c	DATA
'00'	'82'	'00'	KID	'08'	$e_K(RND_{ICC})$

KID key identifier
identifies key in the ISF
(for coding see 'WRITE KEY' from page 94 onwards)

$e_K(RND_{ICC})$ random
authentication data; enciphered random number RND_{ICC} from the STARCOS S card

Response

SW1	SW2
'90'	'00'

Status Bytes

Code	Description
’61 08’	for T=0 only normal processing; 8 bytes response data available
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’69 85’	incorrect key type *or* no valid random number present
’6A 00’	key not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

GET CARD DATA

The command *GET CARD DATA* either reads a serial number, the version number of the operating system or the chip configuration data. The serial number is unique for every card and may be used to calculate card-specific keys.

Notes

- For a definite identification of the card, the chip serial number coded by the manufacturer is used; the number consists of 8 bytes.
- The version number of the operating system is coded in TLV format and contains information about:
  - operating system version
  - customer version
  - card functions.

  For detailed information see 'Versions' from page 116 onwards.
- The chip configuration data comprises manufacturer information such as the EEPROM write time and the chip ID.
- As it is a system command, *GET CARD DATA* is always permitted.

Command

CLA	INS	P1	P2	L_e
PU	'F6'	'00'	OP	

OP operation mode
'00' serial number
'01' version number of operating system
'02' chip configuration data

L_e length
'00' all bytes are returned

Response

DATA	SW1	SW2
card data	'90'	'00'

Status Bytes

Code	Description
'61 xx'	for T=0 only normal processing; xx bytes response data available
'67 00'	incorrect length
'6A 86'	parameters P1/P2 incorrect
'6D 00'	incorrect INS
'6E 00'	incorrect CLA
'6F 81'	system error

GET CHALLENGE

The command *GET CHALLENGE* serves for the request of random numbers from the STARCOS S card. STARCOS S stores the random numbers transferred in RAM.

Notes

- The random number is always 8 bytes long, although 1 to 8 bytes may be read. In case of an authentication, however, all 8 bytes must be used.
- A random number transmitted will be stored in RAM until
  - it is used for a command *CRYPT*, *INTERNAL AUTHENTICATE*, *EXTERNAL AUTHENTICATE* or *MUTUAL AUTHENTICATE*
  - a new random number is transmitted.

Command

CLA	INS	P1	P2	L_e
’00’	’84’	’00’	’00’	’00’ - ’08’

L_e length
’00’ all 8 bytes are returned

Response

DATA	SW1	SW2
random number	’90’	’00’

Status Bytes

Code	Description
’61 08’	for T=0 only normal processing; 8 bytes response data available
’65 00’	write error
’67 00’	incorrect length
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

GET RESPONSE

❍ The command *GET RESPONSE* is only used with transmission protocol T=0.

The terminal requests response data from the STARCOS S card with the command *GET RESPONSE*.

A so-called CASE 4 command cannot transmit and receive data simultaneously; therefore, the actual application command must be followed by a command *GET RESPONSE* – according to ISO/IEC 7816-4.
The STARCOS S card confirms correct execution of the CASE 4 command with the code ’61 xx’, xx indicating the length of available response data. This data may be retrieved by one (L_e = xx) or more ($L_e \leq$ xx) commands *GET RESPONSE*.

Note

❍ If the CASE 4 command used is executed correctly, the length of the response data will be indicated in SW2.

Command

CLA	INS	P1	P2	L_e
’00’	’C0’	’00’	’00’	

L_e length of expected response data

Response

DATA	SW1	SW2
response data	’90’	’00’

Status Bytes

Code	Description
’61 xx’	for T=0 only normal processing; xx bytes response data available
’64 00’	*GET RESPONSE* not expected
’6C xx’	incorrect length; maximum available length xx
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 00’	no data available
’6F 81’	system error

INCREASE

The command *INCREASE* increases a value stored in the card (e.g. for a counter or purse function). The card receives only the modification value and calculates the new current value itself.

The current value is stored and changed in an EF with the structure compute. The file structure compute is based on the EF file structure cyclic, but the records have a fixed structure.This file can be read using the command *READ RECORD*, but cannot be written with the command *UPDATE RECORD*. The current value is read from this file, increased by the given modification value and the result is stored as the new current value.

Internally, the current value is automatically secured with a checksum (2 bytes, EDC, AT&T); when creating this file, element size must be ≥ 5 (see 'compute' on page 12).

The command *INCREASE* may only be executed if the AC *INCREASE* of the EF is fulfilled.

Notes

- During installation, the starting element of an EF is written automatically with DATA = '00 00 00 00 00'.
- The EF may be selected with:
  - *SELECT FILE*
  - *READ RECORD* (with EF short identifier)
  - *INCREASE*
  - *DECREASE*
  - *SECURE INCREASE*
  - *SECURE DECREASE*
  - *CREATE*

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	'32'	EF	'00'	'03'		'00' - '06'

EF elementary file
file to be used

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0 #	0 #	0 #	0 #	0 #				currently selected EF short identifier (1-31)
					1	1	0	

DATA
modification value (3 bytes)

L_e length
’00’ all 6 bytes are returned

Response

DATA	SW1	SW2
new current value || modification value	’90’	’00’

Status Bytes

Code	Description
’61 xx’	for T=0 only normal processing; xx bytes response data available
’62 84’	file locked
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file not found
’6A 80’	overflow
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

INTERNAL AUTHENTICATE

The STARCOS S card uses the command *INTERNAL AUTHENTICATE* to authenticate to the terminal using the cryptographic DES method. For this authentication, STARCOS S enciphers the random number received (see 'Internal authentication' on page 33).

❍ *INTERNAL AUTHENTICATE* is only available for STARCOS S versions 1.2-1-1-0, 1.2-1-2-0 and 1.2-1-3-0 (see 'Versions' on page 116).

The command *INTERNAL AUTHENTICATE* may only be executed if the AC of the key used is fulfilled.

Notes

- The key must already be stored in an ISF having the attribute Authenticate.
- The DES encipherment for authentication uses no padding.
- As the terminal performs the check, the KFPC in the card is not altered.
- The current random number is marked as invalid.
- The key should be allowed explicitly for *INTERNAL AUTHENTICATE*.

Command

CLA	INS	P1	P2	L_c	DATA	L_e
'00'	'88'	'00'	KID	'08'	RND_{IFD}	'00' - '08'

KID key identifier
identifies key in the ISF
(for coding see 'WRITE KEY' from page 94 onwards)

RND_{IFD} random
authentication data; random number from the terminal

Response

DATA	SW1	SW2
e_K (RND)	'90'	'00'

Status Bytes

Code	Description
’65 00’	EDC error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’69 85’	incorrect key type
’6A 00’	key not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

KEY STATUS

After specification of a key identifier, the command *KEY STATUS* returns both the current and the initial values of the error counter KFPC.

The command *KEY STATUS* may only be executed if the AC of the key used is fulfilled.

Command

CLA	INS	P1	P2	L_c
PU	’F2’	’00’	KID	’00’ - ’01’

KID key identifier
identifies key in the ISF
(for coding see ’WRITE KEY’ from page 94 onwards)

Response

DATA	SW1	SW2
initial KFPC, KFPC	’90’	’00’

Status Bytes

Code	Description
’61 01’	for T=0 only normal processing; 1 byte response data available
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’6A 00’	key not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

LOCK FILE

The command *LOCK FILE* locks or unlocks an EF. A locked file may not be read/written, until it has been unlocked with a second command *LOCK FILE*.

LOCK FILE features two modes:

– Unlock
  corresponds to Rehabilitate according to ETSI 726-3

– Lock
  corresponds to Invalidate according to ETSI 726-3

The command *LOCK FILE* may only be executed if the corresponding AC for *LOCK Lock* or *LOCK Unlock* is fulfilled.

Notes

- *LOCK FILE* can only be applied to EFs, not to DFs or the MF.
- The command *LOCK FILE* changes the bit File locked in the EF-Info byte of the FCB.

Command

CLA	INS	P1	P2	L	DATA
PU	’76’	’03’	OP	’02’	FID

OP operation mode
  ’00’ Unlock
  ’FF’ Lock

FID file identifier (2 bytes)
  1st byte = file qualifier = ’00’
  2nd byte = file index

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’62 84’	file locked
’65 00’	EDC error or write error
’67 00’	incorrect length
’6A 00’	file not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

MUTUAL AUTHENTICATE

The command *MUTUAL AUTHENTICATE* is used for mutual authentication of the STARCOS S card and the terminal using the cryptographic DES method (see ’Mutual authentication’ on page 34).

The command *MUTUAL AUTHENTICATE* may only be executed if the AC of the key used is fulfilled.

Notes

- The authentication requires a prior terminal request for a random number from the card (command *GET CHALLENGE*) and a request for the serial number (command *GET CARD DATA*).
- The DES encipherment for authentication uses no padding.
- As the card also performs the check, the KFPC of the key used may be altered.
- The current random number is marked as invalid.
- The key should be allowed explicitly for *MUTUAL AUTHENTICATE*.
- After successful execution, the consecutive state defined in the key record is attained.

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	’8A’	ACP	KID	’18’	$e_K(RND_{IFD} \| RND_{ICC} \| CD)$	’00’ - ’10’

ACP application control parameter
’45’ key in the ISF

KID key identifier
identifies key in the ISF
(for coding see ’WRITE KEY’ from page 94 onwards)

RND_{IFD} random
authentication data; random number from the terminal

RND_{ICC} random
authentication data; random number from the STARCOS S card

CD serial number (see ’GET CARD DATA’ on page 61)

Response

DATA	SW1	SW2
$e_K(RND_{ICC} \| RND_{IFD})$	’90’	’00’

Status Bytes

Code	Description
’61 10’	for T=0 only normal processing; 10 bytes response data available
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’69 85’	incorrect key type *or* no valid random number present
’6A 00’	key not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

READ BINARY

The command *READ BINARY* reads data only from EFs with transparent structure. The parameters P1/P2 specify the starting position for the read operation (if necessary, also the file); the parameter L_e specifies the number of bytes to be read.

The command *READ BINARY* may only be executed if the AC *READ* of the EF is fulfilled.

Notes

- The EF may be selected with:
  - *SELECT FILE*
  - *CREATE*
  - *LOCK FILE Unlock*
  - *READ BINARY*/*UPDATE BINARY* (with EF short identifier)
- Data of a blank file is ’FF’.
- Data in a file locked with *LOCK FILE* cannot be read.
- If an EF short identifier is used, the maximum value for the offset is ’FF’.

Command

CLA	INS	P1	P2	L_e
’00’	’B0’	OH	OL	

OH offset high byte for explicit selection

OH = 0#######

b8	b7	b6	b5	b4	b3	b2	b1	Definition
0								
								offset
	#	#	#	#	#	#	#	OH offset high byte

offset = OH x ’100’ + OL (offset low byte)

OH offset high byte for implicit selection

OH = 100#####

b8	b7	b6	b5	b4	b3	b2	b1	Definition
1	0	0						
								file selection
			#	#	#	#	#	short identifier (1-31)

offset = OL (offset low byte)

L_e number of bytes to be read (maximum buffer size)

’00’ all bytes up to end of file

Response

DATA	SW1	SW2
requested data	’90’	’00’

Status Bytes

Code	Description
’61 00’	too much data
’61 xx’	for T=0 only normal processing; xx bytes response data available
’62 82’	end of file reached
’62 84’	file locked
’65 00’	EDC error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file not found
’6A 86’	parameters P1/P2 incorrect
’6B 00’	incorrect offset
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

READ RECORD

The command *READ RECORD* reads records/elements from EFs; addressing depends on the respective file structure. It is also possible to read only prefixes. The required EF is either already selected or specified in the parameter P2 of *READ RECORD*.

The command *READ RECORD* may only be executed if the AC *READ* of the EF is fulfilled.

Notes

- The EF may be selected with:
  - *CREATE*
  - *SELECT FILE*
  - *LOCK FILE Unlock*
  - *READ RECORD*/*UPDATE RECORD* (with EF short identifier)
- *READ RECORD* can only read one record/element at a time.
- *READ RECORD* cannot be applied to EFs with transparent structure.
- Data in a file locked with *LOCK FILE* cannot be read.
- If P2 contains an EF short identifier, it always refers to an EF of the currently selected level.

Command

CLA	INS	P1	P2	L_e
’00’	’B2’	NR	AM	

NR number; depends on file structure
linear fixed record number of record to be read (’01’-’FE’)
cyclic number of element to be read (’01’-’FE’)

AM access mode

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0	0	0	0	0				currently selected EF short identifier (1-31)
					1	0	0	coding of P1 record/element number

L_e number of bytes to be read (maximum buffer size)
’00’ all bytes of record/element

❍ From EFs with the structure compute a maximum of 5 bytes may be read, irrespective of the specified record length (see ’compute’ on page 12).

Response

DATA	SW1	SW2
requested data	’90’	’00’

Status Bytes

Code	Description
’61 xx’	for T=0 only normal processing; xx bytes response data available
’62 84’	file locked
’65 00’	EDC error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file or record not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

REGISTER DF

The command *REGISTER DF* allocates a physical memory block for a DF and specifies FID and AID of the DF. The installation of the DF with the command *CREATE DF* (see page 50) may then be conducted later.

The memory requirement of an application may be calculated as follows:

Use	Number/bytes (decimal)	Availability
DF header	20	per DF
EF preheader	8	per DF
ISF preheader	8	per DF
EF header	20	per EF
key (header + body)	20	per key
cyclic pointer	4	per cyclic and compute EF

Notes

- *REGISTER DF* can only be performed from the MF level.
- If sufficient EEPROM memory is not available, *REGISTER DF* will be aborted.
- During registration, the required memory must be specified in bytes. If the value is not a multiple of 4, it is automatically rounded off to a multiple of 4 bytes.
- An application identifier with ’00 00 00 00 00 00 00 00’ is incorrect.

Command

CLA	INS	P1	P2	L_c	DATA
PU	’52’	SH	SL	’0A’	DF-ID || AID

SH SPACE high byte

SL SPACE low byte

DF-ID DF identifier
1st byte ≠ ’00’
2nd byte random

AID application identifier
8 bytes, blanks must be padded with ’20’

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 85’	MF not selected
’6A 00’	DF-ID or DF-name already exists
’6A 80’	incorrect DF-ID
’6A 84’	insufficient memory
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

SECURE DECREASE

The command *SECURE DECREASE* decreases a value stored in the card (e.g. for a counter or purse function). The card receives only the modification value and calculates the new current value itself.

The current value is stored and changed in an EF with the structure compute. The file structure compute is based on the EF file structure cyclic, but the records have a fixed structure. This file can be read using the command *READ RECORD*, but cannot be written with the command *UPDATE RECORD*. The current value is read from this file, decreased by the given modification value and the result is stored as the new current value. This value modification is only possible after an authentication using a MAC in command and response.

Internally, the current value is automatically secured with a checksum (2 bytes, EDC, AT&T); when generating this file, element size must be ≥ 5 (see 'compute' on page 12).

The command *SECURE DECREASE* may only be executed if the AC *SECURE DECREASE* of the EF is fulfilled.

Notes

- The first element is initialised automatically with '00 00 00 00 00'.
- *SECURE DECREASE* requires a prior request from the card for a random number; the random number must still be valid.
- The EF may be selected with:
  - *SELECT FILE*
  - *INCREASE*
  - *SECURE INCREASE*
  - *DECREASE*
  - *SECURE DECREASE*
  - *READ RECORD* (with EF short identifier)
  - *CREATE*
- The MAC (4 bytes) in the command is generated from an XOR operation of the first 8 bytes of the transmitted command with a random number (8 bytes) requested from the card. The result is the basis for the MAC operation (using DES) with only the first 4 bytes being added to the command.
  MAC_{IFD} = bytes $_{1\text{-}4}$ of
  $e_K(RND_{ICC}$ XOR (CLA || INS || P1 || P2 || L_c || modification value))
  In addition, the terminal provides a random number with 8 bytes which the card uses for calculating the ciphered value in the response.

- The MAC (4 bytes) in the response is generated from an XOR operation of the random number provided by the terminal (8 bytes total; 6 bytes from response with 2 bytes ’00’). The result is the basis for the MAC operation (using DES) with only the first 4 bytes being added to the response.
  MAC_{ICC} = bytes $_{1-4}$ of
  $e_K(RND_{IFD}$ XOR (new current value || modification value || ’00 00’))
  The complete response has 10 bytes (3 bytes new current value, 3 bytes modification value, and 4 bytes MAC).

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	’34’	EF	KID	’0F’	modification value || MAC_{IFD} || RND_{IFD}	’00’ - ’0A’

EF elementary file
file to be used

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0 #	0 #	0 #	0 #	0 #				currently selected EF short identifier (1-31)
					1	1	0	

KID key identifier
identifies key in the ISF
(for coding see ’WRITE KEY’ from page 94 onwards)

DATA
modification value (3 bytes), followed by MAC (4 bytes) and random number RND_{IFD} (8 bytes) used by the card for generating the ciphered value in the response

Response

DATA	SW1	SW2
new current value || modification value || MAC_{ICC}	’90’	’00’

Status Bytes

Code	Description
’61 0A’	for T=0 only normal processing; 10 bytes response data available
’62 84’	file locked
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’69 83’	key locked
’69 85’	incorrect key type *or* no valid random number present
’6A 00’	file or key not found
’6A 80’	underflow
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

SECURE INCREASE

The command *SECURE INCREASE* increases a value stored in the card (e.g. for a counter or purse function). The card receives only the modification value and calculates the new current value itself.

The current value is stored and changed in an EF with the structure compute. The file structure compute is based on the EF file structure cyclic, but the records have a fixed structure. This file can be read using the command *READ RECORD*, but cannot be written with the command *UPDATE RECORD*. The current value is read from this file, increased by the given modification value and the result is stored as the new current value. This value modification is only possible after an authentication using a MAC in command and response.

Internally, the current value is automatically secured with a checksum (2 bytes, EDC, AT&T); when generating this file, element size must be ≥ 5 (see 'compute' on page 12).

The command *SECURE INCREASE* may only be executed if the AC *SECURE INCREASE* of the EF is fulfilled.

Notes

- The first element is initialised automatically with '00 00 00 00 00'.
- *SECURE INCREASE* requires a prior request from the card for a random number; the random number must still be valid.
- The EF may be selected with:
  - *SELECT FILE*
  - *INCREASE*
  - *SECURE INCREASE*
  - *DECREASE*
  - *SECURE DECREASE*
  - *READ RECORD* (with EF short identifier)
  - *CREATE*
- The MAC (4 bytes) in the command is generated from an XOR operation of the first 8 bytes of the transferred command with a random number (8 bytes) requested from the card. The result is the basis for the MAC operation (using DES) with only the first 4 bytes being added to the command.
  MAC_{IFD} = bytes $_{1\text{-}4}$ of
  $e_K(RND_{ICC}$ XOR (CLA || INS || P1 || P2 || L_c || modification value))
  In addition, the terminal provides a random number with 8 bytes which the card uses for calculating the ciphered value in the response.

- The MAC (4 bytes) in the response is generated from an XOR operation of the random number provided by the terminal (8 bytes total; 6 bytes from response with 2 bytes ’00’). The result is the basis for the MAC operation (using DES) with only the first 4 bytes being added to the response.
  MAC_{ICC} = bytes $_{1\text{-}4}$ of
  $e_K(RND_{IFD}$ XOR (new current value || modification value || ’00 00’))
  The complete response has 10 bytes (3 bytes new current value, 3 bytes modification value and 4 bytes MAC).

Command

CLA	INS	P1	P2	L_c	DATA	L_e
PU	’36’	EF	KID	’0F’	modification value || MAC_{IFD} || RND_{IFD}	’00’ - ’0A’

EF elementary file
file to be used

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0	0	0	0	0				currently selected EF short identifier (1-31)
					1	1	0	

KID key identifier
identifies key in the ISF
(for coding see ’WRITE KEY’ from page 94 onwards)

DATA
modification value (3 bytes), followed by MAC (4 bytes) and random number RND_{IFD} (8 bytes) used by the card for generating the ciphered value in the response

Response

DATA	SW1	SW2
new current value || modification value || MAC_{ICC}	’90’	’00’

Status Bytes

Code	Description
’61 0A’	for T=0 only normal processing; 10 bytes response data available
’62 84’	file locked
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’69 83’	key locked
’69 85’	incorrect key type *or* no valid random number present
’6A 00’	file or key not found
’6A 80’	underflow
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

SELECT FILE

The command *SELECT FILE* activates an existing file or file level. When applying *SELECT FILE* to

– EF
STARCOS S opens the respective file, which can then be addressed by some commands directly without prior implicit selection.

– DF
the complete application is activated, i.e. STARCOS S changes to the new level.

The MF may be selected from any level with *SELECT FILE*.

Notes

- EFs are selected using a file identifier (FID); DFs are selected either using an FID or an application identifier (AID).
- The selection of a new application automatically deactivates the application previously selected.

Command

CLA	INS	P1	P2	L_c	DATA
’00’	’A4’	SC	SO		FID/AID

SC selection control
’00’ selection with FID
’02’ selection with EF-ID
’04’ selection with AID

SO selection options
’00’ no file control information

FID file identifier
EF-FID = 2 bytes, 1st byte = ’00’
DF-FID = 2 bytes, 1st byte ≠ ’00’
MF-FID = ’3F 00’

AID application identifier
AID = 8 bytes

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’65 00’	EDC error
’67 00’	incorrect length
’6A 00’	file not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

UPDATE BINARY

The command *UPDATE BINARY* writes data only in EFs with transparent structure. Data to be written may be a multiple of one byte and may have an offset length value of up to 15 bits.
The parameters P1/P2 specify the starting position for the write operation (if necessary, also the file); the parameter L_c specifies the number of bytes to be written.

The command *UPDATE BINARY* may only be executed if the AC *UPDATE* of the EF is fulfilled.

Notes

- The EF may be selected with:
  - *SELECT FILE*
  - *CREATE*
  - *LOCK FILE Unlock*
  - *READ BINARY*
  - *UPDATE BINARY*
- Data of a blank file is ’FF’.
- Data cannot be written to a file locked with *LOCK FILE*.
- If the number and position of bytes to be written exceeds the file range, no data will be written at all.
- Data in the file will be overwritten by new data (update).
- *UPDATE BINARY* enables implicit selection of an EF with FID in P1 and 8 bits offset length.

Command

CLA	INS	P1	P2	L_c	DATA
’00’	’D6’	OH	OL		

OH offset high byte for explicit selection
OH = 0#######

b8	b7	b6	b5	b4	b3	b2	b1	Definition
0								
								offset
	#	#	#	#	#	#	#	OH offset high byte

offset = OH x ’100’ + OL (offset low byte)

OH offset high byte for implicit selection

OH = 100#####

b8	b7	b6	b5	b4	b3	b2	b1	Definition
1	0	0						
								file selection
			#	#	#	#	#	short identifier (1-31)

offset = OL (offset low byte)

L_c record length

length of data to be written, maximum buffer size

DATA

data to be written

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’62 84’	file locked
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file not found
’6A 86’	parameters P1/P2 incorrect
’6B 00’	incorrect offset
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

UPDATE RECORD

The command *UPDATE RECORD* writes data in EFs with linear fixed or cyclic structure. New data is always written as a complete record, as it is not possible to change only part of a record. The required EF is either already selected or specified in the parameter P2 of *UPDATE RECORD*.

UPDATE RECORD features two modes:

– Update (for files with linear fixed structure)
  existing record will be overwritten.
– Append (for files with cyclic structure)
  new record/element will be generated; due to the cyclic structure, the most obsolete record will be overwritten.

The command *UPDATE RECORD* may only be executed if the AC *UPDATE* of the EF is fulfilled.

Notes

- The EF may be selected with:
  – *CREATE*
  – *SELECT FILE*
  – *LOCK FILE Unlock*
  – *READ RECORD*/*UPDATE RECORD* (with EF short identifier)
- *UPDATE RECORD* can only write one record/element at a time.
- *UPDATE RECORD* cannot be applied to EFs with transparent structure.
- Data cannot be written to a file locked with *LOCK FILE*.
- If P2 contains an EF short identifier, it always refers to an EF of the currently selected level.
- Files with cyclic structure do not allow addressing elements to be overwritten.

Command

CLA	INS	P1	P2	L_c	DATA
’00’	’DC’	NR	AM		

NR number
  depends on file structure
  linear fixed record number of record to be written (’01’ - ’FE’)
  cyclic number of element to be written (’00’)

AM access mode

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								file selection
0 #	0 #	0 #	0 #	0 #				currently selected EF short identifier (1-31)
								coding of P1
					1	0	0	for linear fixed: overwrite existing record
					1 (0	1 1	0 1)	for cyclic: (according to ISO/IEC 7816-4) generate record/element

L_c record length

length of data to be written, maximum buffer size

DATA data to be written

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’62 84’	file locked
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	incorrect file type
’69 82’	no file selected
’6A 00’	file or record not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

VERIFY

The command *VERIFY* performs a user authentication with the PIN. The authentication is a comparison of the stored secret number with the value transmitted; the process will only be successful, if the values match in all 8 bytes (see 'Global Key' on page 27).

❍ If the authentication fails, the KFPC value will be decremented by one; if the KFPC is 0, the PIN record will be blocked. If, however, a certain number of attempts is still allowed, this value will be indicated in the lower nibble of SW2.

Notes

- A blocked PIN makes the authentication fail, but does not change the KFPC.
- After successful execution, the consecutive state defined in the key record is attained.

Command

CLA	INS	P1	P2	L_c	DATA
'00'	'20'	'20'	KID	'08'	

KID key identifier
identifies key in the ISF
(for coding see 'WRITE KEY' from page 94 onwards)

DATA
PIN

Response

SW1	SW2
'90'	'00'

Status Bytes

Code	Description
’63 Cx’	incorrect PIN, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked or no valid global PIN
’69 85’	incorrect key type (no PIN)
’6A 00’	PIN not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

VERIFY AND CHANGE

The command *VERIFY AND CHANGE* combines the function of the command *VERIFY* (which in this case also authenticates the PUK) with one of the following functions:

– changing value of an existing PIN

– resetting the KFPC of a PIN
The PUK authentication resets the expired KFPC of a PIN and gives a new PIN value. The key identifier of the respective PIN is stored in the PUK record.

Notes

- The PIN record is changed only after a successful user authentication with PIN or PUK (see ’VERIFY’ on page 90).
- The reactivation of a PIN is only possible from the currently selected level. It is thus not possible to reactivate a DF PIN or MF PIN from a DF using an MF PUK.
- After successful execution, the consecutive state defined in the key record is attained.

Command

CLA	INS	P1	P2	L_c	DATA
PU	’24’	ACP	KID	’10’	

ACP application control parameter
’20’ verify PIN and change PIN
’30’ verify PUK and change PIN

KID key identifier; identifies verification key in the ISF
(for coding see ’WRITE KEY’ from page 94 onwards)

Further structure of DATA:

VERIFY DATA	NEW VAL
8 bytes	8 bytes

VERIFY DATA PIN/PUK value
NEW VAL new PIN value

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’63 Cx’	incorrect PIN, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	PUK record contains pointer to incorrect key type
’69 83’	key locked or no valid global PIN
’69 85’	incorrect key type (no PIN/PUK)
’6A 00’	PIN not found
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

WRITE KEY

The command *WRITE KEY* creates or overwrites authentication keys and user keys in the ISF.
STARCOS S supplies every key with an internal EDC code of 2 bytes.

WRITE KEY features two modes:

– Install
  A new key is generated; *WRITE KEY* will not be executed, if the KID specified in parameter P2 is already used.

– Update
  An already existing key is overwritten; *WRITE KEY* will not be executed, if the KID specified in parameter P2 does not have a key.

The command *WRITE KEY* overwrites a key only if the bits WR or WORM and Virgin have been set in the key header byte Write Info.

❒ ***From the cryptographic perspective, the multiple use of keys generally involves a security risk (see 'Attributes' on page 29).***

Note

❍ Keys always have a length of 8 bytes.

Command

CLA	INS	P1	P2	L_c	DATA
PU	'F4'	OP	KID		

OP operation mode
'00' Install
DATA contains key header and key
'01' Update
DATA contains only key

KID key identifier

contains key number and information about the level (MF/DF) of the ISF
Key addressing structure according to ISO/IEC 7816-4

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								level
0 1								master ISF of MF current ISF of current level, i.e. last activated level
	0	0						RFU
			#	#	#	#	#	number (1-31)

Key addressing structure according to STARCOS X coding

b8	b7	b6	b5	b4	b3	b2	b1	Definition
#	#	#	#	#				number (1-31)
								level
					1 1	0 0	0 1	master ISF of MF current ISF of current level, i.e. last activated level

L_c length of key header + key component

depending on OP parameter

OP	L_c
'00' Install	'11'
'01' Update	'08'

Further structure of DATA:

Key Header								
ACV	WR-AC	KFPC-Init	KFPC	RFU	AKD	WR-Info	Key-Attr.	Key
2 bytes	1 byte	½ byte	½ byte	2 bytes	1 byte	1 byte	1 byte	8 bytes

ACV byte 1

contains the key's AC; the key may only be used if the AC is fulfilled.

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								compare operation
0 0 1 1	0 1 0 1							= < ≥ ≠
								reference value
			0 1					current state MF current state current
				#	#	#	#	compare value for reference value 0 = highest state 15 = lowest state

ACV byte 2

key ≠ PUK

byte 2 contains consecutive state which is accessed after authentication with this key

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								consecutive state (positive case)
				0 1	0 1	0 1	0 1	highest state lowest state

key = PUK

byte 2 contains the KID corresponding to the PIN to be unblocked

b8	b7	b6	b5	b4	b3	b2	b1	Definition
#	#	#	#	#				number (1-31)
								level
					1	0	1	current ISF of current level, i.e. last activated level

❍ For reasons of security, referring to master (’xxxxx100’) is not allowed; in order to refer to a master PIN, it must be addressed as current PIN.

WR-AC write access condition
contains the AC for *WRITE KEY Update*

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								compare operation
0 0 1 1	0 1 0 1							= < ≥ ≠
								RFU
								reference value
			0 1					current state MF current state current
				#	#	#	#	compare value for reference value 0 = highest state 15 = lowest state

KFPC error counter

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								KFPC Init
								KFPC

❍ If initial value KFPC Init and error counter KFPC are set to ’FF’, the key fault presentation counter will not be active.

WR-Info Write Info

contains coded in b2/b1 the information whether or not *WRITE KEY* is permitted for this key record

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								write condition
0								not virgin
1								virgin
						0	0	write never; WRITE KEY forbidden generally
						0	1	write once; WRITE KEY permitted if virgin = 1 and WR-AC is fulfilled; after write operation, virgin = 0 is set thus allowing a write operation only once
						1	1	write multiple; WRITE KEY permitted if WR-AC is fulfilled
								RFU

Key-Attr. key attribute

specifies the intended purpose of the key

b8	b7	b6	b5	b4	b3	b2	b1	Definition
1								AKD exists
	1							SECURE DECREASE
		1						SECURE INCREASE
								RFU
				1				DES Authenticate
					1			DES Crypt
						1		PUK
							1	PIN

AKD Additional Key Description for DES Authenticate
defines authentication type

b8	b7	b6	b5	b4	b3	b2	b1	Definition
								RFU
								MUTUAL AUTHENTICATION
					1 0			allowed not allowed
								EXTERNAL AUTHENTICATION
						1 0		allowed not allowed
								INTERNAL AUTHENTICATION
							1 0	allowed not allowed

Response

SW1	SW2
’90’	’00’

Status Bytes

Code	Description
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 81’	key cannot be overwritten (write never or write once)
’69 85’	key is no PIN
’6A 00’	key not found
’6A 84’	insufficient memory
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

Card Completion

The production of a STARCOS S card ends with the completion. During this procedure, the card's operating system is activated and the basic structures are loaded in the chip EEPROM. This chapter gives an overview of the completion procedure and describes the commands allowed only during this phase.

Life Cycle

Life cycle periods

From the point of view of the operating system, the life cycle of a STARCOS S card comprises of the following steps after chip production:

– completion
  activate the card's operating system and load the basic structures in the chip EEPROM
– initialisation
  generate the file hierarchy of the operating system and define the keys used
– personalisation
  enter personal data of the card user and activate application(s)

❍ For detailed information on initialisation and personalisation see 'Sample Generation of a STARCOS S Card' from page 117 onwards.

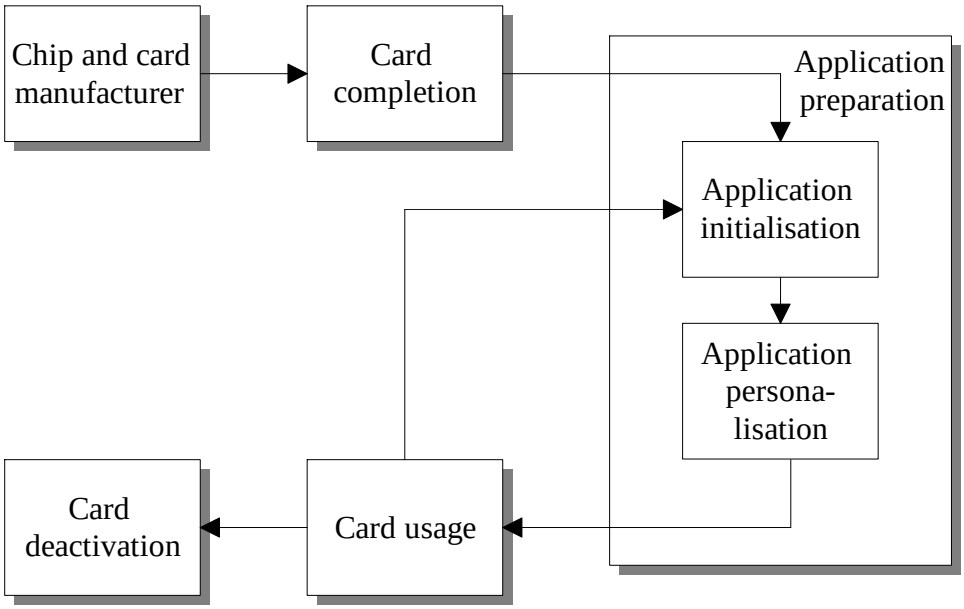


Fig. 30 Life cycle periods

Completion

Basically, the completion procedure generates files and structures for the operating system STARCOS S. Only the commands *CHECK KEY, LOAD COMPLETION DATA, COMPLETION END* and *GET CARD DATA* are available during the completion procedure to this end.
During completion, variable data such as

– initial value and key for random number generator
– key for CREATE MF
– ATR
– operating system version number
– customer-specific commands

will be loaded into the EEPROM.

Completion commands

The special completion commands *CHECK KEY, LOAD COMPLETION DATA, COMPLETION END* feature the same structure as the general commands of the operating system (see 'Structures' on page 43).

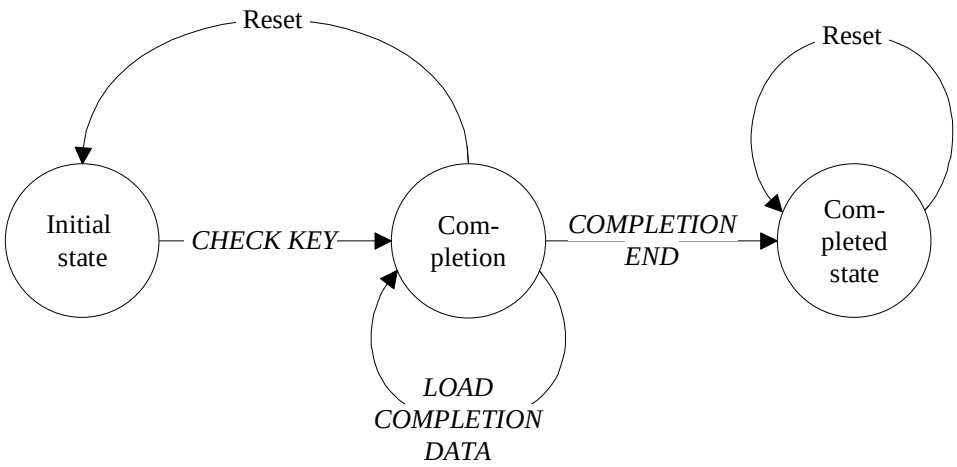


Fig. 31 Completion commands

Using keys

For security reasons, the completion procedure is secured by using keys with the command *CHECK KEY* (see page 106).

The chip manufacturer enters a key in the EEPROM for authenticating the completion. This EEPROM key is used to calculate a card AUTH-KEY. Two authentication types are available, depending on the EEPROM key:

– card-individual authentication
  If the lowest Bit of the EEPROM key is 0, the chip serial number (also entered by the chip manufacturer) will be enciphered with the AUTH-KEY. The resulting encipherment is AUTH-DATA, to be transmitted for authentication in the data component of the command *CHECK KEY*. In this way, AUTH-DATA is different for every card.

– batch authentication
  If the lowest Bit of the EEPROM key is 1, the AUTH-KEY is directly used as AUTH-DATA; the serial number will not be enciphered. AUTH-KEY and as a consequence also AUTH-DATA will only be altered, after the EEPROM key has been changed.

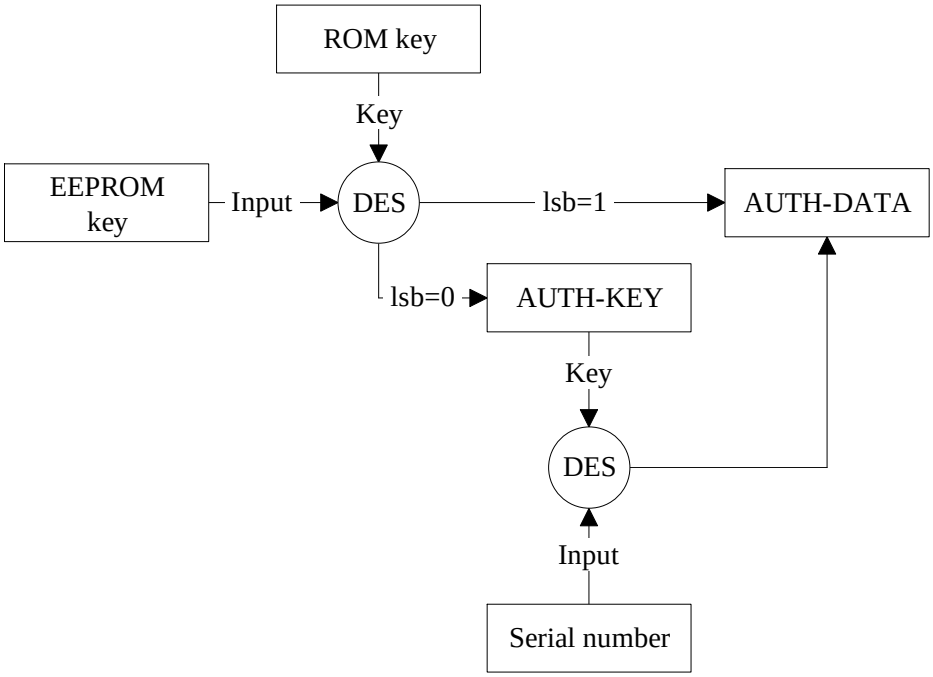


Fig. 32 Authentication keys

CHECK KEY

The command *CHECK KEY* is used for authentication of the completion system and the STARCOS S card.

During authentication, a key fault presentation counter with an initial value of 5 will be used. After correct authentication the key fault presentation counter will be reset to the initial value.

The command *CHECK KEY* is only allowed, if
– the card has not been completed yet
(i.e. *COMPLETION END* has not yet been executed)
– the KFPC has not expired.

Command

CLA	INS	P1	P2	L_c	DATA
’80’	’10’	’00’	’00’	’08’	AUTH-DATA

AUTH-DATA calculated authentication key (see page 105)

Response

SW1	SW2
’90’	’00’

Status Byte

Code	Description
’63 Cx’	incorrect key, x represents the number of retries
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’69 83’	key locked
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

LOAD COMPLETION DATA

The command *LOAD COMPLETION DATA* is only allowed after executing the command *CHECK KEY* first and is used to load operating system data in the chip EEPROM.
The parameters P1/P2 specify the starting position for the write operation; the parameter L_c specifies the number of bytes to be written.

Command

CLA	INS	P1	P2	L_c	DATA
’80’	’12’	HB	LB		EEPROM DATA

HB high byte of EEPROM address

LB low byte of EEPROM address

L_c 1-128 bytes

Response

SW1	SW2
’90’	’00’

Status Byte

Code	Description
’65 00’	write error
’67 00’	incorrect length
’69 00’	incorrect state
’6A 86’	parameters P1/P2 incorrect
’6D 00’	incorrect INS
’6E 00’	incorrect CLA
’6F 81’	system error

COMPLETION END

The command *COMPLETION END* is only allowed after executing the command *CHECK KEY* first and is used to terminate the completion of the STARCOS S operating system.
If the checksum transmitted corresponds to the one calculated internally, the card will be set to completed state.

Command

CLA	INS	P1	P2	L_c	DATA
’80’	’14’	’00’	’00’	’04’	length of EEP data || checksum

length of EEP data
number of bytes starting at ’80 20’ to be included in the checksum

checksum
checksum (AT&T)

Response

SW1	SW2
’90’	’00’

Status Byte

Code	Description
’65 00’	EDC error or write error
’67 00’	incorrect length
’69 00’	incorrect state
’6A 86’	parameters P1/P2 incorrect

Appendix

In the following you will find basic information on the transmission protocols used by STARCOS S for better understanding of the functions, technical data of the smartcard hardware as well as a sample generation of a STARCOS S card.

In addition, the appendix contains a glossary of terms and abbreviations used and the index.

Transmission Protocols

Two transmission protocols for smartcards have been established on an international level: the byte protocol T=0 and the block protocol T=1. Depending on the version, STARCOS S uses the protocol T=0, T=1 or a combination of both protocols.

Select transmission protocol

If both protocols are used, first protocol T=0 will be used after the ATR, in accordance with ISO/IEC 7816-3. Directly after the ATR, T=1 can be selected with PTS (Protocol Type Selection). Since the conversion factor of STARCOS S 1.2 is fixed to 372, the PTS sequence for switching to T=1 is ’FF 11 11 FF’.

❒ ***Other PTS sequences are not allowed with STARCOS S 1.2 and lead to an endless loop in accordance with ISO/IEC 7816-3 if used.***

Voltage Supply and Clock Frequency

The smartcard gets both its voltage supply and clock frequency from the terminal via the respective pins. The electric activation follows a defined sequence, called power-on sequence (see ISO/IEC 7816-3).

The clock frequency supplied determines the processor speed of the card. As the reception and transmission routines are implemented in the software, the clock frequency together with the internal conversion factor also determine the transmission rate. Further parameters like character waiting time (CWT), block guard time (BGT) and block waiting time (BWT) are in turn derived from the transmission rate.

Transmission Parameters

The following example calculation determines the STARCOS S parameters for a clock frequency of 3.571 MHz and an internal conversion factor of 372.

❒ ***All values used are only examples!***

Transmission rate

The transmission rate specifies the number of bits transmitted per second. As the clock frequency determines the processing time for a command cycle of the CPU, the interval for a bit transmission is calculated with an integer conversion factor from the clock frequency. The conversion factor must be sufficient for the CPU to perform reception and storage commands before the next bit follows.

$$\frac{\text{clock frequency}}{\text{conversion factor}} = \text{data transmission rate}$$

$$\frac{3571200\ \text{Hz}}{372} = 9600\ \frac{1}{\text{s}}$$

Elementary time unit

An elementary time unit (etu) is the interval necessary for the transmission of one bit. This interval is calculated from the transmission rate as follows:

$$\frac{1}{\text{bit}\ \frac{1}{\text{s}}} = 1\ \text{etu}$$

$$\frac{1}{9600\ \frac{1}{\text{s}}} = 104{,}1666\ \mu\text{s}$$

Start bit/parity bit/stop bit

For STARCOS S, every byte (8 data bits) is accompanied (according to ISO/IEC 7816-3) by one start bit, one parity bit and two stop bits, which make a total of 12 bits per character.

- Start bit
  serves for synchronisation of the communicating parties at the beginning of a character; both then have a defined start position for every character. Possible differences which may not influence a single character may thus not sum up and interfere with the transmission.
- Parity bit
  secures a character against transmission errors. As STARCOS S uses even parity, this bit is set by the data link layer (layer 2) in a way that the number of 1-bits (without start and stop bits) is even.
- Stop bit
  provides an interval at the end of a character for the receiver to process and store the byte received before the next character follows; STARCOS S uses 2 full stop bits for sending.

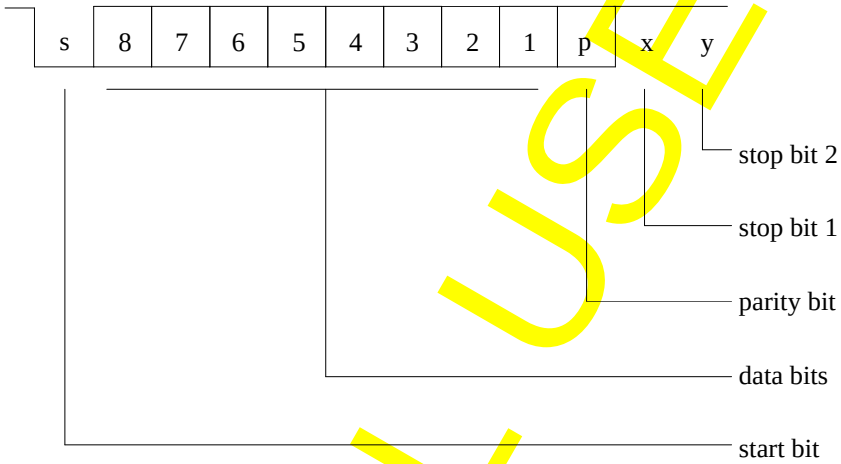


Fig. 33 Character structure according to ISO/IEC 7816-3

The stop bits are always high and the start bit is always low which means that the synchronisation may re-enter with the trailing edge of the start bit.

The transmission interval for one character depends on the number and duration of the bits to be transmitted; the interval for a character with 12 bits would be:

$$12 \times 104.166\ \mu s = 1.250\ ms$$

Character waiting time

Character waiting time (CWT) is the maximum interval between two characters transmitted (i.e. between two start bits). If this interval is exceeded, the receiver assumes that the transmission has been terminated.

The card informs the terminal in bit TB_3 of the ATR about the character waiting integer (CWI) value it uses to calculate the CWT:

$$CWT = 2^{CWI} + 11\ etu = 2^4 + 11 = 27\ etu$$

Block guard time/ block waiting time

Block guard time (BGT) and block waiting time (BWT) control the switching between transmitter and receiver.

BGT specifies the point of time when the receiver may begin transmitting. The default for the BGT is 22 etu. Optionally, the BGT may be set in a wide range.

BWT is the maximum interval for the terminal to wait for a response from the card. If this interval is exceeded without a proper waiting time extension (WTX), the terminal assumes a card problem.

The card informs the terminal in bit TB_3 of the ATR about the block waiting integer (BWI) value it uses to calculate the BWT:

$$BWT = 11\ etu + \frac{2^{BWI}}{10}\ s$$

$$BWT = 11\ etu + \frac{2^4}{10}\ s = 1{,}6\ s$$

Transmission Protocol T=0

According to ISO/IEC 7816-3, the data transmission on the physical layer with the byte protocol T=0 uses 1 start bit, 8 data bits, 1 parity bit and 2 stop bits (see page 112).

The receiver signals a possible transmission error with a low level on the I/O line; the transmitter then repeats the byte received incorrectly.

The protocol conversion factor is set to a fixed value of 372. The protocol T=0 categorises commands according to ISO/IEC 7816-4 in four different schemes, called CASES:

CASE 1
short format: no command data
no response data

CASE 2
short format: no command data
response data up to maximum buffer size

CASE 3
short format: command data up to maximum buffer size
no response data

CASE 4
short format: command data up to maximum buffer size
response data up to maximum buffer size

For CASE 4, the actual application command must be followed by a command GET RESPONSE, in accordance with ISO/IEC 7816-4. This command transmits the response data from card to terminal.

Transmission Protocol T=1

Whereas T=0 is a byte transmission protocol, T=1 is a block transmission protocol; i.e. the smallest unit to be transmitted and secured is a block with a minimum of 4 bytes. In the case of T=1, the actual information is combined with 4 additional bytes containing transmission-specific protocol information.

NAD	PCB	LEN	INFORMATION	XOR

Fig. 34 Block structure for transmission protocol T=1

The complete block is structured as follows:

– Node address byte (NAD)
  contains coded in the upper nibble (7-5) the transmitter and in the lower nibble (3-1) the receiver of a block.
  The card retrieves its "name" within the system from the first NAD-Byte following the ATR.
– Protocol control byte (PCB)
  serves for exchanging of control information such as chaining mode, request for repetition etc. between the communicating parties.
– Length byte (LEN)
  informs the receiver about the number of information bytes in the block; this partially serves for correct detection of the block end.
– INFORMATION
  contains the actual application information. Its length varies with the command; in the case of control information, this component may be omitted. The structure of the information is not defined in the T=1 protocol, but is part of the application protocol. This information component is also called APDU (Application Data Unit).
– Checksum (XOR)
  is a checksum covering all previous bytes.

As in the case of T=0, data transmission for T=1 occurs with 1 start bit, 8 data bits, 1 parity bit and 2 stop bits (see page 112) in accordance with ISO/IEC 7816-3.

The protocol conversion factor is set to a fixed value of 372.

Smartcard Hardware

STARCOS S 1.2 has been implemented on the single-chip microcomputer Siemens SLE 44C40S. This chip features the following device data:

Memory management

256 bytes RAM	working area
16 kbytes ROM	operating system
4 kbytes EEPROM	operating system, applications and data

– EEPROM properties
  10 years data retention; 100,000 write/erase cycles per EEPROM-page (1 page = 1 byte) guaranteed
– data formats
  1, 4 and 8 bits
– addressing and registers
  8 bits
– frequency
  maximum of 7.5 MHz (standard 1 through 5 MHz)
– power supply
  +5 V ± 10%, <10 mA/5 MHz, internal sleep mode ≤200 µA with clock, <100 µA without clock (standard 35 µA)
– module contacts complying with ISO/IEC 7816-2

Versions

The following STARCOS S 1.2 versions are available:

Version	Conv. Factor	Protocol		EEPROM	Notes
		T=0	T=1	SLE 44C40S	
1.2-1-1-0	372	✓	✓	3500 bytes	with PTS
1.2-1-2-0	372	✓		3500 bytes	without PTS
1.2-1-3-0	372		✓	3500 bytes	without PTS
1.2-1-5-0	372		✓	3700 bytes	without PTS, without enciphering commands
1.2-1-6-0	372	✓	✓	3700 bytes	with PTS, without enciphering commands
1.2-1-7-0	372	✓		3700 bytes	without PTS, without enciphering commands

Fig. 35 STARCOS S Versions

Sample Generation of a STARCOS S Card

The following example will illustrate how the STARCOS S card functions together with the application. During generation, the individual phases may be independent of time and location, each being protected with security mechanisms.

Generation

This phase comprises of the production of the STARCOS S card by G&D as well as its completion. The term completion refers to the loading of system-relevant references and additional information (required by STARCOS S for operation) into the card's EEPROM.

❍ Executing all of the following phases and the responsibility for them depends on agreements made between G&D, the card issuer and the service provider.

MF Definition

The MF structure plays and important role, since it contains the definition of rights for installing applications. The card issuer has priority as compared to the service provider, since he has the right to define MFs; thus the card issuer determines who installs which applications at what time.
Loading an application consists of registering and initialising. Therefore, MF definition may consist of a combination of the following processes:

– no further applications may be loaded:
  This is the case, if the card issuer installs several applications on one card and does not want any extensions in the future.
  The MF will then not allow the command *REGISTER*; to load the application in the first place, the card issuer must register and initialise the applications prior to terminating the MF initalisation with the command *CREATE End*.

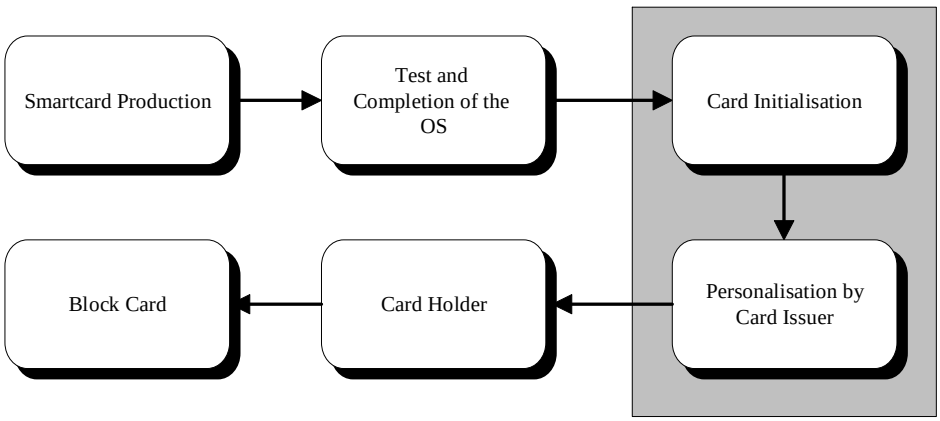


Fig. 36 Applications loaded by the card issuer

– Applications may be loaded as desired:
The MF now allows both the commands *REGISTER* and *CREATE*.
The card issuer might, of course, also have loaded applications on the card.

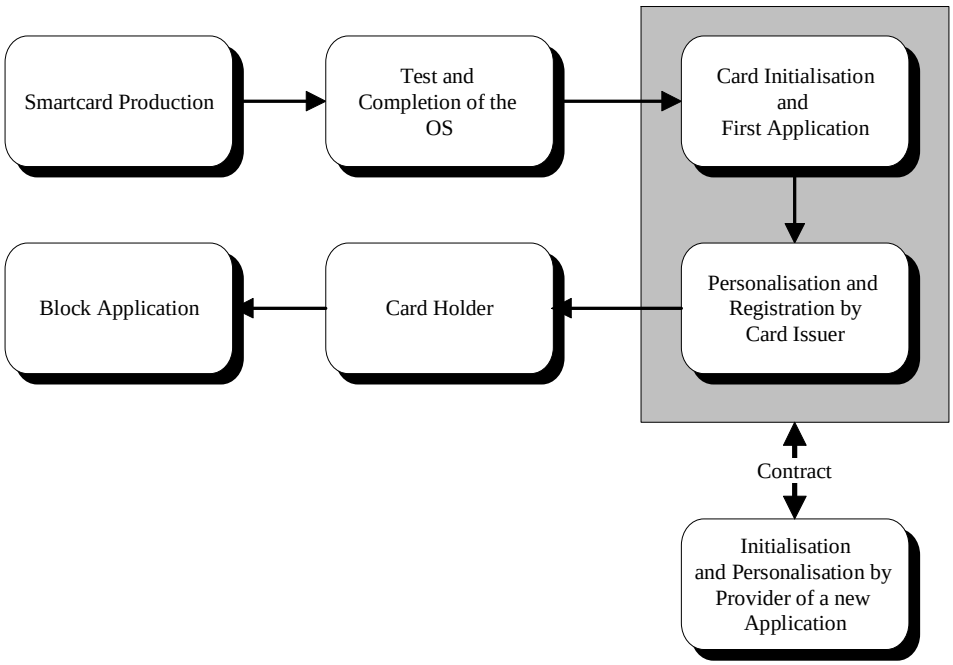


Fig. 37 Applications may be loaded as desired

Load applications

Registering and initialising an application (if required, together with MF definition) might follow one after the other or independent of time or location.

If the registration occurs independent of a DF initialisation, STARCOS S offers the possibility to secure the command *CREATE*. To do so, a state must be specified with an AC during MF generation which allows the command *CREATE DF*.

Command *REGISTER*

The command *REGISTER* is only required for parent DFs; the application name will be registered and the required memory on the card will be allocated. This memory area cannot be increased or decreased at a later stage.

Command *CREATE*

The command *CREATE* initialises the application (if required after successful authentication). After the initialisation has been terminated with the command *CREATE End*, the security structure of the application is activated, i.e. the ACs are verified by the operating system.

Abbreviations Used

AC	Access Condition	specifies the initial state, the compare operator and the reference to the current state (current or MF)
ACP	Access Control Parameter	authentication parameter
ACV	Access Control Value	indicates the access condition and the consecutive state
AID	Application Identifier	
AKD	Additional Key Description	
APDU	Application Data Unit	basic module of communication on layer 7
ATR	Answer To Reset	response after power-on sequence of card; contains card ID
BGT	Block Guard Time	point of time when the smartcard may begin transmission (default 22 etu)
BWT	Block Waiting Time	maximum interval for the terminal to wait for a response from the smartcard
CBC	Cipher Block Chaining	chained decipherment of several blocks with 8 bytes each
CLA	Class Byte	contains encoded command structure and transmission protection
CRP	Command Response Pair	
CST	Consecutive State	state accessed after correct execution of a command in the state machine
CWT	Character Waiting Time	maximum interval between two characters transmitted by the smartcard
DES	Data Encryption Standard	symmetric encipherment algorithm
DF	Dedicated File	structure underlying the MF (comparable to directories); may contain a single application
EDC	Error Detection Code	
EF	Elementary File	stores the actual application data
etu	Elementary Time Unit	
FCB	File Control Block	block with management information for each file type
FID	File Identifier	
FM	File Manager	controls data access
ICC	Integrated Chip Card	smartcard
IFD	Interface Device	general term for smartcard terminal
INS	Instruction Byte	contains instruction code of layer 7
ISF	Internal Secret File	special EF used for storing keys (PIN, PUK, DES)

KFPC	Key Fault Presentation Counter	error counter for authentication key
KID	Key Identifier	contains key number and position of ISF which holds the key
LEN	Length Byte	contains block length specification (T=1)
MAC	Message Authentication Code	value for secure messaging calculated cryptographically
MF	Master File	root directory of STARCOS S file system
NAD	Node Address Byte	contains coded transmitter and receiver of a block (T=1)
PCB	Protocol Control Byte	contains control information for communication
PIN	Personal Identification Number	
PTS	Protocol Type Select	switch transmission protocols
PUK	Personal Unblocking Key	special PIN for unblocking a PIN with expired KFPC
RND	Random Number	
RO	Read Only	
RSA	Rivest, Shamir, Adleman	cryptographic method; named after its inventors' last names
RW	Read/Write	
SSC	Send Sequence Counter	communication counter
SSD	Security Status Description	authentication parameter
SW	Status Word	application status byte
TM	Transmission Manager	controls data transmission
WO	Write Only	
WORM	Write Once Read Multiple	
WR	Write	
WTX	Waiting Time Extension	extends the maximum interval for the terminal to wait for a response from the smartcard for time-consuming functions

Index