AN 077520

ICODE SLRC400 Command Set

Rev. 2.0 — November 2009

Application note

Document information

Info	Content
Keywords	ICODE SLRC400 Command Set, User & Reference Manual, ICODE 1, ICODE SLI, ICODE SLI-S, ICODE SLI-L, ICODE EPC, ICODE UID
Abstract	This document describes the functionality of the command set for SLEV400 ICODE PEGODA Reader.

founded by Philips

Revision history

Rev	Date	Description
2.0	2009 November	ICODE EPC, UID and SLI-S/ L command set added, ICODE SLI updated
1.0	2001 November	First published version>

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. General Information

1.1 Scope

This document describes the functionality of the command set for SLEV400 ICODE PEGODA Reader. It includes the functional description of the used commands and gives details, how to use or design-in this device from a system and software viewpoint.

The default configuration for the SLEV400 uses the SLRC400 as the contactless reader IC.

1.2 General Description for ICODE

The ICODE PEGODA read/write device SLEV400 is ready to be connected to a PC.

Fig 1 shows the basic overview of the SLEV400´s software concept. Different levels of the PC libraries can be identified.

- **Application Level**

  This level is user specific and might be used by the user to implement own applications and test programs. The evaluation kit packages for the SLRC400 provide the *ICODE UniversalDemo* program to show the functionality of the SLRC400 with ICODE 1, ICODE SLI, ICODE SLI-S, ICODE SLI-L, ICODE UID and ICODE EPC labels.

- **SLEV400 Command Set**

  This document describes the library giving the user the possibility to adapt the application to the ICODE PEGODA reader. All necessary settings and command are explained in detail in that document.

- **HostRDCom**

  This document describes the communication between the SL EV400 and a host PC.

The supported operating systems are limited to the Microsoft Windows Platform. Depending on the serial communication over the USB Win2000/XP are supported. The content of this document should be precise enough, to give the user the possibility writing own communication libraries for other operating systems.

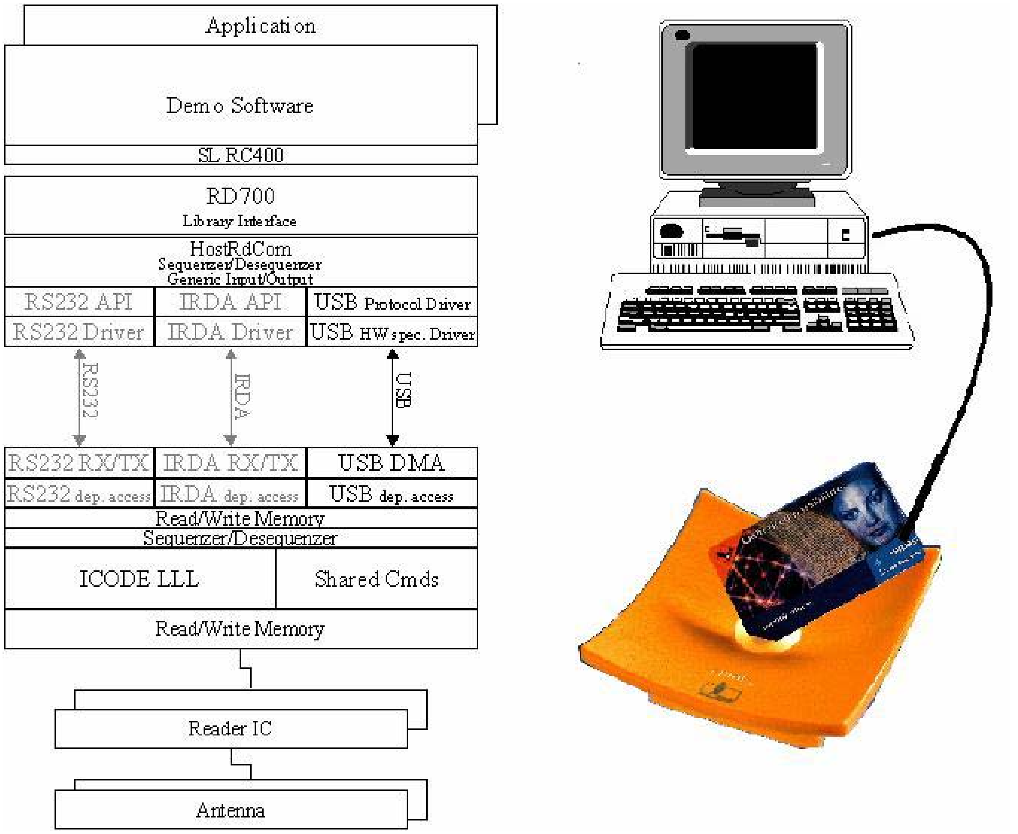

Fig 1. ICODE PEGODA general software overview

2. SLEV400 Command Set

2.1 General Description of Serial Communication

The SLEV400 reader can only be connected via serial data interfaces. The default configuration offers a USB connection. Additionally, the command set includes additionally RS232 and IrDA interface to a host.

The serial data stream consist depend from the selected interface type- and transfer data.

- Frame data depend on the selected interface
- Transfer data depend only on the selected command

To explain this dependency, the expected serial transfer data stream is described at command level. From a reader point of view the transfer data consists of an IN-transfer and an OUT-transfer.

- IN-transfer data is sent from host to the reader module
- OUT-transfer data is sent from the reader module to the host

The processing status of a command (mostly the return value) is part of the framing data and therefore not described at function level.

Each function is described with corresponding function prototype and stream data composition. The number of bytes occupied by this parameter is written in brackets.

Multiple byte parameters are converted to the serial byte stream with the least significant byte first.

Example:

short value 0x0A05

is converted to

data[x] = 0x05

data[x+1] = 0x0A

long value 0x04030201

is converted to

data[x] = 0x01

data[x+1] = 0x02

data[x+2] = 0x03

data[x+3] = 0x04

NOTE:

Pay attention, that the order of the parameter variables within the data stream may be different to the order in the function prototype. The order of parameters in the function prototype is given by the logical matching of the parameters. The data-stream's order is given by data direction and data length. A word -aligned access to multiple byte parameters are possible.

2.2 SLRC400 Interface Wrappers

The SLRC400 library is a wrapper library over Rd700 and HostRdCom. In order to provide a simpler (but less flexible) interface handling the library introduces two new functions.

Table 1. SLRC400 Interface wrappers

Function name	Function call
Sl400InterfaceOpen	*Signed char Sl400InterfaceOpen (unsigned long mode, unsigned long options)*
Sl400InterfaceClose	*signed char Sl400InterfaceClose (void)*

2.2.1 SL400InterfaceOpen

```
signed char Sl400InterfaceOpen (unsigned long mode,
                                unsigned long options)
```

Parameters:

mode (IN) 4 bytes interface type description
0x30 USB
0x40 RS232
0x50 IrDA

options (IN) 4 bytes interface options
depending on the interface type, this parameter is used to specified additional parameters.
For USB and IrDA devices, this parameter is ignored.
For RS232 devices the COM-pot can be specified e.g. 1 for COM1 or 2 for COM2.

Returns:

MI_OK

This function uses the HostRdCom interface to open a connection to the reader and use this handle for following function calls of this library. Nearly all functions of the Rd700 library are equipped with a new interface, where this handle is used.

2.2.2 SL400InterfaceClose

```
signed char Sl400InterfaceClose (void)
```

Parameters: *none*

Returns:

MI_OK

This function corresponds to the *Sl400InterfaceOpen* function. Each time, the used interface should be released; this function has to be called.

3. Modules

The MFRD700 command set contains of several modules covering different functionality:

Table 2. Modules

Module	Description
Administration command set	Several commands for reader IC administration and configuration
ICODE 1 command set	ICODE 1 commands
ICODE EPC command set	ICODE EPC commands
ICODE UID command set	ICODE UID commands
ICODE SLI and ISO15693 specific commands	ICODE SLI and/or ISO15693 commands
ICODE SLI-S command set	ICODE SLI-S commands

3.1 Administration Command Set

The administration command set covers several commands for reader IC administration and configuration.

3.1.1 Included Functions

Table 3. Administration commands

Function name	Function call
I1PcdRfReset	*signed char I1PcdRfReset (unsigned short ms)*
I1PcdConfig	*signed char I1PcdConfig (void)*
I1SetBitPhase	*signed char I1SetbitPhase (unsigned char bitphase)*

NOTE: In case of an error, the appropriate error code is set. Nevertheless, all received data are returned. This feature helps to debug the errors. Even if all data seems to be received correctly (data is filled up with reasonable values), a CRC, parity or other error could be reported.

3.1.2 Function Description

3.1.2.1 I1PcdRfReset

```
signed char I1PcdRfReset (unsigned short ms)
    IN      ms(2)
    OUT
```

Parameters:

ms (IN) time period in milliseconds. Defines the switch off time of the reader IC's RF-field in milliseconds.

Returns:

MI_OK always

This function turns off the RF-field for a specified time in milliseconds by setting the variable *ms*. Elapsing this time the RF-field is turned on approximately 1 millisecond later. If time variable *ms* is set to 0, the RF-field is turned off.

3.1.2.2 I1PcdConfig

```
signed char I1PcdConfig (void)
        IN
        OUT
```

Parameters: none

Returns:

MI_OK always

This function has to be called before the first data is written to the SLRC400 in order to perform the internal configuration. A reset of the reader IC is done and several registers are set.

3.1.2.3 I1SetBitPhase

```
signed char I1SetBitPhase (unsigned char bitphase)
        IN      bitphase(1)
        OUT
```

Parameters:

bitphase (IN) value of the bitphase

Returns:

MI_OK always

This function writes a given value to the bitphase register of the SLRC400.

3.1.2.4 SL400LibInfo

```
const char* Sl400LibInfo (unsigned long key)
        IN      key(8)
        OUT
```

Parameters:

key (IN) specifies the information to return

This function is used to get the version of the library.

key may have two values:

- `SL400_VERSION (0x0C)`
  function returns a char* to the version.
- `SL400_BUILD   (0x0D)`
  function returns an unsigned long int.

3.2 ICODE 1 Command Set

Labels of the ICODE 1 family support a defined set of instructions. The SLRC400 fully supports communication with these labels.

For further information on the labels command set please refer to the according product description of the ICODE 1 IC.

3.2.1 Included Functions

Table 4. ICODE 1 command set

Function name	Function call
I1init_StdMode	*signed char I1init_StdMode (void)*
I1init_FastMode	*signed char I1init_FastMode (void)*
I1output_read	*signed char I1output_read (unsigned char tse_hash,* *unsigned char blnr,* *unsigned char nobl,* *unsigned char unsel,* *unsigned short *len,* *unsigned char *data)*
I1output_anticoll_select	*signed char I1output_antcoll_select (unsigned char tse_hash,* *unsigned short *len,* *unsigned char *data)*
I1output_write	*signed char I1output_write (unsigned char hash,* *unsigned char blnr,* *unsigned char *wr_data,* *unsigned char *wrts,* *unsigned short *len,* *unsigned char *data)*
I1output_halt	*signed char I1output_halt (unsigned char hash,* *unsigned char *hts,* *unsigned short *len,* *unsigned char *data)*
I1output_eas	*signed char I1output_eas (unsigned short *len,* *unsigned char *data)*
I1calc_tse_hash	*singed char I1calc_tse_hash (int ts,* *unsigned char hash,* *unsigned char *tse_hash)*
I1_reset_quiet_bit	*signed char I1_reset_quiet_bit (void)*

3.2.2 Function Description ICODE 1

3.2.2.1 I1init_StdMode

```
signed char I1init_StdMode (void)
      IN
      OUT
```

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* to initialize this mode or during program execution to switch from ICODE 1 “Fast Mode” to the ICODE 1 “Standard Mode”.

3.2.2.2 I1init_FastMode

```
singed char I1init_FastMode (void)
      IN
      OUT
```

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* to initialize this mode or during program execution to switch from ICODE 1 “Standard Mode” to the ICODE 1 “Fast Mode”.

3.2.2.3 I1calc_tse_hash

```
signed char I1calc_tse_hash (int ts,
                             unsigned char hash,
                             unsigned char *tse_hash)
      IN     ts(1), hash(1)
      OUT    tse_hash(1)
```

Parameters:

ts	(IN) timeslot information, possible values 1, 4, 8, 16, 32, 64, 128 and 256.
hash	(IN) the starting bit position within the serial number which should the label take to calculate it’s timeslot.
tse_hash	(OUT) combination of *ts* and *hash*.

Returns:

MI_OK

The value *tse_hash* is used within different commands. It is possible to use this command or to calculate the value of *tse_hash* at the PC side with the same algorithm which is used in the low level library.

3.2.2.4 I1output_readI1

```
signed char I1output_read (unsigned char tse_hash,
                           unsigned char blnr,
                           unsigned char nobl,
                           unsigned char unsel,
                           unsigned short *len,
                           unsigned char *data)
      IN     tse_hash(1), blnr(1), nobl(1), unsel(1)
      OUT    data(len)
```

Parameters:

tse_hash (IN) combination of *ts* and *hash* which was calculated with *I1calc_tse_hash*.

blnr (IN) number of the first block within the memory of the label which should be influenced by the command.

nobl (IN) number of blocks that should be influenced by the command starting with the number of the first block.

unsel (IN) 0 -> selected read (first the command “*I1output_anticoll_select*” is necessary); 1 -> unselected read.

len (OUT) for character arrays an additional length information is necessary.

data (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the byte are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function reads out data from the label starting with the specified block address (*blnr*).

Depending on the number of chosen timeslots (*ts*), the response “DATA” includes *ts* byte information about the status of each timeslot (as described in the data sheet for the ICODE 1 IC), followed by the data of each label at it’s timeslot.

e.g.: If there is one label within the RF field, the number of timeslots is 4 and the label responses within the second timeslot the response “*DATA*” would look like this:

Byte	Byte	Byte	Byte	nobl *4 Byte	nobl * 4 Byte	nobl *4 Byte	nobl *4 Byte
[HEX]	[HEX]	[HEX]	[HEX]	[HEX]	[HEX]	[HEX]	[HEX]
0x01	0x00	0x01	0x01	0x00	Label response	0x00	0x00

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x05	I1_COUNTERR
0x08	I1_WEAK_COLLISION

Remark:

The read command can be executed as selected or unselected read. For the selected read the command *I1output_anticoll_select* has to be done first.

3.2.2.5 I1output_anticoll_select

```
signed char I1output_anticoll_select (unsigned char tse_hash,
                                      unsigned short *len,
                                      unsigned char *data)
     IN      tse_hash(1)
     OUT     data(len)
```

Parameters:

tse_hash	(IN) combination of *ts* and *hash* which was calculated with *I1calc_tse_hash*.
len	(OUT) for character arrays an additional length information is necessary.
data	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the byte are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function execution the anticollison select routine as it described in the data sheet for the ICODE 1 IC.

Depending on the number of chosen timeslots (*ts*), the response “DATA” includes *ts* * 4 bytes information or the serial number of the label.

e.g.: If there are 4 labels within the RF field, the number of timeslots is 4 and 1 label response within the second timeslot and all other responses within the third timeslot the response “DATA” would look like this:

4 Byte	4 Byte	4 Byte	4 Byte
[HEX]	[HEX]	[HEX]	[HEX]
4 * 0x00	Serial number	0x03 + 3 * 0x00	4 * 0x00

To get the other serial numbers it is necessary to repeat the *I1output_anticoll_select* command until all serial numbers are known or all timeslots are occupied.

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x04	I1_SNRERR
0x05	I1_COUNTERR
0x06	I1_TSOCC
0x08	I1_WEAK_COLLISION

3.2.2.6 I1output_write

```
signed char I1output_write (unsigned char hash,
                            unsigned char blnr,
                            unsigned char *wr_data,
                            unsigned char *wrts,
                            unsigned short *len,
                            unsigned char *data)
```

```
IN   hash(1), hts(4)
OUT  data(len)
```

Parameters:

hash (IN) the starting bit position within the serial number which should the label take to calculate it's timeslot.

blnr (IN) number of the first block within the memory of the label which should be influenced by the command.

wr_data (IN) array of 4 byte data which should be write to the label beginning with block *blnr*.

wrts (IN) array of 4 byte which includes the timeslot information, at which labels the data should be written.

len (OUT) for character arrays an additional length information is necessary.

data (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the byte are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function execution the write routine as it described in the data sheet for the ICODE 1 IC.

Before executing this command it is necessary to execute the *I1output_anticoll_select*, after this command each detected label is fixed to its timeslot. With *wrts* it is possible to mask all timeslot at which the *write* command should be executed.

Depending on the number chosen timeslots (*ts*), the response “DATA” includes *ts* bytes information (status about the timeslot).

e.g.: If there are 4 labels within the RF field, the number of timeslots is 4, each label is fixed to one timeslot (with *I1output_anticoll_select*) and *wrts* = {0x01, 0x00, 0x00, 0x00} only the label within the first timeslot would be written. The response “DATA” would like this:

Byte	Byte	Byte	Byte
[HEX]	[HEX]	[HEX]	[HEX]
0x16	0x00	0x16	0x16

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x04	I1_SNRERR
0x05	I1_COUNTERR
0x06	I1_TSOCC
0x08	I1_WEAK_COLLISION
0x10	I1_NO_WRITE_OK

3.2.2.7 I1output_halt

```
signed char I1output_halt (unsigned char hash,
                           unsigned char *hts,
                           unsigned short *len,
                           unsigned char *data)
      IN   hash(1), hts(4)
      OUT  data(len)
```

Parameters:

hash	(IN) the starting bit position within the serial number which should the label take to calculate it’s timeslot.
hts	(IN) array of 4 byte which includes the timeslot information, which labels should be set into HALT mode.
len	(OUT) for character arrays an additional length information necessary.
data	(OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the byte are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function execution the halt routine as it described in the data sheet for the ICODE 1 IC.

Before executing this command it is necessary to execute the *I1output_anticoll_select*, after this command each detected label is fixed to its timeslot. With *hts* it is possible to mask all timeslot at which the *halt* command should be executed.

Depending on the number chosen timeslots (*ts*), the response “DATA” includes *ts* bytes information (status about the timeslot).

e.g. If there are 4 labels within the RF field, the number of timeslots is 4, each label is fixed to one timeslot (with *I1output_anticoll_select*) and *hts* = {0x01, 0x00, 0x00, 0x00} only the label within the first timeslot would be set to halt. The response “DATA” would look like this:

Byte	Byte	Byte	Byte
[HEX]	[HEX]	[HEX]	[HEX]
0x16	0x00	0x16	0x16

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x04	I1_SNRERR
0x05	I1_COUNTERR
0x06	I1_TSOCC
0x08	I1_WEAK_COLLISION
0x20	I1_NO_HALT_OK

3.2.2.8 I1output_eas

```
signed char I1output_eas (unsigned short *len,
                          unsigned char *data)
     IN
     OUT data(len)
```

Parameters:

len (OUT) for character arrays an additional length information is necessary.

data (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function execution the EAS routine as it is described in the data sheet for the ICODE 1 IC.

The response “DATA” includes 1 byte information of the EAS value and it look like this:

Byte
[HEX]
EAS Value

The values between 8 and 255 could occur.

3.2.2.9 I1_reset_quiet_bit

```
signed char I1_reset_quiet_bit (void)
     IN
     OUT
```

Parameters: none

Returns:

MI_OK
I1_TIMEOUT

If a label is in the QUIET mode this function will clears the *quiet bits* on that label. More information about *quiet bit* is given in the data sheet for the ICODE 1 IC.

3.3 ICODE EPC Command Set

Labels of the ICODE EPC family support a defined set of instructions. The SLRC400 fully supports communication with these labels.

For further information on the labels command set please refer to the according product description of the ICODE EPC IC.

3.3.1 Included Functions

Table 5. ICODE EPC command set

Function name	Function call
EPCPcdConfig	*signed char EPCPcdConfig (void)*
EPCBeginRound	*signed char EPCBeginRound (unsigned char *mask, unsigned char masklength, unsigned char nbrslots, unsigned char hash, unsigned short *resplen, unsigned char *resp)*
EPCWrite	*signed char EPCWrite (unsigned char blnr, unsigned char data)*
EPCDestroy	*signed char EPCDestroy (unsigned char *epc, unsigned char *destroy_code)*

3.3.2 Function Description ICODE EPC

3.3.2.1 EPCPcdConfig

```
signed char EPCPcdConfig (void)
```

IN

OUT

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* in order to enable the reader’s EPC mode.

3.3.2.2 EPCBeginRound

```
signed char EPCBeginRound (unsigned char *mask,
                           unsigned char masklength,
                           unsigned char nbrslots,
                           unsigned char hash,
                           unsigned short *resplen,
                           unsigned char *resp)
     IN   mask(masklength), masklength(1), nbrslots(1), hash(1)
     OUT  resp(resplen)
```

Parameters:

mask	(IN) fractional or complete EPC for selecting certain labels.
masklength	(IN) length of the mask in bits.
nbrslots	(IN) coded number of timeslots as described in the EPC IC specification.
hash	(IN) used to generate a timeslot, maximal value 0x1F.
resplen	(OUT) amount of bytes which were written to the response-buffer.
resp	(OUT) buffer for received data.

Returns:

MI_OK

This function executes the BeginRound routine as it is described in the data sheet for the ICODE EPC IC.

Depending on the number of chosen timeslots (*ts*), the response *resp* includes 1 + *ts* bytes of data in the case of no label within the field.

For every successful recognized label 14 bytes (12 bytes EPC and 2 bytes CRC) are added after its corresponding timeslot-byte.

e.g.: The response in the case of 4 timeslots, one label in the field, answering in timeslot no. 2:

1 Byte	1 Byte	1 Byte	1 Byte	14 Byte	1 Byte
[HEX]	[HEX]	[HEX]	[HEX]	[HEX]	[HEX]
0x01	0x01	0x01	0x00	14 * 0xXY	0x01
fix slot	slots no. 0	slots no. 1	label found in slot 2	EPC and CRC	slot no. 3

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x04	I1_SNRERR
0x05	I1_COUNTERR
0x06	I1_TSOCC
0x08	I1_WEAK_COLLISION

3.3.2.3 EPCWrite

```
signed char EPCWrite (unsigned char blnr,
                      unsigned char data)
     IN   blnr(1), data(1)
     OUT
```

Parameters:

blnr	(IN) block number to which the data should be written
data	(IN) data to write

Returns:

MI_OK

This function executes the write routine as it described in the data sheet for the ICODE EPC IC.

There is no response from the label after sending this command. In order to verify the written data, a *BeginRound* should be performed.

3.3.2.4 EPCDestroy

```
signed char EPCDestroy (unsigned char *epc,
                        unsigned char *destroy_code)
     IN   epc(12), destroy_code(3)
     OUT
```

Parameters:

epc	(IN) EPC from the label which is to be destroy.
destroy_code	(IN) destroy code, which was formerly written to the label.

Returns:

MI_OK

This function executes the destroy routine as it is described in the data sheet for the ICODE EPC IC.

There is no response from the label after sending this command. In order to verify that the label is destroyed, a *BeginRound* should be performed.

3.4 ICODE UID and ICODE UID-OTP Command Set

Labels of the ICODE UID and ICODE UID-OTP family support a defined set of instructions. The SLRC400 fully supports communication with these labels.

For further information on the labels command set please refer to the according product description of the ICODE UID IC and the ICODE UID-OTP IC.

3.4.1 Included Functions

Table 6. ICODE UID and ICODE UID-OTP command set

Function name	Function call
UIDPcdConfig	*signed char UIDPcdConfig (void)*
UIDBeginRound	*signed char UIDBeginRound (unsigned char *mask,* *unsigned char masklength,* *unsigned char nbrslots,* *unsigned short *resplen,* *unsigned char *resp)*
UIDWrite	*signed char UIDWrite (unsigned char blnr,* *unsigned char data)*
UIDDestroy	*signed char UIDDestroy (unsigned char *idd,* *unsigned char *destroy_code)*

3.4.2 Function Description ICODE UID and ICODE UID-OTP

3.4.2.1 UIDPcdConfig

```
signed char UIDPcdConfig (void)
```

IN

OUT

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* in order to enable the reader's UID mode.

3.4.2.2 UIDBeginRound

```
signed char UIDBeginRound (unsigned char *mask,
                           unsigned char masklength,
                           unsigned char nbrslots,
                           unsigned short *resplen,
                           unsigned char *resp)
     IN   mask(masklength), masklength(1), nbrslots(1)
     OUT  resp(resplen)
```

Parameters:

mask	(IN) fractional or complete *IDD* for selecting certain labels.
masklength	(IN) length of the mask
nbrslots	(IN) coded number of timeslots as described in the UID IC specification
resplen	(OUT) amount of bytes which were written to the response-buffer.
resp	(OUT) buffer for received data.

Returns:

MI_OK

This function executes the BeginRound routine as it is described in the data sheet for the ICODE UID IC and the ICODE UID-OTP IC.

Depending on the number of chosen timeslots (*ts*), the response *resp* includes 1 + *ts* bytes of data in the case of no label within the field.

For every successful recognized label 21 bytes (12 bytes user data (UD), 2 bytes UD-CRC, 5 bytes UID and 2 bytes UID-CRC) are added after its corresponding timeslot-byte.

e.g.: The response in the case of 4 timeslots, one label in the field, answering in timeslot no. 2.

1 Byte	1 Byte	1 Byte	1 Byte	14 Byte	1 Byte
[HEX]	[HEX]	[HEX]	[HEX]	[HEX]	[HEX]
0x01	0x01	0x01	0x00	14 * 0xXY	0x01
fix slot	slots no. 0	slots no. 1	label found in slot 2	UD, UD-CRC, UID, UID-CRC	slot no. 3

The following status could occur:

0x00	I1_OK
0x01	I1_NO_TAG
0x02	I1_CRCERR
0x03	I1_COLLERR
0x04	I1_SNRERR
0x05	I1_COUNTERR
0x06	I1_TSOCC
0x08	I1_WEAK_COLLISION

3.4.2.3 UIDWrite

```
signed char UIDWrite (unsigned char blnr,
                      unsigned char data)
     IN   blnr(1), data(1)
     OUT
```

Parameters:

blnr (IN) blocknumber to which the data should be written
data (IN) data to write.

Returns:

MI_OK

This function executes the Write routine as described in the data sheet for the ICODE UID IC and ICODE UID-OTP IC.

There is no response from the label after sending this command. In order to verify the written data, a *BeginRound* should be performed.

3.4.2.4 UIDDestroy

```
signed char UIDDestroy (unsigned char *idd,
                        unsigned char *destroy_code)
     IN   idd(19), destroy_code(3)
     OUT
```

Parameters:

idd (IN) the whole identifier data (IDD) from the label which is to be destroy.
destroy_code (IN) destroy code, which as formerly written to the label.

Returns:

MI_OK

This function executes the Destroy routine as it is described in the data sheet for the ICODE UID IC and the ICODE UID-OTP IC.

There is no response from the label after sending this command. In order to verify that the label is destroyed, a *BeginRound* should be performed.

3.5 ICODE SLI and ISO15693 specific Command Set

Labels of the ICODE SLI family support a defined set of instruction. The SLRC400 fully supports communication with these labels.

For further information of the labels command set please refer to the according product description of the ICODE SLI IC.

3.5.1 Included Functions

Table 7. ICODE SLI and ISO15693 command set

Function name	Function call
I2init_StdMode_15693	*signed char I2init_StdMode_15693 (void)*
I2init_FastMode_15693	*signed char I2init_FastMode_15693 (void)*
ISO15693_Read_sm	*signed char ISO156936_Read_sm (unsigned char flags,* *unsigned char *uid,* *unsigned char blnr,* *unsigned char nbl,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Inventory	*signed char ISO15693_Inventory (unsigned char flags,* *unsigned char AFI,* *unsigned char masklength,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15963_Write_sm	*signed char ISO15693_Write_sm (unsigned char flags,* *unsigned char *uid,* *unsigned char blnr,* *unsigned char nbl,* *unsigned char *data,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Stay_Quiet	*signed char ISO15693_Stay_Quiet (unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Lock_Block	*signed char ISO15693_Lock_Block (unsigned char flags,* *unsigned char *uid,* *unsigned char blnr,* *unsigned short *resplen,* *unsigned char *resp)*

Function name	Function call
ISO15693_Select	*signed char ISO15693_Select (unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15963_Reset_To_Ready	*signed char ISO15693_Reset_To_Ready* *(unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Write_AFI	*signed char ISO15693_Write_AFI (unsigned char flags,* *unsigned char *uid,* *unsigned char AFI,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Lock_AFI	*signed char ISO15693_Lock_AFI (unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Write_DSFID	*signed char ISO15693_Write_DSFID (unsigned char flags,* *unsigned char *uid,* *unsigned char DSFID,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Lock_DSFID	*signed char ISO15693_Lock_DSFID (unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Get_System_ Information	*signed char ISO15693_Get_Sytsem_Information* *(unsigned char flags,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Get_Multiple_Block_ Security	*signed char ISO_Get_Multiple_Block_Security* *(unsigned char flags,* *unsigned char *uid,* *unsigned char blnr,* *unsigned char nbl,* *unsigned short *resplen,* *unsigned char *resp)*

Function name	Function call
ISO15693_Inventory_Read	*signed char ISO15693_Inventory_Read* *(unsigned char flags,* *unsigned char ManCode,* *unsigned char AFI,* *unsigned char masklength,* *unsigned char *uid,* *unsigned char blnr,* *unsigned char nbl,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15963_Fast_Inverntory_Read	*signed char ISO15693_Fast_Inventory_Read* *(unsigned char flags,* *unsigned char ManCode,* *unsigned char AFI,* *unsigned char masklength,* *unsigned char *uid,* *unsigned char blnr,* *unsigned char nbl,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Set_Eas	*signed char ISO15693_Set_Eas (unsigned char flags,* *unsigned char ManCode,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Reset_Eas	*signed char ISO15693_Reset_Eas (unsigned char flags,* *unsigned char ManCode,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*
ISO15693_Lock_Eas	*signed char ISO15693_Lock_Eas (unsigned char flags,* *unsigned char ManCode,* *unsigned char *uid,* *unsigned short *resplen,* *unsigned char *resp)*

Function name	Function call
ISO15693_Eas_Alarm	*signed char ISO15693_Eas_Alarm (unsigned char flags,* *unsigned char ManCode,* *unsigned char *uid,* *unsigned char bEAS_ID_MaskLength,* *unsigned char *pbEAS_ID,* *unsigned short *resplen,* *unsigned char *resp)*

3.5.2 Function Description ICODE SLI

3.5.2.1 I2init_StdMode_15693

```
signed char I2init_StdMode_15693 (void)
        IN
        OUT
```

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* to initialize this mode or during program execution to switch from *ISO15693 Fast*, *ICODE 1 Standard* or *ICODE 1 Fast* to the *ISO15693 Standard Mode*.

3.5.2.2 I2init_FastMode_15693

```
signed char I2init_FastMode_15693 (void)
        IN
        OUT
```

Parameters: none

Returns:

MI_OK

This function has to be called after *I1PcdConfig* to initialize this mode or during program execution to switch from *ISO15693 Standard*, *ICODE 1 Standard* or *ICODE 1 Fast* to the *ISO15693 Fast Mode*.

3.5.2.3 ISO15693_Read_sm

```
signed char ISO15693_Read_sm (unsigned char flags,
                              unsigned char *uid,
                              unsigned char blnr,
                              unsigned char nbl,
                              unsigned short *resplen,
                              unsigned char *resp)
    IN   flags(1), uid(8), blnr(1), nbl(1)
    OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
blnr (IN) number of the first block within the memory of the label which should be influenced by the command.
nbl (IN) number of blocks that should be influenced by the command starting with the number of the first block.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the byte are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is *I1_OK* or *I2_NO_ERR* the followed bytes are the response of the label as described in ISO15693.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.4 ISO15693_Inventory

```
signed ISO15693_Inventory (unsigned char flags,
                           unsigned char AFI,
                           unsigned char masklength,
                           unsigned char *uid,
                           unsigned short *resplen,
                           unsigned char *resp)
      IN   flags(1), AFI(1), masklength(1), uid(8)
      OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
AFI (IN) application family identifier
masklength (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

If the *inventory* command is used for 1 timeslot the response has the following format:

If status is "*I1_OK*", "*I2_NO_ERR*" or "*I1_COLLERR*"

1st Byte	2nd Byte	2 Bytes	8 Bytes
Status	Collision position	Flags & DSFID	Part or whole UID

Else the response includes only the status!

If the 16 timeslots are used the response is a combination of the above described values. If one or more labels answer within a timeslot the response for this timeslot includes 13 byte, if an error occur within a timeslot the response includes 1 byte.

e.g.: 1st timeslot is free, at the second is a collision, at the third no error occur only one label answer and all others are free.

The response looks like:

1st TS	2nd TS				3rd TS				4th TS		16th TS
0x01	0x03	0x11	Flags & DSFID	Part of UID	0x00	0x00	Flags & DSFID	Part of UID	0x01	…	0x01

Remark:

The collision position includes 16 bit for the "*Flags & DSFID*" and 1 bit for the collision!

Consider that, it is possible to use the value of "collision position" as value for masklength.

3.5.2.5 ISO15693_Write_sm

```
signed ISO15693_Write_sm (unsigned char flags,
                          unsigned char *uid,
                          unsigned char blnr,
                          unsigned char nbl,
                          unsigned char *data,
                          unsigned short *resplen,
                          unsigned char *resp)
     IN   flags(1), uid(8), blnr(1), nbl(1), data(4)
     OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
blnr (IN) number of the first block within the memory of the label which should be influenced by the command.
nbl (IN) number of blocks that should be influenced by the command starting with the number of the first block. Must be set to 1 for ICODE SLI.
data (IN) array of 4 byte data which should be write to the label beginning with block *blnr*.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.6 ISO15693_Stay_Quiet

```
signed ISO15693_Stay_Quiet (unsigned char flags,
                            unsigned char *uid,
                            unsigned short *resplen,
                            unsigned char *resp)
     IN   flags(1), uid(8)
     OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
resplen *(*OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the second byte is the byte for the collision position, the followed bytes are the response of the label as described in ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.7 ISO15693_Lock_Block

```
signed ISO15693_Lock_Block (unsigned char flags,
                            unsigned char *uid,
                            unsigned char blnr,
                            unsigned short *resplen,
                            unsigned char *resp)
    IN   flags(1), uid(8), blnr(1)
    OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.

uid (IN) unique identifier of the IC.

blnr (IN) number of the first block within the memory of the label which should be influenced by the command.

resplen (OUT) for character arrays an additional length information is necessary.

resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.8 ISO15693_Select

```
signed ISO15693_Select (unsigned char flags,
                        unsigned char *uid,
                        unsigned short *resplen,
                        unsigned char *resp)
     IN   flags(1), uid(8)
     OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.9 ISO15693_Reset_To_Ready

```
signed ISO15693_Reset_To_Ready (unsigned char flags,
                                unsigned char *uid,
                                unsigned short *resplen,
                                unsigned char *resp)
     IN   flags(1), uid(8)
     OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.10 ISO15693_Write_AFI

```
signed ISO15693_Write_AFI (unsigned char flags,
                           unsigned char *uid,
                           unsigned char AFI,
                           unsigned short *resplen,
                           unsigned char *resp)
      IN   flags(1), uid(8), AFI(1)
      OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
AFI (IN) application family identifier.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.11 ISO15693_Lock_AFI

```
signed ISO15693_Lock_AFI (unsigned char flags,
                          unsigned char *uid,
                          unsigned short *resplen,
                          unsigned char *resp)
     IN   flags(1), uid(8)
     OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.12 ISO15693_Write_DSFID

```
signed ISO15693_Write_DSFID (unsigned char flags,
                            unsigned char *uid,
                            unsigned char DSFID,
                            unsigned short *resplen,
                            unsigned char *resp)
     IN   flags(1), uid(8), DSFID(1)
     OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.
uid (IN) unique identifier of the IC.
DSFID (IN) data storage format identifier.
resplen (OUT) for character arrays an additional length information is necessary.
resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.13 ISO15693_Lock_DSFID

```
signed ISO15693_Lock_DSFID (unsigned char flags,
                            unsigned char *uid,
                            unsigned short *resplen,
                            unsigned char *resp)
    IN   flags(1), uid(8)
    OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.

uid (IN) unique identifier of the IC.

resplen (OUT) for character arrays an additional length information is necessary.

resp (OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.14 ISO15693_Get_System_Information

```
signed ISO15693_Get_System_Information (unsigned char flags,
                                        unsigned char *uid,
                                        unsigned short *resplen,
                                        unsigned char *resp)
      IN   flags(1), uid(8)
      OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.15 ISO15693_Get_Multiple_Block_Security

```
signed ISO15693_Get_Multiple_Block_Security (unsigned char flags,
                                             unsigned char *uid,
                                             unsigned char blnr,
                                             unsigned char nbl,
                                             unsigned short *resplen,
                                             unsigned char *resp)
      IN   flags(1), uid(8), blnr(1), nbl(1)
      OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
blnr	(IN) number of the first block within the memory of the label which should be influenced by the command.
nbl	(IN) number of blocks that should be influenced by the command starting with the number of the first block.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type several functions are declared for adding data at the end of the data stream. If the data type consists several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

First byte of *resp* is the status. If the status is “*I1_OK*” or “*I2_NO_ERR*” the followed bytes are the response of the label as described in the ISO15693. For all other status the length of the response is only 1 byte, the status byte.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.16 ISO15693_Inventory_Read

```
signed ISO15693_Inventory_Read (unsigned char flags,
                                unsigned char ManCode,
                                unsigned char AFI,
                                unsigned char masklength,
                                unsigned char *uid,
                                unsigned char blnr,
                                unsigned char nbl,
                                unsigned short *resplen,
                                unsigned char *resp)
     IN   flags(1), ManCode(1), AFI(1), masklength(1), uid(8),
          blnr(1), nbl(1)
     OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
ManCode	(IN) manufacturer code.
AFI	(IN) application family identifier
masklength	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
blnr	(IN) number of the first block within the memory of the label which should be influenced by the command.
nbl	(IN) number of blocks that should be influenced by the command starting with the number of the first block.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

If the *inventory* command is used for 1 timeslot the response has the following format:

If status is “*I1_OK*”, “*I2_NO_ERR*” or “*I1_COLLERR*”

1st Byte	2nd Byte	1 Byte	x Bytes
Status	Collision position	Flags	Response of the labels, as described in the data sheet “ICODE SLI”. The value x is know before sending the command. x = nbl * 4 and rest of UID if available (see flags “ICODE SLI)

Else the response includes only the status!

If 16 timeslots are used the response is a combination of the above described values. If one or more labels answer within a timeslot the response for this timeslot includes 13 bytes, if an error occur within a timeslot the response includes 1 byte.

e.g.: 1st timeslot is free, at the second is a collision, at the third no error occur only one label answers and all others are free.

The response looks like this:

1st TS	2nd TS				3rd TS				4th TS		16th TS
0x01	0x03	0x11	Flags	Response	0x00	0x00	Flags	Response	0x01	…	0x01

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

Remark:

The collision position includes 8 bit for the “*Flags*” and 1 bit for the collision!

Consider that, it is possible to use the value of “collision position” as value for mask length.

3.5.2.17 ISO15693_Fast_Inventory_Read

```
signed ISO15693_Fast_Inventory_Read (unsigned char flags,
                                      unsigned char ManCode,
                                      unsigned char AFI,
                                      unsigned char masklength,
                                      unsigned char *uid,
                                      unsigned char blnr,
                                      unsigned char nbl,
                                      unsigned short *resplen,
                                      unsigned char *resp)
     IN   flags(1), ManCode(1), AFI(1), masklength(1), uid(8),
          blnr(1), nbl(1)
     OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
ManCode	(IN) manufacturer code.
AFI	(IN) application family identifier
masklength	(IN) as defined in ISO15693.
uid	(IN) unique identifier of the IC.
blnr	(IN) number of the first block within the memory of the label which should be influenced by the command.
nbl	(IN) number of blocks that should be influenced by the command starting with the number of the first block.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle functionality depending to the flags are described in the ISO15693.

If the *inventory* command is used for 1 timeslot the response has the following format:

If status is “*I1_OK*”, “*I2_NO_ERR*” or “*I1_COLLERR*”

1st Byte	2nd Byte	1 Byte	x Bytes
Status	Collision position	Flags	Response of the labels, as described in the data sheet “ICODE SLI”. The value x is know before sending the command. x = nbl * 4 and rest of UID if available (see flags “ICODE SLI”)

Else the response includes only the status!

If 16 timeslots are used the response is a combination of the above described values. If one or more labels answer within a timeslot the response for this timeslot includes 13 bytes, if an error occur within a timeslot the response includes 1 byte.

e.g.: 1st timeslot is free, at the second is a collision, at the third no error occur only one label answers and all others are free.

The response looks like this:

1st TS	2nd TS				3rd TS				4th TS		16th TS
0x01	0x03	0x11	Flags	Response	0x00	0x00	Flags	Response	0x01	…	0x01

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

Remark:

The collision position includes 8 bit for the “*Flags*” and 1 bit for the collision!

Consider that, it is possible to use the value of “collision position” as value for mask length.

3.5.2.18 ISO15693_Set_Eas

```
signed ISO15693_Set_Eas (unsigned char flags,
                         unsigned char ManCode,
                         unsigned char *uid,
                         unsigned short *resplen,
                         unsigned char *resp)
     IN   flags(1), ManCode(1), uid(8)
     OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
ManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command enables the EAS mode if this mode is not locked.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the specification (1 byte flags respectively 1 byte flags and 1 byte label-error code).

If a reader error occurs, only the status-byte is transmitted.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.19 ISO15693_Reset_Eas

```
signed ISO15693_Reset_Eas (unsigned char flags,
                           unsigned char ManCode,
                           unsigned char *uid,
                           unsigned short *resplen,
                           unsigned char *resp)
      IN   flags(1), ManCode(1), uid(8)
      OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
ManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command disables the EAS mode if this mode is not locked.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the specification (1 byte flags respectively 1 byte flags and 1 byte label-error code).

If a reader error occurs, only the status-byte is transmitted.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.20 ISO15693_Lock_Eas

```
signed ISO15693_Lock_Eas (unsigned char flags,
                          unsigned char ManCode,
                          unsigned char *uid,
                          unsigned short *resplen,
                          unsigned char *resp)
    IN   flags(1), ManCode(1), uid(8)
    OUT  resp(resplen)
```

Parameters:

flags	(IN) as defined in ISO15693.
ManCode	(IN) manufacturer code. 0x04 NXP Semiconductors.
uid	(IN) unique identifier of the IC.
resplen	(OUT) for character arrays an additional length information is necessary.
resp	(OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command locks the EAS mode and the EAS ID (only SLI-S & SLI-L).

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the specification (1 byte flags respectively 1 byte flags and 1 byte label-error code).

If a reader error occurs, only the status-byte is transmitted.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.5.2.21 ISO15693_Eas_Alarm

```
signed ISO15693_Eas_Alarm (unsigned char flags,
                           unsigned char ManCode,
                           unsigned char *uid,
                           unsigned char bEAS_ID_MaskLength,
                           unsigned char *pbEAS_ID,
                           unsigned short *resplen,
                           unsigned char *resp)
```

```
IN   flags(1), ManCode(1), uid(8), bEAS_ID_MaskLength(1),
     pbEAS_ID(0-2)
OUT  resp(resplen)
```

Parameters:

flags (IN) as defined in ISO15693.

ManCode (IN) manufacturer code. 0x04 NXP Semiconductors.

uid (IN) unique identifier of the IC.

bEAS_ID_MaskLength (IN) length of the EAS ID in bits (SLI-S & SLI-L).

pbEAS_ID (IN) fractal or complete EAS ID in order to address only certain labels (SLI-S & SLI-L).

resplen (OUT) for character arrays an additional length information is necessary.

Resp (OUT) depending on the data type of several functions are declared for adding data at the end of the data stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command sends the EAS alarm. With *option flag "0"*, no EAS-ID is transmitted. Use this mode for ICODE SLI. *Option flag "1"* is only supported by ICODE SLI-S and ICODE SLI-L.

First byte of the response is the status. The further response depends on the mode. For detailed information, please refer to the label datasheets.

The following status could occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6 ICODE SLI-S, ICODE SLI-L Command Set

Labels of the ICODE SLI-S, ICODE SLI-L family support all instructions that are supported by the ICODE SLI labels as well as additional SLI-S, SLI-L commands, which are described in this part of the document.

The further information on the labels command set please refer to the according product description of the ICODE SLI-S and ICODE SLI-L ICs.

3.6.1 Included Functions

Table 8. ICODE SLI-S, ICODE SLI-L specific command set

Function name	Function call
ICodeISO15693_PwdProtectEAS	*signed char ICodeISO15693_PwdProtectEAS* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ICodeISO15693_WriteEAS_ID	*signed char ICodeISO15693_WriteEAS_ID* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short wEAS_ID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ICodeISO15693_ReadEPC	*signed char ICodeISO156936_ReadEPC* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ICodeISO15693_GetRandom Number	*signed char ICodeISO15693_GetRandomNumber* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ISO15963_SetPwd	*signed char ISO15693_SetPwd (unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bPwdID,* *unsigned char *pbPwd,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*

Function name	Function call
ISO15693_WritePwd	*signed char ISO15693_WritePwd* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bPwdID,* *unsigned char *pbPwd,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ICodeISO15693_LockPWD	*signed char ICodeISO15693_LockPWD* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bPWD_ID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ICodeISO15693_64BitPWD Protection	*signed char ICodeISO15693_64BitPWDProtection* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*
ISO15693_ProtectPage	*signed char ISO15693_ProtectPage* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bPageNo,* *unsigned char bProtectionStatus,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ICodeISO15693_GetMultipleBlock ProtStatus	*Signed char ICodeISO15693_GetMultipleBlockProtStatus* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bFirstBlock,* *unsigned char bNoOfBlocks,* *unsigned short *pwRespLength,* *unsigned char *pbResp)*

Function name	Function call
ISO15693_LockPageProtection Condition	*signed char ISO15693_LockPageProtectionCondition* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned char bPageNo,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ISO15693_DestroyS	*signed char ISO15693_DestroyS* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ISO15693_InventoryReadS	*signed char ISO15693_InverntoryReadS* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char bAFI,* *unsigned char bMaskLen,* *unsigned char *pbUID,* *unsigned char bBlockNo,* *unsigned char bNoOfBlocks,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ISO15693_FastInverntoryReadS	*signed char ISO15693_FastInventoryReadS* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char bAFI,* *unsigned char bMaskLen,* *unsigned char *pbUID,* *unsigned char bBlockNo,* *unsigned char NoOfBlocks,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*
ICodeISO15693_EnablePrivacy	*signed char ISO15693_EnablePrivacy* *(unsigned char bFlags,* *unsigned char bManCode,* *unsigned char *pbUID,* *unsigned short *pbRespLen,* *unsigned char *pbResp)*

3.6.2 Function Description ICODE SLI-S, SLI-L

3.6.2.1 ICodeISO15693_PwdProtectEAS

```
signed char ICodeISO15693_PwdProtectEAS
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8),
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags (IN) as defined in ISO15693.
bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID (IN) unique identifier of the IC.
pwRespLength (OUT) for character arrays an additional length information is necessary.
pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function enables the password protection for EAS. The password has to be transmitted before using the *SetPassword* command.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification. If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.2 ICodeISO15693_WriteEAS_ID

```
signed char ICodeISO15693_PwdProtectEAS
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short wEAS_ID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), wEAS_ID(2)
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags (IN) as defined in ISO15693.
bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID (IN) unique identifier of the IC.
wEAS_ID (IN) EAS-ID to be set.
pwRespLength (OUT) for character arrays an additional length information is necessary.
pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function is used to set the EAS-ID. If EAS is password protected, the password has to be transmitted before.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification. If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.3 ICodeISO15693_ReadEPC

```
signed char ICodeISO15693_ReadEPC (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8)
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
pwRespLength	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command reads the EPC.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the specification (1 byte flags, 12 bytes EPC respectively 1byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.4 ICodeISO15693_GetRandomNumber

```
signed char ICodeISO15693_GetRandomNumber
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8)
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags (IN) as defined in ISO15693.

bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.

pbUID (IN) unique identifier of the IC.

wEAS_ID (IN) EAS-ID to be set.

pwRespLength (OUT) for character arrays an additional length information is necessary.

pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command is used for the password functionality.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags, 2 bytes random number respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.5 ISO15693_SetPwd

```
signed char ICodeISO15693_SetPwd (unsigned char bFlags,
                                  unsigned char bManCode,
                                  unsigned char *pbUID,
                                  unsigned char bPwdID,
                                  unsigned char *pbPwd,
                                  unsigned short *pbRespLen,
                                  unsigned char *pbResp)
```

```
IN   bFlags(1), bManCode(1), pbUID(8), bPwdID(1), pbPwd(4)
OUT  pbResp(pbRespLen)
```

Parameters:

bFlags (IN) as defined in ISO15693.

bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.

pbUID (IN) unique identifier of the IC.

bPwdID (IN) password identifier (0x01 read, 0x02 write, 0x04 privacy, 0x08 destroy, 0x10 EAS).

pbPwd (IN) password. Please refer to ICODE SLI-S specification for detailed information.

pbRespLen (OUT) for character arrays an additional length information is necessary.

pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function sets a password in order to use a password protection function. For detailed information please refer to the ICODE SLI-S specification.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.6 ISO15693_WritePwd

```
signed char ISO15693_WritePwd (unsigned char bFlags,
                               unsigned char bManCode,
                               unsigned char *pbUID,
                               unsigned char bPwdID,
                               unsigned char *pbPwd,
                               unsigned short *pbRespLen,
                               unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), bPwdID(1), pbPwd(4)
     OUT  pbResp(pbRespLen)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
bPwdID	(IN) password identifier (0x01 read, 0x02 write, 0x04 privacy, 0x08 destroy, 0x10 EAS).
pbPwd	(IN) password. Please refer to ICODE SLI-S specification for detailed information.
pbRespLen	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function writes a new password into the related memory. For detailed information please refer to the ICODE SLI-S specification.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.7 ICodeISO15693_LockPwd

```
signed char ICodeISO15693_LockPwd (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned char bPWD_ID
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), bPwdID(1)
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
bPwdID	(IN) password identifier (0x01 read, 0x02 write, 0x04 privacy, 0x08 destroy, 0x10 EAS).
pwRespLength	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command locks the specific password. It can’t be changed any more.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.8 ICodeISO15693_64BitPWDProtection

```
signed char ICodeISO15693_64BitPWDProtection
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8)
     OUT pbResp(pwRespLength)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
pwRespLength	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command sets the *64 bit password protection*. It can’t be changed any more.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.9 ISO15693_ProtectPage

```
signed char ICodeISO15693_ProtectPage
                                (unsigned char bFlags,
                                 unsigned char bManCode,
                                 unsigned char *pbUID,
                                 unsigned char bPageNo,
                                 unsigned char bProtectionStatus,
                                 unsigned short *pbRespLen,
                                 unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), bPageNo(1),
          bProtectionStatus(1)
     OUT  pbResp(pbRespLen)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
bPageNo	(IN) page to be protected.
bProtectionStatus	(IN) level of protection.
pbRespLen	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command sets the protection status of the page. For detailed information please refer to the ICODE SLI-S specification.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.10 ICodeISO15693_GetMultipleBlockProtStatus

```
signed char ICodeISO15693_GetMultipleBlockProtStatus
                                   (unsigned char bFlags,
                                    unsigned char bManCode,
                                    unsigned char *pbUID,
                                    unsigned char bFirstBlock,
                                    unsigned char bNoOfBlocks,
                                    unsigned short *pwRespLength,
                                    unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), bFirstBlock(1),
          bNoOfBlocks(1)
     OUT  pbResp(pwRespLength)
```

Parameters:

bFlags (IN) as defined in ISO15693.
bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID (IN) unique identifier of the IC.
bFirstBlock (IN) first block for reading out protection status.
bNoOfBlocks (IN) number of protection states to be read.
pwRespLength (OUT) for character arrays an additional length information is necessary.
pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This function reads a certain number of block protection states.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags and *bNoOfBlocks* bytes respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.11 ICodeISO15693_LockPageProtectionCondition

```
signed char ICodeISO15693_LockPageProtectionCondition
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned char bPageNo,
                                   unsigned short *pbRespLen,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8), bPageNo(1)
     OUT pbResp(pbRespLen)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
bpageNo	(IN) block protection status to be locked.
pbRespLen	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command is used to lock the *protection status* of one block. No further changes possible.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.12 ISO15693_DestroyS

```
signed char ISO15693_DestroyS (unsigned char bFlags,
                              unsigned char bManCode,
                              unsigned char *pbUID,
                              unsigned short *pbRespLen,
                              unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8)
     OUT  pbResp(pbRespLen)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
pbRespLen	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command destroys the ICODE SLI-S label if the destroy-s password has been transmitted before.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.13 ICodeISO15693_EnablePrivacy

```
signed char ICodeISO15693_EnablePrivacy
                                  (unsigned char bFlags,
                                   unsigned char bManCode,
                                   unsigned char *pbUID,
                                   unsigned short *pwRespLength,
                                   unsigned char *pbResp)
     IN   bFlags(1), bManCode(1), pbUID(8)
     OUT  pbResp(pwRespLenght)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
pbUID	(IN) unique identifier of the IC.
pwRespLength	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command enables the *privacy mode*. In this mode, the label will not response to any command, except *GetRandomNumber* and *SetPassword*.

First byte of the response is the status. In the case of no error found by the reader, the following bytes are according to the ICODE SLI-S specification (1 byte flags respectively 1 byte label-error code). If an error occurred, only the status-byte is transmitted.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.14 ISO15693_InventoryReadS

```
signed char ISO15693_InventoryReadS (unsigned char bFlags,
                                     unsigned char bManCode,
                                     unsigned char bAFI,
                                     unsigned char bMaskLen,
                                     unsigned char *pbUID,
                                     unsigned char bBlockNo,
                                     unsigned char bNoOfBlocks,
                                     unsigned short *pbRespLen,
                                     unsigned char *pbResp)
```

```
IN   bFlags(1), bManCode(1),bAFI(1), bMaskLen(1),
     pbUID(bMaskLen), bBlockNo(1), bNoOfBlocks(1)
OUT  pbResp(pbRespLen)
```

Parameters:

bFlags	(IN) as defined in ISO15693.
bManCode	(IN) manufacturer code. 0x04 for NXP Semiconductors.
bAFI	(IN) application family identifier.
bMasLen	(IN) length of mask (UID).
pbUID	(IN) unique identifier of the IC.
bBlockNo	(IN) first block to be read.
bNoOfBlocks	(IN) number of blocks to be read.
pbRespLen	(OUT) for character arrays an additional length information is necessary.
pbResp	(OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

The principle of the function is depending on the flags and described in the ISO15693 standard. If the inventory read command is used for 1 timeslot the response has the following format:

If status is “I1_OK”, “I2_NO_ERR” or “I1_COLLERR”

1st Byte	2nd Byte	1 Byte	x Bytes
Status	Collision position	Flags	Response of the labels, as described in the data sheet “ICODE SLI-S”.

Else the response includes only the status!

If 16 timeslots are used the response is a combination of the above described values. If one or more labels answer within a timeslot the response for this timeslot includes 13 byte, if an error occur within a timeslot the response includes 1 byte.

e.g.: 1st timeslot is free, at the second is a collision, at the third no error occur only one label answer and all others are empty.

The response look like this:

1st TS	2nd TS				3rd TS				4th TS		16th TS
0x01	0x03	0x11	Flags	Response	0x00	0x00	Flags	Response	0x01	…	0x01

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

3.6.2.15 ISO15693_FastInventoryReadS

```
signed char ISO15693_FastInventoryReadS (unsigned char bFlags,
                                         unsigned char bManCode,
                                         unsigned char bAFI,
                                         unsigned char bMaskLen,
                                         unsigned char *pbUID,
                                         unsigned char bBlockNo,
                                         unsigned char bNoOfBlocks,
                                         unsigned short *pbRespLen,
                                         unsigned char *pbResp)
```

```
IN   bFlags(1), bManCode(1),bAFI(1), bMaskLen(1),
     pbUID(bMaskLen), bBlockNo(1), bNoOfBlocks(1)
OUT  pbResp(pbRespLen)
```

Parameters:

bFlags (IN) as defined in ISO15693.
bManCode (IN) manufacturer code. 0x04 for NXP Semiconductors.
bAFI (IN) application family identifier.
bMasLen (IN) length of mask (UID).
pbUID (IN) unique identifier of the IC.
bBlockNo (IN) first block to be read.
bNoOfBlocks (IN) number of blocks to be read.
pbRespLen (OUT) for character arrays an additional length information is necessary.
pbResp (OUT) depending on the data type, several functions are declared for adding data at the end of the stream. If the data type consists of several bytes, the bytes are converted with the least significant byte first.

Returns:

MI_OK
I1_TIMEOUT

This command behaves the same as *InventoryReadS* command. Differences exits in the way of data transmission.

The following status can occur:

0x00	I1_OK or I2_NO_ERR
0x01	I2_NO_TAG
0x02	I2_CRCERR
0x03	I2_COLLERR
0x04	I2_SNRERR
0x05	I2_COUNTERR
0x07	I2_FRAMINGERR

4. Return Values Overview

The naming of the return values allows differing between reader and communication returns:

- MI_Reader errors
- I1_Reader errors
- COM_Communication errors (generally)
- COM_IRDA Communication errors from the IRDA interface
- COM_RS232 Communication errors from the RS232 interface
- COM_USB Communication errors from the USB interface

4.1 Table of Return Values

Table 9. Return Values

Value	Name of constant	Short description
0	COM_SUCCESS	Operation successful
0	MI_CHK_OK	Operation successful
0	MI_CRC_ZERO	Operation successful
0	MI_OK	Operation successful
0	I1_OK	Operation successful
0	I1_NO_ERR	Operation successful
-1	MI_CHK_FAILED	Reader: CRC Error
-2	MI_CHK_COMPERR	Reader: Check Compare Error
-2	MI_CRCERR	Reader: CRC Error
-3	MI_EMPTY	Reader: Value Overflow
-6	MI_CODEERR	Reader: Code Error
-8	MI_SERNRERR	Reader: Serial Number Error
-11	MI_BITCOUNTERR	Reader: Bit count error
-12	MI_BYTECOUNTERR	Reader: Byte count error
-13	MI_IDLE	Reader: Idle
-14	MI_TRANSERR	Reader: Transfer Error
-15	MI_WRITEERR	Reader: Write error
-16	MI_INCRERR	Reader: Increment error
-17	MI_DECRERR	Reader: Decrement error
-18	MI_READERR	Reader: Read error
-19	MI_OVFLERR	Reader: Overflow error
-20	MI_POLLING	Reader: Polling
-21	MI_FRAMINGERR	Reader: Framing error
-22	MI_ACCESSERR	Reader: Access error
-23	MI_UNKNOWN_COMMAND	Reader: Unknown Command
-24	MI_COLLERR	Reader: Reset error
-25	MI_INITERR	Reader: Initialization failed

Value	Name of constant	Short description
-25	MI_RESETERR	Reader: Reset Error
-26	MI_INTERFACEERR	Reader: Interface error
-27	MI_ACCESSTIMEOUT	Reader: Access timeout
-30	MI_QUIT	Reader: Quit error
-31	MI_CODINGERR	Reader: Code Error
-53	MI_SENDBUF_OVERFLOW	Reader: Send buffer overflow
-54	MI_BAUDRATE_NOT_SUPPORTED	Reader: Baudrate not supported
-55	MI_SAME_BAUDRATE_REQUIRED	Reader: Same baudrate required
-60	MI_WRONG_PARAMETER_VALUE	Reader: Wrong parameter value
-61	I1_WRONGPARAM	Reader: Wrong parameter
-62	I1_NYIMPLEMENTED	Reader: Command not yet implemented
-63	I1_TSREADY	Reader: TimeSlot is free
-70	I1_TIMEOUT	Reader: Timeout occurred
-71	I1_NOWRITE	Reader: no Write at that TimeSlot
-72	I1_NOHALT	Reader: no Halt at that TimeSlot
-73	I1_MISS_ANTICOLL	Reader: “*I1output_anticoll_select*” have to be done first
-82	I1_COMM_ABORT	Reader: COMM Abort
-99	MI_BREAK	Reader: ???
-100	MI_NY_IMPLEMENTED	Reader: Not yet implemented
-109	MI_FIFOERR	Reader: FIFO Error
-110	MI_WRONG_ADDR	Reader: Wrong address
-117	MI_WRONG_TEST_MODE	Reader: Wrong Test mode
-118	MI_TEST_FAILED	Reader: Test failed
-120	MI_COMM_ABORT	Reader: COMM Abort
-121	MI_INVALID_BASE	Reader: Invalid base
-123	MI_WRONG_VALUE	Reader: Wrong value
-124	MI_VALERR	Reader: Value Error
-151	MI_NO_VALUE	Reader: No Value
-180	MI_WRONG_BASEADDR	Reader: Wrong base address
-199	MI_NO_ERROR_TEXT_AVAIL	Reader: No Error Text available
-254	MI_DRIVER_FAILURE	Reader: Driver failure
-255	MI_INTERFACE_FAILURE	Reader: Interface failure
-260	MI_SERERR	Reader: Serial Number Error
-262	MI_RECBUF_OVERFLOW	Reader: Overflow of the receive buffer
-1001	COM_ERROR	HostRdCom: No overloaded function found
-1002	COM_NO_INTERFACE_HANDLE	HostRdCom: No valid interface handle
-1003	COM_INTERFACE_OPEN	HostRdCom: Interface is already opened
-1004	COM_INTERFACE_NOT_OPEN	HostRdCom: Interface is not opened
-1005	COM_CREATE_FILE_FAILED	HostRdCom: Command CreateFile() failed
-1006	COM_PURGE_COMM_FAILED	HostRdCom: Command PurgeComm() failed

Value	Name of constant	Short description
-1007	COM_GET_COMM_STATE_FAILED	HostRdCom: Command GetCommState() failed
-1008	COM_SETUP_COMM_FAILED	HostRdCom: Command SetupComm() failed
-1009	COM_SET_COMM_STATE_FAILED	HostRdCom: Command SetCommState() failed
-1010	COM_SET_COMM_MASK_FAILED	HostRdCom: Command SetMask() failed
-1011	COM_SET_COMM_TIMEOUTS_FAILED	HostRdCom: Command SetCommTimeouts failed
-1012	COM_WRONG_VALUE	HostRdCom: Passed parameter - wrong value
-1016	COM_READER_NOT_IN_RANGE	HostRdCom: Discovery failed - Reader not in range
-1017	COM_CONNECT_FAILED	HostRdCom: Connecting to reader failed
-1018	COM_NEW_FAILED	HostRdCom: New() failed - insufficient memory
-1019	COM_INVALID_WT_HANDLE	HostRdCom: Invalid worker thread handle
-1020	COM_START_WT_FAILED	HostRdCom: Starting worker thread failed
-1021	COM_INVALID_CB_HANDLE	HostRdCom: Passed callback handle is invalid
-1022	COM_LEN_OVERFLOW	HostRdCom: Buffer length overflow
-1023	COM_RS232_SERCOM_ERR	HostRdCom: Error on RS232 interface
-1024	COM_RS232_SEND_CMD_NO_DLE_ERR	HostRdCom: No DLE received from reader error
-1025	COM_RS232_SEND_DEVICE_ERR	HostRdCom: Error sending data to reader via RS232
-1026	COM_RS232_RESP_CMD_NAK_ERR	HostRdCom: Reader response: NAK
-1027	COM_TIMEOUT	HostRdCom: Timeout occurred
-1028	COM_RS232_RESP_TO_ERR	HostRdCom: First received character from reader not STX (RS232)
-1029	COM_RS232_RESP_OVERFLOW_ERR	HostRdCom: Response buffer overflow (RS232)
-1030	COM_RS232_RECV_DEVICE_ERR	HostRdCom: Error receiving data from reader via RS232
-1031	COM_RS232_RESP_UNDERFLOW_ERR	HostRdCom: To less bytes received from reader (RS232)
-1032	COM_RS232_DATALENGTH_ERR	HostRdCom: Wrong number of bytes received from reader (RS232)
-1033	COM_RS232_CHECKSUM_ERR	HostRdCom: Checksum error (RS232)
-1034	COM_RS232_TX_RX_SEQ_ERR	HostRdCom: Sequence numbers not equal (RS232)
-1035	COM_RS232_COPY_DATA_ERR	HostRdCom: Error copying data to command object (RS232)
-1036	COM_IRDA_SELECT_FAILED	HostRdCom: Command Select() failed (IrDA)
-1037	COM_IRDA_SEND_TIMEOUT	HostRdCom: Send timeout error (IrDA)
-1038	COM_IRDA_SOCKET_NOT_READY	HostRdCom: Socket not ready for transmitting data (IrDA)
-1039	COM_IRDA_SEND_DEVICE_ERR	HostRdCom: Error sending data to reader via IrDA
-1040	COM_IRDA_RECV_DEVICE_ERR	HostRdCom: Error receiving data from reader via IrDA
-1041	COM_IRDA_RECV_TIMEOUT	HostRdCom: Receive timeout error (IrDA)

Value	Name of constant	Short description
-1042	COM_IRDA_TX_RX_SEQ_ERR	HostRdCom: Sequence numbers not equal (IrDA)
-1043	COM_IRDA_COPY_DATA_ERR	HostRdCom: Error copying data to command object (IrDA)
-1044	COM_IRDA_LEN_ERR	HostRdCom: Wrong number of bytes received from reader (IrDA)
-1045	COM_NO_PROTOCOL_SET	HostRdCom: No protocol set
-1046	COM_USB_DLL_LOAD_ERR	HostRdCom: Error loading USB Dll
-1047	COM_USB_MISSING_FCT_ADDR	HostRdCom: Error loading function addresses (UBS)
-1048	COM_USB_SEND_DEVICE_ERR	HostRdCom: Error sending data to reader (USB)
-1049	COM_USB_RECV_DEVICE_ERR	HostRdCom: Error receiving data from reader (USB)
-1050	COM_USB_TX_RX_SEQ_ERR	HostRdCom: Sequence numbers not equal (USB)
-1051	COM_USB_LEN_ERR	HostRdCom: Wrong number of bytes received from reader (USB)
-1052	COM_USB_COPY_DATA_ERR	HostRdCom: Error copying data to command object (IrDA)
-1053	COM_USB_NO_DEVICE_FOUND	HostRdCom: No device found (USB)
-1054	COM_USB_SEND_TIMEOUT	HostRdCom: Timeout period exceeded while writing to a device (USB)
-1055	COM_USB_RECV_TIMEOUT	HostRdCom: Timeout periode exceeded while reading from a device USB)
-1056	COM_USB_FILE_NOT_FOUND	HostRdCom: File descriptor not longer valid (USB)
-1057	COM_USB_ACCESS_DENIED	HostRdCom: Device could not be accessed (USB)
-1058	COM_RS232_ETX_DLE_EXPECTED	HostRdCom: Receive error at ISO3964 protocol (RS232)

5. Legal information

5.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

5.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer’s own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

5.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

5.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

5.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP B.V.

6. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: November 2009
Document identifier: AN 077520

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: November 2009

Document identifier: AN 077520