AN12207

SCI2C Protocol Specification

Rev. 1.6 — 11 June 2018 **Application Note**

Document information

Info	Content
Keywords	I²C-bus, Smart Card, APDU mapping
Abstract	This document contains the definition of a Smart Card I²C (SCI2C) Protocol using an Inter-IC (I²C) based physical interface and data link layer, a SMBus based network layer and bus protocol as well as a mapping layer to convey ISO/IEC 7816-4 based communication. It defines a model where the terminal (Reader/writer) is mapped to an I²C master device and the card is mapped to an I²C slave device. It allows for control and data communication between the communication partners and provides a mapping of the so-called Application Protocol Data Units. It supports microcontroller driven smart cards with full featured operating systems supporting APDUs (Application Protocol Data Units) operating in an I²C-bus environment.

Revision history

Rev	Date	Description
1.6	20180611	Editorial updates, no content change
1.5	20170131	Correct note in chapter 13.10.3 / Figure 5
1.4	20151217	Add additional recommendation for cold reset See chapter 4.2, 11.1 and 13.4
1.3	20141008	Fix some minor mistake in figures and examples; add parameter FWI_{DEF} add definition of Protocol Binding Confirmation code (Table 14) correct definition for protocol exception trigger in chapter 13.9 corrections/clarification in chapters 13.10.3 and 13.10.4 adding chapter 24"I2C address (Informative)" correct Maximum Data Bytes (Slave to Master) Table 9 correct Maximum Data Bytes (Master to Slave) Table 11
1.2	20120426	Correction of $CDB_{SM,MAX}$ values; update of value definition table (CDB, t_{CMDG}, t_{PDE}); removal of slave NACK requirement on wrong EDCCB; clean-up of missing references
1.1	20100929	Add list of abbreviations; add some missing requirements; fix some mistakes in figures and in the examples; add a bus clear procedure; change text on how to split payload streams into chunks; switch from CRC to LRC based error detection code; switch to a different block splitting; add status command; add missing requirement for the communication speed data element; add frame waiting time parameter as new low level data element
1.0	20100624	Initial release

Contact information

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

1. Introduction

1.1 Scope

The scope of this document covers the definition of a Smart Card I2C (SCIIC) Protocol using an Inter-IC (I2C) based physical interface and data link layer, a SMBus based network layer and bus protocol as well as a mapping layer to convey [7816-4] based communication.

1.2 Audience

This document is intended for the use of manufacturers wanting to implement a master device or a slave device supporting SCIIC.

1.3 Applicable Document and References

The following documents contain provisions that are referenced in this specification. The latest version including all published amendments applies unless a publication data is explicitly stated.

[7816-3]	ISO/IEC 7816-3: “Identification cards - Integrated circuit cards – Card with contacts – Electrical interface and transmission protocols”, Third edition 2006-11-01
[7816-4]	ISO/IEC 7816-4: “Identification cards - Integrated circuit cards - Organization, security and commands for interchange”, Second edition 2005-01-15
[7816-10]	ISO/IEC 7816-10: “Identification cards - Integrated circuit(s) cards with contacts – Electronic signals and answer to reset for synchronous cards“, Third edition 2003-11-01
[8825-1]	ISO/IEC 8825-1: “Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) “, Fourth edition 2008-12-15
[13239]	ISO/IEC 13239: “Information technology — Telecommunications and information exchange between systems — High-level data link control (HDLC) procedures“, Third edition 2002-07-15
[I2C-bus]	UM10204 - I^2C-bus specification and user manual, Rev. 03, 19 June 2007, NXP Semiconductors, http://ics.nxp.com/support/documents/interface/pdf/i2c.bus.specification.pdf
[PMBUS-1]	Power System Management Protocol Specification, Part I – General Requirements, Transport and Electrical Interface, Revision 1.1, 5 February 2007, SMI Forum http://pmbus.org/specs.html

[PMBUS-2] Power System Management Protocol Specification, Part II – Command Language, Revision 1.1, 5 February 2007, SMI Forum http://pmbus.org/specs.html

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels, March 1997

[SMBUS] System Management Bus Specification, Version 2.0 August 3rd 2000, SBS Implementers Forum, http://www.sbs-forum.org/specs/

1.4 Notational Conventions

1.4.1 Notations

The following notations apply to this document:

Notation	Description
XYh	Hexadecimal notation. Values expressed in hexadecimal form are followed by a lower case “h”. For example, 27509 decimal is expressed in hexadecimal as 6B75h.
XYb	Binary notation. Values expressed in binary form are followed by a lower case “b”. For example, 82h hexadecimal is expressed in binary as 10000010b.
xx	More than one value possible.
-	Bit position that is not relevant for the definition of the element.

Table 1 Notational Conventions

1.4.2 Bit numbering

The bit numbering applied in this document is b0 to b7. The bit b0 refers to the least significant bit of a byte and b7 to the most significant bit of a byte.

I.e. a value of 10000000b is a byte (eight bits in size) where the ‘1’ is the most significant bit b7.

1.4.3 Reserved for Future Use

An entity transmitting bytes or bits defined as RFU (Reserved for Future Use) SHALL set these bytes or bits to the value indicated, or to zero if no value is given.

An entity receiving bytes or bits defined as RFU SHALL disregard these bytes or bits and SHALL keep the same interpretation of any other field of the whole response, unless explicitly stated otherwise.

Note: An entity transmitting bytes or bits defined as Any Value may set these bytes or bits to any specific value. How to process bytes or bits received that are defined as Any Value is out of scope of this specification, where not explicitly stated otherwise.

1.4.4 Drawings

The I^2C-bus and SMBus packet drawings as used within this specification follow the definitions as made in [SMBUS].

1.4.5 Logic Levels

A logic ‘0’ on the SCL and SDA lines are called LOW and a logic ‘1’ is called HIGH within this specification.

1.4.6 TLV data object

This specification is using simple TLV (Tag Length Value) data objects to encode some of the information being exchanged between slave device and master device. The encoding used follows the definition of BER-TLVs (Basic Encoding Rules-TLV) as defined within [7816-4] and [8825-1].

A TLV data object consists of two to three fields:

- Tag field (REQUIRED)

- Length field (REQUIRED)
- Value field (OPTIONAL)

1.4.6.1 Tag field of a TLV data object

The tag field consists of a single byte encoding the tag type and number. For an encoding compatibility with [7816-4] and [8825-1] the bits b6 and b7 SHALL both be set to 1b to indicate a tag in private class and bit b5 SHALL be set to 0b to indicate a primitive encoding of the data object.

1.4.6.2 Length field of a TLV data object

The length field consists of a single byte encoding a number N in the range from 0 to 127. The length field content in the range from 128 to 255 is RFU.

1.4.6.3 Value field of a TLV data object

The value field consists of N bytes encoding the value information of the tag. The value field is not present in case the length field contains an N equal to 0.

1.5 Special Word Usage

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

1.6 Abbreviations

This chapter provides an overview of the abbreviations as used in this document.

Abbreviation	Description
AID	Application IDentifier
APDU	Application Protocol Data Unit
ASCII	American Standard Code for Information Interchange
ATR	Answer To Reset
BER	Basic Encoding Rules
BR	Bit Rate
B UF	BUFfer
CDB	Command Data Byte
CLA	CLAss
CMD	CoMmanD
CRC	Cyclic Redundancy Chec
C-APDU	Command-APDU
DEF	DEFault
EDC	Error Detection Code
EDCCB	Error Detection Code Control Byte
FWI	Frame Waiting Integer
HDLC	High-level Data Link Control
IEC	International Electrotechnical Commission
INIT	INITial
INS	INStruction
ISO	International Organization for Standardization
I/O	Input/Output

Abbreviation	Description
LEN	LENgth
LRC	Longitudinal Redundancy Code
MAX	MAXimum
MD	Master Delay
MDPV	Master Device Protocol Version
MDPVMA	Master Device Protocol Version MAjor
MDPVMI	Master Device Protocol Version Minor
MS	Master to Slave
N	Number
P1	Parameter 1
P2	Parameter 2
PBS	Protocol Binding Selection
PCB	Protocol Control Byte
PD	Power Down
PE	Parameter Exchange
PEC	Packet Error Code
PWRSAV	PoWeR SAVing
RATR	Read Answer To Reset
RFU	Reserved for Future Use
RST	ReSeT
R-APDU	Response-APDU
TLV	Tag Length Value
S	Slave
SADDR	Slave ADDRess
SBS	Smart Battery System
SCI2C	Smart Card Inter-IC
SCL	Serial Clock Line
SDA	Serial Data Line
SDPV	Slave Device Protocol Version
SDPVMA	Slave Device Protocol Version Major
SDPVMI	Slave Device Protocol Version Minor
SM	Slave to Master
SMBus	System Management Bus
SMI	System Management Interface
UM	User Manual
WNCMD	Wakeup Next CoMmanD
W	Wakeup

Table 2 Abbreviations

2. Overview

This document contains the definition of a Smart Card I²C (SCIIC) Protocol using an Inter-IC (I²C) based physical interface and data link layer, a SMBus based network layer and bus protocol as well as a mapping layer to convey [7816-4] based communication.

It defines a model where the terminal (Reader/writer) is mapped to an I²C master device and the card is mapped to an I²C slave device. It allows for control and data communication between the communication partners and provides a mapping of the so called Application Protocol Data Units.

The combination of above allows for control and data exchange between the communication partners and provides a mapping of the so called APDUs (Application Protocol Data Units, see [7816-4]). It supports microcontroller driven smart cards with full featured operating systems supporting APDUs operating in an I²C-bus environment (see [I2C-bus]).

The protocol defined in this document allows a smart card to operate besides slaves on a single bus with the support for multi-slave and multi-master as well.

The content of this document is not to be confused with memory cards compatible with [7816-10] and is also not related to document of manufactures that do map functionality of synchronous cards to specific defined APDUs to access them in an abstract way.

The informative chapters at the end of this specification, declared by the term informative in brackets after the chapter title, are for information only and do in no way overrule any definitions.

3. Signals

This chapter defines the electrical signals to be supported by a master device and a slave device compliant to this specification.

The following signals SHALL be supported:

- SCL serial clock line
- SDA serial data line

The following signal MAY be supported by a slave device:

- RST_N reset signal input

Note: This specification does not contain the assignment of signals to specific pins of a package. This is left to the device and package related specification of a product.

4. Operating Conditions

This chapter describes the basic operation conditions as well as the state changes based on the commands received. The state machine of the slave device is provided in chapter 18.

4.1 Power On

The slave device SHALL enter the READY state when supply is applied and the RST_N signal is set to high state. In case the RST_N signal is set to low state when supply is applied the slave device will only enter the READY state after the end of the sequence as defined in chapter 13.4.

4.2 Reset

The slave device SHALL support means to be reset without the need to disconnect the power supply. A reset is used by the master device to reset all non-persistent states inside the slave device. There are two kinds of resets defined:

- Cold Reset (RECOMMENDED) and
- Soft Reset (REQUIRED)

The Cold Reset is issued by a sequence as defined in chapter 13.4.

The Soft Reset is issued by a command as defined in chapter 13.5.

The Cold Reset sequence as well as the Soft Reset command SHALL reset all non-persistent information inside the slave device and it SHALL switch the slave device into its READY state.

Note: In case of exceptional processing or a failure to recover from communication issues the slave SHALL be reset using a Cold Reset or a power cycle.

4.3 Power Saving

The slave device SHALL support means to switch to a power saving mode. This specification allows for devices to implicitly switch to a PWRSAV (PoWeR SAVing) state but it also defines two special commands to enable the PWRSAV state:

- Power Down and
- Power Down for Cold Reset

The Power Down command puts the device into the PWRSAV state when the slave device is in ACTIVE state or it puts the device into the PWRSAV* state when the slave device is in READY state. From there the slave device awaits a Wakeup command to switch back to its original state before entering the power saving mode. The Power Down command as well as the PWRSAV state and the PWRSAV* state SHALL NOT reset non-persistent states inside the slave device.

The Power Down for Reset command puts the device into the WAITRESET state. From there the slave device will only react on a Cold Reset.

The Power Down commands are defined in chapter 13.1 and chapter 13.2 and the Wakeup command is defined in chapter 13.3.

5. I²C-bus

The I²C-bus as referred within this document is defined in [I2C-bus].

The SCL and SDA signals created by the master device SHALL be compliant with the input voltage definitions of [I2C-bus].

The SCL and SDA signals created by the slave device MAY be compliant with the input and voltage definitions of [I2C-bus].

The master device SHALL support a 7-bit slave addressing.

The master device MAY apply a bus clear as defined in [I2C-bus] and it SHALL only use the bus clear in case it detects the unlikely event where the SDA signal is stuck LOW creating a bus jam.

The slave device SHALL support a 7-bit addressing.

Note: There are no requirements for a slave device to be fully compliant to the input and voltage definitions of [I2C-bus] to allow slave devices with generic I/O driver capabilities to comply with this specification. It's left to the system implementation or system integrator to identify required and suitable combinations of master devices and slave devices based on the design of the bus, the routing of the signals and the combination of devices hooked on the bus.

6. SMBus

6.1 Generic Requirements

The SMBus (System Management Bus) as referred within this document refers to [SMBUS]. The definitions in this chapter specify options to be supported and possible constraints compared to [SMBUS].

The master device SHALL support the clock low extension as specified in [SMBUS].

The slave device MAY use the clock low extension as specified in [SMBUS] in case it can't support the speed of the master device at certain times.

The PEC (Packet Error Code) as specified in [SMBUS] SHALL NOT be used.

The slave device MAY NOT support the SMBus host notifies protocol as specified in [SMBUS].

The slave device SHALL acknowledge its own address if not otherwise stated in this specification.

The slave device MAY NOT support the address resolution proposal as specified in [SMBUS].

The slave device MAY NOT support the SMBSUS signal as specified in [SMBUS].

The slave device MAY NOT support the SMBALERT signal as specified in [SMBUS].

Note: Even though the PEC is not supported the integrity of higher layer data exchanged is taken care of via an error detection code specified in this specification.

Note: There are no requirements for a slave device to be fully compliant to the input and voltage definitions of [SMBUS] to allow slave devices with generic I/O driver capabilities to comply with this specification. It's left to the system implementation or system integrator to identify required and suitable combinations of master devices and slave devices based on the design of the bus, the routing of the signals and the combination of devices hooked on the bus.

7. PMBus

The PMBus (Power Management Bus) as referred within this document refers to [PMBUS-1] and [PMBUS-2]. The definitions in this chapter specify options to be supported and possible constraints compared to [PMBUS-1] and [PMBUS-2].

The master device as specified in this specification is not compliant to [PMBUS-1] and [PMBUS-2] even though it's be possible to implement a master device supporting both specifications.

The slave device as specified in this specification is not compliant to [PMBUS-1] and [PMBUS-2]. Anyhow most of the definitions in [PMBUS-1] are taken from [SMBUS] and therefore do not differ a lot.

In general, the maximum amount of data bytes within a SMBus packet, as defined within this specification, is compliant to the definitions made in [PMBUS-1].

The command code values as used for the activation of a slave device according to this specification are not compliant to PMBus but they are not overlapping existing commands as defined within [PMBUS-2].

Note: The commands Wakeup, SoftReset as well as Read Answer to Reset do all use command codes declared as Reserved within [PMBUS-2] and therefore a detection and separation of slave devices implementing this specification vs. implementing the PMBus specifications is possible.

8. Slave Address Assignment

The default address of a slave device compliant to this specification SHALL be within range of SADDR (Slave ADDRess, see chapter 21).

The slave device MAY be configurable to use another slave address. The means to configure the actual used slave address is out of scope of this specification.

The slave device SHALL acknowledge addresses provided by the master devices matching its own assigned address if not otherwise specified in this document.

The slave device SHALL NOT acknowledge addresses provided by the master device not matching the address assigned to it.

Master devices compliant to this specification SHALL address slave devices compliant to this specification only by using the single address the slave device is configured to respond at.

9. Communication Speed

The master device SHALL support byte level clock stretching and it MAY support bit level clock stretching.

The master device SHALL use an initial maximum communication speed of BR_{INIT} (Bit Rate INITial, see chapter 21) in case it does not support bit level clock stretching.

The slave device SHALL only use byte level clock stretching when being addressed by the master device using a communication speed of up to $BR_{S,MAX}$.

The slave device SHALL be able to slow down the bit rate by using bit level clock stretching when being addressed by the master device using a communication speed above $BR_{S,MAX}$.

Note: The initial maximum communication speed may be limited by restrictions of other slave devices on the same bus.

10. Timing Requirements

10.1 Command Delay Time

The master device SHALL wait at least t_{CMDG} (time, ComManD Guard, see chapter 21) between two commands issued on the bus. This time SHALL be the time between the end of a Stop Condition of a previous command and the start of the Start Condition of a following command.

The slave device SHALL be able to receive a next command within t_{CMDG} after the end of a Stop Condition of a previous command if not otherwise stated within this specification.

Note: The Command Delay Time does not apply for a single command using a Repeated Start Condition.

11. Bus Error conditions

11.1 Bus Jam

A bus jam describes a condition where either SCL or SDA or both SCL and SDA stay low continuously.

Note: In case of SCL stay LOW, the slave SHALL be reset using a Cold Reset.

11.2 No Address Acknowledgement

NoAddrAck (No Address Acknowledgement) is a condition where the slave device does not acknowledge a slave address sent by the master device.

11.3 No Data Acknowledge

NoDataAck (No Data Acknowledgement) is a condition where either the slave device does not acknowledge a byte sent by the master device or where the master device does not acknowledge a byte sent by the slave device.

12. Bus Packets

The master device and the slave device SHALL implement the following SMBus packets:

- Quick command
- Send byte
- Block write
- Block read

The packets listed above SHALL be implemented in compliance with [SMBUS] if not otherwise stated in this specification.

The master device SHALL use the above packets when addressing a slave device compliant to this specification.

The master device SHALL NOT combine the above packets using repeated start conditions when addressing a slave device compliant to this specification.

The master device SHALL indicate the data direction by a single bit following the slave address.

The master SHALL indicate a master to slave data direction by setting the bit to 0b (Wr) and a slave to master data direction by setting the bit to 1b (Rd).

The command code definition as contained in [SMBUS] is used by this specification to exchange protocol related information called PCB (Protocol Control Byte). For the use of a PCB, which is needed in both directions, this specification extends the definitions thereof (see chapter 12.2 and chapter 12.4).

The following chapters define the used SMBus packets in full detail. The representation used for the figures equals the one used in [SMBUS]. It describes the data on the SCL line in a timeline where the first field shown (most top left in the figures) is the first data sent and the last field shown (most bottom right in the figures) is the last data sent. The number above a field defines the number of data bits of the field when sent. The number below a field defines the logical level of the SDA signal at that time.

Note: Even though this specification extends [SMBUS] to exchange a PCB in both directions the coding of the packet format as such has not been changed. This

has been kept to allow hosts using an SMBus compliant master device to exchange commands according to this specification as well. As a result of the decision to stay backwards compatible the data exchange capabilities between the master device and a slave device is asymmetrical. A master device can send one byte more to the slave in a single command compared to the number of bytes a slave can return in a single command.

Note: A repeated start condition is to be used by the master device for the Block read packet only. The slave device does not recognize bus packets when combined using repeated start conditions.

12.1 Quick Command

The following figure shows the format of the Quick command packet:

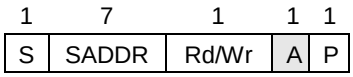


Figure 1 Quick command packet

Note: This specification uses the read indication and the write indication of the Quick packet to code a command. This is done to ease the protocol implementation and to optimize the amount of data exchanged.

12.2 Send Byte

The following figure shows the format of the Send byte packet:

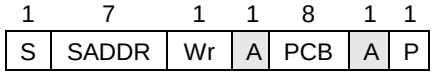


Figure 2 Send byte packet

Note: Devices according to this specification exchange a PCB within the single byte contained within the Send byte whereas [SMBUS] specifies a generic data byte.

12.3 Block write

The following figure shows the format of the Block write packet:

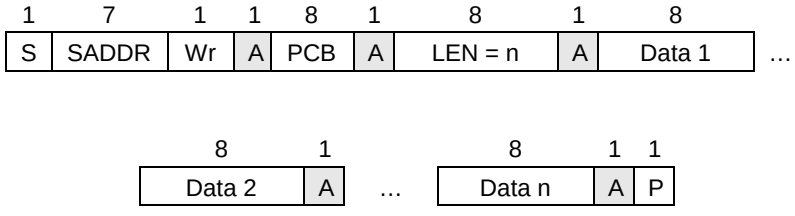


Figure 3 Block write packet

The field LEN within the Block write encodes the number of data bytes being sent by the master device to the slave device.

The master device SHALL be able to send any number from 0 up to and including $CDB_{MS,DEF}$ (Command Data Bytes, Master to Slave, DEFault, see chapter 21) data bytes in a Block write packet to a slave device.

The master device SHALL be able to detect that the slave device does not acknowledge one of the bytes within a packet and in such case of an unexpected not acknowledge the master device SHALL raise a Protocol Exception.

The master device SHALL create a Stop Condition to indicate the end of a write transfer.

The slave device SHALL be able to retrieve any number from 0 up to and including $CDB_{MS,DEF}$ data bytes in a Block write packet from a master device.

The slave device SHALL acknowledge all bytes received from the master device.

The slave device SHALL be able to detect a mismatch of data bytes received within the Block write packet compared to the LEN byte as contained within the packet and in such case of a mismatch the slave device SHALL raise a Protocol Exception (see chapter 14.2).

12.4 Block read

The following figure shows the format of the Block read packet:

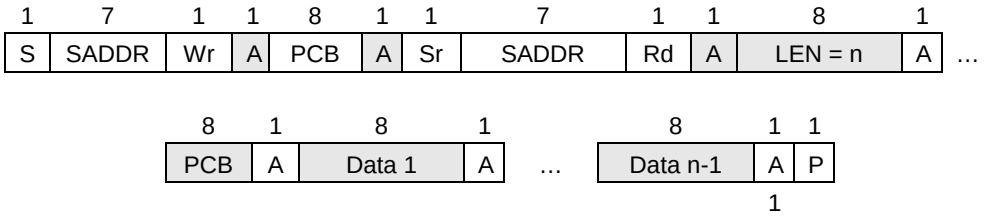


Figure 4 Block read packet

The field LEN within the Block read encodes the number of data bytes + 1 being received by the master device to the slave device.

The master device SHALL be able to retrieve any number from 0 up to and including $CDB_{SM,DEF}$ (Command Data Bytes, Slave to Master, DEFault, see chapter 21) data bytes in a Block read packet from a slave device.

The master device SHALL acknowledge all bytes read from the slave device except for the last byte, which the master device SHALL NOT acknowledge to indicate the end of the read transfer.

The slave device SHALL be able to return any number from 0 up to and including $CDB_{SM,DEF}$ data bytes in a Block read packet to a slave device.

The slave device SHALL be able to detect that the master device does not acknowledge one of the bytes within a packet and in such case of an unexpected not acknowledge the slave device SHALL raise a Protocol Exception.

The slave device SHALL be able to detect that the master device acknowledges also the last byte and in such case of an unexpected acknowledge the slave device SHALL raise a Protocol Exception.

The master device SHALL be able to detect a mismatch of data bytes received within the Block read packet compared to the LEN byte as contained within the packet and in case of a mismatch the master device SHALL raise a Protocol Exception (see chapter 14.2).

In case the master device receives a PCB from the slave device indicating an exception (see chapter 14.1.1) the master device SHALL raise a Protocol Exception.

Note: The PCB field returned by the slave device is included in the LEN variable. As a result of that the LEN is set to 1 even though no actual data bytes are being returned by the slave device. Assuming $CDB_{SM,DEF}$ to be 31 and the maximum 31 data bytes are actually returned by the slave device the LEN field has a value of 32.

Note: Slave devices according to this specification return a PCB next to the byte(s) whereas [SMBUS] only specifies data bytes to be returned.

Note: The last byte as used above means the last byte according to the LEN field indicated by the slave device. The master device is expected to read exactly the amount of bytes as indicated by the slave device. Anyhow the master device may not read all bytes

13. Commands

This chapter defines command set being used. These commands are constructed based on the bus packets as defined in chapter 12.

The following commands are defined:

- Power Down
- Power Down for Cold Reset
- Wakeup
- Cold Reset
- Soft Reset
- Read Answer to Reset
- Parameter Exchange
- Protocol Binding Selection
- Status
- Data Exchange
    - Master to Slave Data Transmission
    - Slave to Master data Transmission

The following chapters specify the commands in full detail.

Note: Cold Reset is not a command being constructed based on a bus packet but based on the RST_N signal.

13.1 Power Down

The Power Down command SHALL be based on the Quick command packet and SHALL contain the data direction bit set to 0b (Wr) (see chapter 12.1).

The master device MAY NOT implement the Power Down command.

The master device MAY at any time issue the Power Down command.

The master device SHALL only assume the slave device to have entered the PWRSAV state in case the Power Down command has been acknowledged.

The master device SHALL do $N_{RETRY,PD}$ (number of retries power down) attempts to put the device into the PWRSAV state in case the slave does not acknowledge its slave address before raising a Protocol Error. This retry limit SHALL NOT apply in the case that the previous command was the last command of a chain of Master to Slave Data Transmission commands.

The slave device SHALL be able to receive a Power Down command when its current state is the READY state and when its current state is the ACTIVE state.

On reception of a Power Down command the slave device SHALL change its state to PWRSAV when its current state is the ACTIVE state and it SHALL change its state to PWRSAV* when its current state is the READY state.

The slave device SHALL have entered the PWRSAV state or the PWRSAV* state (its power saving modes) within a time t_{PDE} (time, Power Down Enter see chapter 21) after it has received the Stop Condition of the Power Down command.

Note: Acknowledged as used above means in this case that the slave device has acknowledged the address byte.

Note: The slave device MAY implement an automatic power down by entering the PWRSAV state after a defined time of no communication when its current state is the ACTIVE state.

Note: The slave device MAY implement an automatic power down by entering the PWRSAV* state after a defined time of no communication when its current state is the READY state.

13.2 Power Down for Cold Reset

The Power Down for Cold Reset command SHALL be based on the Quick command packet and SHALL contain the data direction bit set to 1b (Rd) (see chapter 12.1).

The master device MAY implement the Power Down for Cold Reset command.

The master device MAY at any time issue the Power Down for Cold Reset command to slave devices supporting the command.

The master device SHALL only assume the slave device to have entered the PWRSAV state in case the Power Down for Cold Reset command has been acknowledged.

The master device SHALL do $N_{RETRY,PDCR}$ (number of retries power down for cold reset) attempts to put the device into the PWRSAV state in case the slave does not acknowledge its slave address before raising a Protocol Error. This retry limit SHALL NOT apply in the case that the previous command was the last command of a chain of Master to Slave Data Transmission commands.

The slave device SHALL implement the Power Down for Cold Reset command when supporting the Cold Reset sequence.

In case the slave device supports the Power Down for Cold Reset command it SHALL be able to receive a Power Down for Cold Reset command when its current state is the READY state and when its current state is the ACTIVE state.

On reception of a Power Down for Cold Reset command the slave device SHALL change its state to WAITRESET when its current state is the ACTIVE state or its current state is the READY state.

The slave device SHALL have entered the PWRSAV state within a time t_{PDE} (time, Power Down Enter, see chapter 21)) after it has received the Stop Condition of the command and it SHALL only leave the PWRSAV state when detecting a Cold Reset.

Note: Acknowledged as used above means in this case that the slave device has acknowledged the address byte.

Note: The slave device does not react on communication on the bus anymore after the Power Down for Cold Reset command.

Note: This command may be used to put a slave device implementing the I^2C-bus in software to stay in the PWRSAV state while a master communicates with other slave devices on the same bus.

13.3 Wakeup

The Wakeup command SHALL be based on the Send Byte packet.

The master device MAY NOT implement the Wakeup command.

The master device MAY at any time issue the Wakeup command.

The PCB contained SHALL have a value in accordance with Table 3.

Value	Meaning
--001111b	Protocol Data – Wakeup command

Table 3 Wakeup – PCB Master to Slave

The bits b0 to b5 set to 001111b indicate the Wakeup command.

The bits b6 and b7 of the PCB of the Wakeup command are RFU.

The master device SHALL issue a Stop Condition to conclude the command and release the bus, without sending the following PCB, in case the slave device does not acknowledge its slave address within the command.

The master device SHALL always assume the slave device to have left the PWRSAV state and being ready to receive command again even if during the Wakeup command a NoAddrAck error occurred.

The master device SHALL send the next command to the slave device no later than t_{WNCMD} (time, Wakeup Next ComManD, see chapter 21) after the end of Stop Condition of the Wakeup command.

In case the Wakeup command ends in a NoAddrAck and the command following the Wakeup command end in a NoAddrAck or NoDataAck the master device SHALL do $N_{RETRY,W}$ (Number of RETRY Wakeup) attempts to wake up the slave device from the PWRSAV state before raising a Protocol Error.

The slave device SHALL be able to receive a Wakeup command when its current state is the READY state, when its current state is the ACTIVE state as well as when its current state is the PWRSAV state.

The slave device SHALL change its state back to the state being in before entering the PWRSAV state.

The slave device SHALL be ready to be addressed by the master device within t_{CMDG} after the end of the Stop Condition of the Wakeup command.

Note: A NoAddrAck error may be caused by slave devices that are not able to actually decode correctly a wakeup command while being in the PWRSAV state but they indeed need to wake up based on state changes of bus signals and then are able to detect the next command issued on the bus.
In principle such different behavior of acknowledgement could be used by the master device to differentiate or even identify different slave device implementations.

13.4 Cold Reset

It is RECOMMENDED the master device supports the Cold Reset sequence in cases of exceptional processing or to recover from communication issues.

The master device MAY at any time issue a Cold Reset sequence to slave devices supporting the RST_N signal.

The Cold Reset sequence is issued by a master device having access to the RST_N signal input of the slave device by pulling the RST_N signal to low state for at least t_{RSTL} (time, ReSeT Low, see chapter 21) and then releasing to high state again afterwards.

The master device SHALL wait at least t_{RSTG} (time, ReSeT Guard, see chapter 21) after releasing the RST_N signal input of the salve device to high state before sending any other command on the bus.

The master device SHALL send the Read Answer to Reset as next command after a Cold Reset sequence to the slave device and only after successful execution of the Read Answer to Reset command the master device SHALL assume the slave device has reset all its non-persistent information.

The slave device SHALL implement the Cold Reset sequence when supporting a RST_N reset signal input.

In case the slave device supports the Cold Reset sequence the slave device SHALL at any time be able to react on a Cold Reset sequence.

On reception of a Cold Reset sequence the slave device SHALL change its state to the READY state.

The slave device SHALL be ready to receive the next command within a time of t_{RSTG} after the master device has released the RST_N signal to high state.

Note: A Cold Reset is sometimes also called Power-on Reset.

13.5 Soft Reset

The Soft Reset command SHALL be based on the Read block packet.

The master device MAY NOT implement the Soft Reset command.

The master device MAY at any time issue the Soft Reset command.

The PCB contained SHALL have a value in accordance with Table 4.

Value	Meaning
--011111b	Protocol Data – Soft Reset command

Table 4 Soft Reset – PCB Master to Slave

The bits b0 to b5 set to 011111b indicate the Soft Reset command.

The bits b6 and b7 of the PCB of the Soft Reset command are RFU.

The master device SHALL NOT assume the slave device to have successfully executed a soft reset after the Soft Reset command has been acknowledged.

The master device SHALL wait at least t_{RSTG} (see chapter 21) after the end of the Stop Condition of the Soft Reset command before sending any other command on the bus.

The master device SHALL send the Read Answer to Reset as next command after a Soft Reset command to the slave device and only after successful execution of the Read

Answer to Reset command the master device SHALL assume the slave device has reset all its non-persistent information.

The master device SHALL do $N_{RETRY,SRST}$ (number of retries soft reset) attempts to reset the slave device in case the slave does not acknowledge its slave address before raising a Protocol Error. This retry limit SHALL NOT apply in the case that the previous command was the last command of a chain of Master to Slave Data Transmission commands.

The slave device SHALL be able to receive a Soft Reset command when its current state is the READY as well as when its current state is the ACTIVE state.

The slave device SHALL change its state to the READY state when receiving a Soft Reset command and its current state is either the READY or the ACTIVE state.

The PCB returned by the slave device SHALL have a value in accordance with Table 5.

Value	Meaning
00000000b	RFU

Table 5 Soft Reset – PCB Slave to Master

The bits b0 to b7 of the PCB returned by the slave device are RFU.

The slave device SHALL NOT return any data bytes within the command (except the PCB itself). The slave device SHALL set the LEN field to 01h accordingly.

The slave device SHALL be ready to receive the next command within a time of t_{RSTG} (see chapter 21) after the end of the Stop Condition of the Soft Reset command.

Note: Acknowledged as used above means in this case that the slave device has acknowledged the address byte as well as the single data byte of the Soft Reset command.

13.6 Read Answer to Reset

The Read Answer to Reset command SHALL be based on the Read block packet.

The master device SHALL issue the Read Answer to Reset command to check if the slave device is in READY state and to read the slave device information contained therein.

The PCB sent by the master device SHALL have a value in accordance with Table 6.

Value	Meaning
--101111b	Protocol Data – Read Answer to Reset command

Table 6 Read Answer to Reset – PCB Master to Slave

The bits b0 to b5 set to 101111b indicate the Read Answer to Reset command.

The bits b6 and b7 of the PCB send by the master device are RFU.

The master device SHALL do $N_{RETRY,RATR}$ (number of retries read answer to reset) attempts in case the slave does not acknowledge its slave address before raising a Protocol Error. This retry SHALL include a retry of the previous sequence or command in case it was a Cold Reset, Soft Reset or Wakeup. The retry limit SHALL NOT apply in the case that the previous command was the last command of a chain of Master to Slave Data Transmission commands.

The slave device SHALL only acknowledge the command when in READY state.

The slave device SHALL change state to the ACTIVE state when receiving a Read Answer to Reset command and its current state is the READY state.

The PCB returned by the slave device SHALL have a value in accordance with Table 7.

Value	Meaning
00000000b	RFU

Table 7 Read Answer to Reset – PCB Slave to Master

The bits b0 to b7 of the PCB returned by the slave device are RFU.

The slave device SHALL return its Answer to Reset data within the data field of the command in accordance with the definition in chapter 15.

A maximum of $CDB_{ATS,MAX}$ (Command Data Bytes, Answer To Reset, MAXimum) are allowed to be returned as data bytes by the slave device.

The master device SHALL only apply and use the information returned by the slave device within its Answer to Reset data for commands following the Read Answer to Reset command.

13.7 Parameter Exchange

The Parameter Exchange command SHALL be based on the Read block packet.

The master device MAY NOT implement the Parameter Exchange command.

The master device MAY issue the Parameter Exchange command to either indicate the support of number of data bytes in a Master to Slave Data Transmission command above $CDBI_{SM,DEF}$ or to check for number of data bytes in a Slave to Master Data Transmission command above $CDBI_{MS,DEF}$ supported by the slave device.

The master device SHALL preferably issue the Parameter Exchange command after the Read Answer to Reset command and before any other command.

The PCB sent by the master device SHALL have a value in accordance with Table 8.

Value	Meaning
--111111b	Protocol Data – Parameter Exchange command
YY------b	YYb codes $CDBI_{SM,MAX}$

Table 8 Parameter Exchange – PCB Master to Slave

The bits b0 to b5 set to 111111b of the PCB send by the master device indicate the Parameter Exchange command.

The bits b6 and b7 of the PCB send by the master device code the $CDBI_{SM,MAX}$ (Command Data Bytes Integer, Slave to Master, MAXimum) value. Based on the announced $CDBI_{SM,MAX}$ value the slave device MAY send up to $CDB_{SM,MAX}$ (Command Data Bytes, Slave to Master, MAXimum) bytes according to Table 9 within following Block read packet.

$CDBI_{SM,MAX}$	$CDB_{SM,MAX}$
00b	29
01b	61
10b	125
11b	253

Table 9 Maximum Data Bytes – Slave to Master

The $CDB_{SM,MAX}$ as indicated via its $CDBI_{SM,MAX}$ by the master device overrides the $CDB_{SM,DEF}$ within the slave device. The overwrite stays valid until the slave device enters the READY state.

The master device SHALL be able to receive $CDB_{SM,MAX}$ data bytes in a Slave to Master Data Transmission command as indicated by the $CDBI_{SM,MAX}$ sent in the Parameter Exchange command.

The master device SHALL do $N_{RETRY,PE}$ (number of retries parameter exchange) attempts in case the slave does not acknowledge its slave address before raising a Protocol Error.

The slave device SHALL only acknowledge the command when in ACTIVE state.

The slave device SHALL NOT change its state when receiving a Parameter Exchange command.

The PCB returned by the slave device SHALL have a value in accordance with Table 10.

Value	Meaning
----ZZ--b	ZZb codes $CDBI_{MS,MAX}$
--NN----b	NNb codes the bitwise negated $CDBI_{MS,MAX}$
YY------b	YYb codes $CDBI_{SM,MAX}$ confirmation

Table 10 Parameter Exchange – PCB Slave to Master

The bits b0 and b1 of the PCB returned by the slave device are RFU.

The bits b2 and b3 of the PCB returned by the slave device code the $CDBI_{MS,MAX}$ (Command Data Bytes Integer, Master to Slave, MAXimum) value. Based on the announced $CDBI_{MS,MAX}$ value the master device MAY send up to $CDB_{MS,MAX}$ (Command Data Bytes, Master to Slave, MAXimum) bytes according to Table 11 within following Block write packet.

$CDBI_{MS,MAX}$	$CDB_{MS,MAX}$
00b	32
01b	64
10b	128
11b	253

Table 11 Maximum Data Bytes – Master to Slave

The bits b4 and b5 of the PCB returned by the slave device code the bitwise negated $CDBI_{MS,MAX}$ value.

The bits b6 and b7 of the PCB returned by the slave device code the $CDBI_{SM,MAX}$ confirmation containing the acknowledgement of the $CDBI_{SM,MAX}$ received from the master device.

The value of the $CDBI_{SM,MAX}$ confirmation returned by the slave device SHALL have the same value as the $CDBI_{SM,MAX}$ in the PCB send by the master device.

The slave device SHALL NOT return any data bytes within the command (except the PCB itself). The slave device SHALL set the LEN field to 01h accordingly.

The slave device SHALL be able to receive $CDB_{MS,MAX}$ data bytes in a Master to Slave Data Transmission command as indicated by the $CDBI_{MS,MAX}$ returned in the Parameter Exchange command.

The $CDB_{MS,MAX}$ as indicated via its $CDBI_{MS,MAX}$ by the slave overrides the $CDB_{MS,DEF}$ in the master device. The overwrite stays valid until the master device issues a Power Down for Cold Reset, a Cold Reset or a Soft Reset command to the slave device.

The master device SHALL detect a mismatch between the $CDBI_{SM,MAX}$ and the $CDBI_{SM,MAX}$ confirmation and issue a Protocol Error in case of a mismatch (see chapter 14.2).

The master device SHALL bitwise negate one of the $CDBI_{MS,MAX}$ values received, it SHALL compare the result of the calculation with the other value received, it SHALL detect a mismatch between the resulting values $CDBI_{MS,MAX}$ and the bitwise negated $CDBI_{MS,MAX}$ and it SHALL issue a Protocol Error in case of a mismatch (see chapter 14.2).

After the Parameter Exchange command the master device MAY send up to $CDB_{MS,MAX}$ data bytes in a Data Transmission command.

After the Parameter Exchange command the slave device MAY return up to $CDB_{SM,MAX}$ data bytes in a Slave to Master Data Transmission command.

Note: The $CDBI_{SM,MAX}$ is confirmed back by the slave device to allow the master device to detect communication errors during the exchange of the information.

Note: The $CDBI_{MS,MAX}$ is transferred as bitwise negated value as well to allow the master device to detect communication errors.

Note: It's strongly proposed to implement the Protocol Exchange command in case the master device is I2C compliant and supports a $CDBI_{MS}$ above $CDBI_{MS,DEF}$ as this will allow for an improved performance of the data exchange in case of an high amount of data bytes is to be exchanged.

13.8 Protocol Binding Selection

The Protocol Binding Selection command SHALL be based on the Read block packet.

The master device MAY NOT implement the Protocol Binding Selection command.

The master device SHALL only issue the Protocol Binding Selection command to change the protocol binding to another protocol binding then the default one indicated by the slave device (see chapter 15.2.2).

The PCB sent by the master device SHALL have a value in accordance with Table 12.

Value	Meaning
----0011b	Protocol Data – Protocol Binding Selection command
YYYY----b	YYYYb codes the protocol binding request

Table 12 Protocol Binding Selection – PCB Master to Slave

The bits b0 to b3 set to 0011b indicate the Protocol Binding Selection command.

The bits b4 to b7 of the PCB send by the master device code the protocol binding request containing specific protocol binding to be selected (details see Table 14).

The master device SHALL do $N_{RETRY,PBS}$ (number of retries protocol binding selection) attempts in case the slave does not acknowledge its slave address before raising a Protocol Error.

The slave device SHALL only acknowledge the command when in ACTIVE state.

The slave device SHALL NOT change its state when receiving a Protocol binding Selection command.

The PCB returned by the slave device SHALL have a value in accordance with Table 13.

Value	Meaning
YYYY----b	YYYYb codes the protocol binding confirmation

Table 13 Protocol Binding Selection – PCB Slave to Master

The bits b4 to b7 of the PCB returned by the slave device code the protocol binding confirmation containing the acknowledgement of the protocol binding (details see Table 14).

Value YYYY	Selected protocol
0000	protocol indicated with bit 0 of Support protocol bindings element in the ATR (section 15.2.1)
0001	protocol indicated with bit 1 of Support protocol bindings element in the ATR (section 15.2.1)
0010	protocol indicated with bit 2 of Support protocol bindings element in the ATR (section 15.2.1)
0011	protocol indicated with bit 3 of Support protocol bindings element in the ATR (section 15.2.1)
0100	protocol indicated with bit 4 of Support protocol bindings element in the ATR (section 15.2.1)
0101	protocol indicated with bit 5 of Support protocol bindings element in the ATR (section 15.2.1)
0110	protocol indicated with bit 6 of Support protocol bindings element in the ATR (section 15.2.1)
0111	protocol indicated with bit 7 of Support protocol bindings element in the ATR (section 15.2.1)
1---	RFU

Table 14 Protocol Binding Confirmation code

The value of the protocol binding confirmation returned by the slave device SHALL have the same value as the protocol binding request in the PCB send by the master device.

The bits b0 to b3 of the PCB returned by the slave device are RFU.

The slave device SHALL NOT respond any data bytes within the command (except the PCB itself). The slave device SHALL set the LEN field to 01h accordingly.

The slave device SHALL switch to the protocol binding selected by the master device immediately after the end of the Stop Condition of the protocol binding Selection command.

The slave device SHALL be able to receive a next command in the applied protocol binding within t_{CMDG} after the end of the Stop Condition of the Protocol Binding Selection command.

The master device SHALL detect a mismatch between the protocol binding request and the protocol binding confirmation and it SHALL issue a Protocol Exception in case of a mismatch (see chapter 14.2).

The master device SHALL use the selected protocol binding for any command following the Protocol Binding Selection command.

Note: The master device rules are defined to only select a protocol binding indicated to be supported by the slave device (see chapter 15.2.1). As a result of that a slave device not indicating more protocol bindings then the one activated by default could actually skip the support of this command.

13.9 Status

The Status command SHALL be based on the Read block packet.

The master device SHALL implement the Parameter Exchange command.

The master device SHALL issue the Status command to query the status of the slave device and to check for pending protocol exceptions of the slave device.

The PCB sent by the master device SHALL have a value in accordance with Table 15.

Value	Meaning
----0111b	Protocol Data – Status command

Table 15 Status– PCB Master to Slave

The bits b0 to b3 of the PCB set to 0111b indicate the Status command.

The bits b4 to b7 of the PCB are RFU.

The slave device SHALL acknowledge the command when in ACTIVE state and it MAY acknowledge the command when in PROCESSING state.

The slave device SHALL NOT change its state when receiving a Status command.

The PCB returned by the slave device SHALL have a value in accordance with Table 16.

Value	Meaning
----0111b	Protocol Data – Status command
YYYY----b	YYYYb codes the status code

Table 16 Status– PCB Slave to Master

The bits b4 to b7 of the PCB returned by the slave device code the exception information according to Table 21.

The bits b0 to b3 of the PCB returned by the slave device set to 0111b indicate the Status response.

The slave device SHALL NOT respond any data bytes within the command (except the PCB itself). The slave device SHALL set the LEN field to 01h accordingly.

The master device SHALL raise a protocol exception in case the slave device returns status information different from 0000b and 0001b and SHALL continue as described in chapter 14.1.2.

The slave device SHALL only return its current protocol status exception information and it SHALL NOT reset any pending protocol exception.

The slave device MAY NOT acknowledge the Status command when busy.

Note: The Status command allows the master device to detect error occurred on the slave device as well as to detect a busy but still responsive slave device. In the first case an exception handling will be started whereas in the latter case the master device will continue to wait until a timeout occurs.

13.10 Data Exchange

Two kinds of Data Transmission commands are defined for exchanging data:

- Master to Slave Data Transmission and
- Slave to Master Data Transmission

Both commands are used to exchange Payload Streams and Payload Chunks as defined by the mappings (see chapter 16).

Note: The data transmission commands are used to actually exchange in a fully transparent way the Payload Streams. The Payload Streams are typically APDUs that are exchanged.

13.10.1 Protocol Control Byte

The elements contained within a PCB that SHALL be used within Data Transmission commands are based on Table 17.

Value	Meaning
------00b	Protocol Data – Master to Slave Data Transmission command
------10b	Protocol Data –Slave to Master Data Transmission command
----EE--b	EEb codes the error detection code used
-SSS----b	SSSb codes the sequence counter
M-------b	Mb codes the More bit

Table 17 Data Transmission – Protocol Control Byte

The individual elements of the PCB MAY or MAY NOT be used depending on the communication direction or Data Transmission command used (see chapter 13.10.1.3 and chapter 13.10.4).

The bits b0 and b1 set to 00b indicate a Master to Slave Data Transmission command and the bits b0 and b1 set to 10b indicate a Slave to Master Data Transmission command

13.10.1.1 Error Detection Code

The bits b2 and b3 of the PCB encode the error detection code used as defined in Table 18.

Value	Meaning
00b	No error detection code is used
01b	LRC error detection is used (see chapter 15.1.2)

Table 18 PCB –Error Detection Code

The master device MAY NOT use any error detection code.

The master device SHALL only use error detection codes indicated to be supported by the slave device (see chapter 15.1.2).

The master device sets the used error detection code used for communication with the slave device by its indication in the first Master to Slave Data Transmission command.

The master device SHALL NOT switch between different or no error detection code use during a single session.

13.10.1.2 Sequence counter

The bits b4 to b6 of the PCB of a Master to Slave Data Transmission command as well as a Slave to Master Data Transmission command encode each an independent sequence counter of 3 bits in size.

The master device and the slave device manage each two independent sequence counter.

The master device SHALL implement a sequence counter used within Master to Slave Data Transmission commands called $SC_{M,MS}$ (Sequence Counter Master, Master to Slave) and a sequence counter to be compared with the sequence counter received in Slave to Master Data Transmission commands called $SC_{M,SM}$ (Sequence Counter Master Slave to Master).

The slave device SHALL implement a sequence counter used within Slave to Master Data Transmission commands called $SC_{S,SM}$ (Sequence Counter Slave, Slave to Master) and a sequence counter to be compared with the sequence counter received in Master to slave Data Transmission commands called $SC_{S,MS}$ (Sequence Counter Slave, Master to Slave).

Sequence counter are defined to be able to identify disordered or more specifically missed Data Transmission commands.

Sequence counter SHALL be implemented as a rolling code that starts at zero again when it overruns (after 111b it transitions to 000b at the next increment).

The master device SHALL initialize both of its sequence counter to zero at the activation of a slave device.

The master device SHALL increment $SC_{M,MS}$ at any time it has successfully transmitted a Master to Slave Data Transmission command.

The slave device SHALL initialize both of its sequence counter to 111b when entering the READY state.

The slave device SHALL increment $SC_{S,SM}$ at any time it has acknowledged a Slave to Master Data Transmission command.

Note: Acknowledged as used above means in this case that the slave device has acknowledged the address byte of a Slave to Master Data Transmission command.

13.10.1.3 Data Chaining

The bit b7 of the PCB encodes the More bit.

For the usage of data transmission, for both master to slave and as slave to master communication, the More bit set to 1b indicates that the data bytes of this packet to contain a Payload Chunk and not a complete Payload Stream and so to be continued by a subsequent Payload Chunk in the next Data Transmission command (see chapter 16).

The last Data Transmission command of chain SHALL always have the More bit set to 0b.

The master device and the slave device SHALL use the chaining in case the Payload Stream does not fit into a single Data Transmission Command.

The master device SHALL split the Payload Stream into Payload Chunks so that the individual Data Transmission commands with the More bit set to 1b are filled to its

maximum amount of bytes taking the use of the OPTIONAL error detection code into account.

The slave device MAY typically split the Payload Stream into Payload Chunks so that the individual Data Transmission commands with the More bit set to 1b are filled to its maximum amount of bytes.

13.10.2 Length

The LEN (LENgth) byte indicates the number of bytes to send or receive.

13.10.3 Master to Slave Data Transmission

The Master to Slave Data Transmission command SHALL be based on the Write block packet.

The master device SHALL implement the Master to Slave Data Transmission command.

The master device SHALL use the structure for the Master to Slave Data Transmission command as shown in Figure 5 when no error detection code is used:

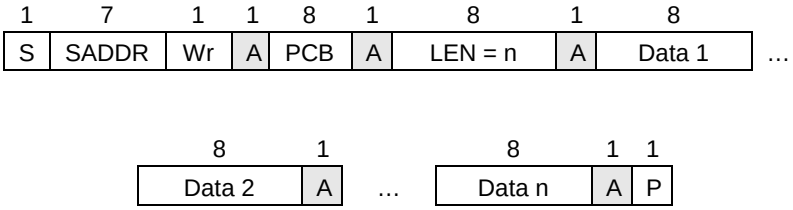


Figure 5 Master to Slave Data Transmission – Structure no EDC

The Master to Slave Data Transmission command SHALL use the structure as shown in Figure 6 when the OPTIONAL error detection code is used:

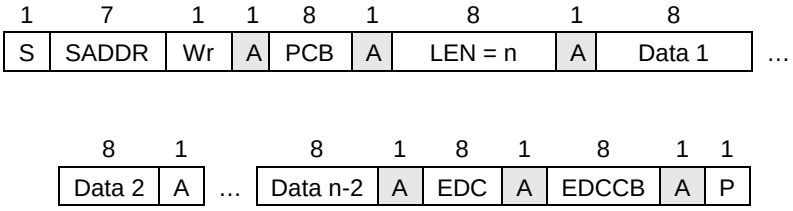


Figure 6 Master to Slave Data Transmission – Structure with EDC

Note: CDB_{MS} equals n in the structure without EDC (Figure 5) and CDB_{MS} equals n-2 in the structure with EDC (Figure 6). Hence, the maximum number of Data bytes ($CDB_{MS,MAX}$) sent by the slave device is the same in both cases.

The master device SHALL issue the command to transmit Payload Streams to the slave device following the rules and order defined below:

1. The master device SHALL send as many Master to Slave Data Transmission commands in a chain indicating data chaining as needed to convey the Payload Stream as defined by the selected mapping and the selected error detection code (see chapter 16).
2. The master device SHALL send (insert) a Status command after each Master to Slave Data Transmission command and if needed retry Status commands failing with an NoAddrAck error for a time of t_{FW} (time, Frame Waiting) + t_{MD} (time, Master Delay) in the following conditions

a. after each Master to Slave Data Transmission command indicating chaining

b. when the complete Payload Stream has been transmitted

3. The master device SHALL analyze the status code received and continue as follows

- The master device SHALL reset its waiting time (to accept waiting for another period of t_{FW}) in case the status code equals 0001b and it has not transmitted the complete Payload Stream.
- The master device SHALL continue sending further data bytes according to step 1 in case the status code equals 0000b and as long as not the complete Payload Stream has been sent
- The master device SHALL start reading a Payload Stream from the slave device by issuing a Slave to Master Data Transmission command (see chapter 13.10.4) in case the status code equals 0000b and the complete Payload Stream has been sent
- The master device SHALL re-transmit the chain of Master to Slave Data Transmission commands transmitted in step 1 since the last Status command has been inserted for $N_{RETRY,DE}$ (number of retries data exchange) times before generating a Protocol Error (see chapter 14.2) in case the status code does not equal 0000b or 0001b

The master device SHALL NOT interlace a chain of Master to Slave Data Transmission commands with any other command then the Status command.

The PCB sent by the master device SHALL be encoded as defined in chapter 13.10.1 and contain $SC_{M,MS}$ as sequence counter.

The master device SHALL respect the maximum number of data bytes being allowed in a single Master to Slave Data Transmission command according to chapter 12.3 and chapter 13.7.

The master device MAY start to poll for a Payload Stream to be available on the slave device by sending a Slave to Master Data Transmission command after the master device has send the last Master to Slave Data Transmission command of a chain.

The slave device SHALL only acknowledge the command when in READY or in ACTIVE state.

The slave device SHALL change state to the ACTIVE state when receiving a Master to Slave Data Transmission command.

The slave device SHALL raise an exception Unexpected Command (see Table 21) if two consecutive Master to Slave Data Transmission commands are received and no exception is pending.

The slave device SHALL accept a sequence counter of 000b within the first PCB received from the master device.

The slave device SHALL be able to analyze the sequence counter received within the PCB from the master device and in case the sequence counter does not equal an increment by one compared to the last sequence counter received the slave device SHALL raise a Protocol Exception and it SHALL expect a sequence counter of 000b within the next PCB received.

The slave device SHALL verify the error detection code, if any, and in case it does not match the value calculated out of the data bytes received it SHALL raise an exception.

The slave device SHALL continue to receive and acknowledge data bytes within a chain of Master to Slave Data Transmission commands even though it MAY NOT expect the amount of data contained to avoid exceptions for reasons other than actual data integrity check failures.

In case no Protocol Exception has been raised the slave device SHALL hand off any complete Payload Stream received to the adjacent higher layer and the slave device SHALL then enter the PROCESSING state.

The slave device MAY NOT acknowledge commands as long as being in PROCESSING state.

The slave device SHALL re-enter the ACTIVE state when a Payload Stream has been made available by the adjacent higher layer to be returned (see chapter 18).

Note: Acknowledged as used above means in this case that the slave device acknowledges the address bytes and all contained data bytes of Slave to Master Data Transmission commands.

Note: The slave device will need to be implemented to avoid buffer overflows created by a master device sending more data bytes then fitting in its Payload Stream buffer.

Note: The slave device typically hands-off Payload Streams to the adjacent higher layer and not individual Payload Chunks as those do only reflect a part of the complete higher layer data. But in case of very long Payload Streams the protocol buffer may not be large enough to fit the whole Payload Stream the slave device may hand off data to the adjacent higher layer after a certain number of data bytes.

13.10.4 Slave to Master Data Transmission

The Slave to Master Data Transmission command SHALL be based on the Read block packet.

The master device SHALL implement the Slave to Master Data Transmission command.

The master device SHALL issue the command to retrieve Payload Streams from the slave device.

The master device SHALL only issue the command after at least one complete Payload Stream has been sent to the slave device using the Master to Slave Data Transmission (see chapter 13.10.2).

The Slave to Master Data Transmission command SHALL use the structure as shown in Figure 7 when no error detection code is used:

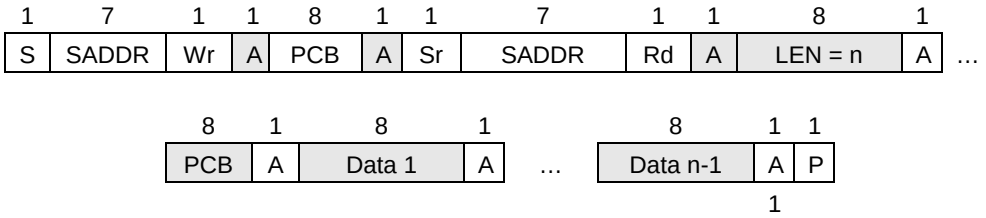


Figure 7 Slave to Master Data Transmission – Structure no EDC

The Slave to Master Data Transmission command SHALL use the structure as shown in the Figure 8 when error detection code is used:

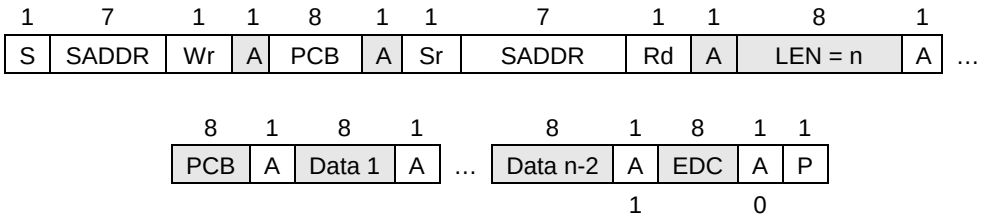


Figure 8 Slave to Master Data Transmission – Structure with EDC

Note: CDB_{SM} equals n-1 in the structure without EDC (Figure 7) and CDB_{SM} equals n-2 in the structure with EDC (Figure 8). Hence, the maximum number of Data bytes ($CDB_{SM,MAX}$) sent by the slave device is the same in both cases.

The PCB sent by the master device SHALL have a value in accordance with Table 19.

`------10b`	Protocol Data – Slave to Master Data Transmission command
`R-------b`	Rb codes the Re-transmission bit

Table 19 Slave to Master Data Transmission – PCB Master to Slave

The master device SHALL set the Re-transmission bit in the PCB to 0b for the first Slave to Master Data Transmission command in a chain.

The master device SHALL indicate a re-transmission of not correctly received Slave to Master Data Transmission commands to the slave device by setting the Re-transmission bit in the PCB send to 1b.

The slave device SHALL be prepared to return a Payload Chunk as many times as the master is requesting it via a Slave to Master Data Transmission command. The slave should confirm it with the bit b7 set to 0b in the return PCB.

The bits b2 to b6 of the PCB sent by the master device are RFU.

The slave device SHALL only acknowledge the command when in ACTIVE state.

The slave device MAY NOT respond to a Slave to Master Data Transmission command when it has not received any complete Payload Stream from the master device before.

The slave device SHALL NOT change its state when receiving a Slave to Master Data Transmission command.

The PCB returned by the slave device SHALL be encoded as defined in chapter 13.10.1 and contain $SC_{S,SM}$ as sequence counter except for the case where the slave device indicated a protocol exception where it indicates this as defined in chapter 14.1.1.

The master device SHALL continue to retry a not acknowledged Slave to Master Data Transmission command for a time of $t_{FW} + t_{MD}$.

The master device SHALL issue a Protocol Error in case it has not got any response after the time $t_{FW} + t_{MD}$.

The master device SHALL accept a sequence counter of 000b within the first PCB received from the slave device.

The master device SHALL be able to analyze the sequence counter received within the PCB from the slave device and in case the sequence counter does not equal an increment by one compared to the last sequence counter received the master device

SHALL issue a Protocol Reset command it SHALL expect a sequence counter of 000b within the next PCB received.

The slave device SHALL indicate the error detection code in the returned PCB according to the error detection code used in the preceding Master to Slave Data Transmission command (see chapter 13.10.1.3).

The slave device SHALL respect the maximum number of data bytes being allowed in a single Master to Slave Data Transmission command according to chapter 12.3 and chapter 13.7.

The master device SHALL be able to detect a protocol exception response returned by the slave device and in such case the master device SHALL reset its sequence counter to 000b and the master device SHALL raise a Protocol Exception.

The master device SHALL continue to send Slave to Master Data Transmission commands as long as the More bit of the PCB returned by the slave device is set to 1b.

The master device SHALL verify the error detection code contained in the current Data Transmission command of a chain, if any, and in case it does not match the value calculated out of the data bytes received from the slave device, The master device SHALL then initiate a retransmission for $N_{RETRY,DE}$ (number of retries data exchange) times before raising a Protocol Error (see chapter 14.2).

Note: The master device will need to be implemented to avoid buffer overflows created by a slave device sending more data bytes then fitting in its Payload Stream buffer.

Note: The master device needs to poll the slave device until a Payload Stream to become available. To make sure that this does not end in a deadlock a maximum time t_{FW} has been defined in which the slave device needs to respond independently if a Payload Stream has become available or not. With that the slave device informs the master device it's still alive but not ready to return a Payload Stream, yet.

Note: For the usage of flow control, only for slave to master communication, the Re-transmission bit is used by the master device to indicate a re-transmission of individual Payload Chunks.

14. Protocol Exception and Error Handling

This chapter defines the exception and error handling of the protocol for both the slave devices and the master devices according to the following two classifications of errors:

- Protocol Exception and
- Protocol Error

14.1 Protocol Exception Handling

The protocol exceptions are split into the exceptions detected by the master device and the one detected by the slave device.

An exception detected by the slave device will be announced to the master device so that after the next command exchange the master is aware of the exception on the slave side as well.

14.1.1 Slave Device detects an Exception

In case a protocol exception is raised within the slave device it SHALL respond to the next command based on a Read block packet a status response using a PCB value as defined in Table 20.

Value	Meaning
----0111b	Status
YYYY----b	Status Code

Table 20 Protocol Control Byte – Status Response

The bits b0 to b3 set to 0111b indicate a status response.

The bits b4 to b7 of the PCB returned by the slave device SHALL code the status code according to Table 21.

Value	Meaning	Remarks
0000b	Normal processing	No exception detected; slave device is ready
0001b	Normal processing	No exception detected, slave device is busy
1000b	No Precise Diagnosis	Exception raised but no further information given
1001b	Over clocking	A too high data rate generated by the master device has been detected (high possibility for a loss of data bits)
1010b	Unexpected Sequence	An invalid I2C sequence involving an unexpected Sr, P, Wr, ACK or NACK condition has been detected
1011b	Invalid Data Length	The number of indicated bytes does not match the actually received number of data bytes or data received does not match multiple bytes of 8 bits
1100b	Unexpected Command	An unexpected (invalid or out of context) command has been received, the sequence counter might be out of sync or too many commands using chaining have been received (buffer overflow)
1101b	Invalid EDC	The EDC verification of the last received command failed
1111	Other Exception	Any other kind of exception not covered by the values in the range from 1001h to 1100h

Table 21 Status Codes

All other values not defined in Table 21 of the status code are RFU.

The slave device MAY return specific status code indications in the status response but doing so the slave device SHALL always return the value 1000h instead of the specific status code allowing the master device to detect that the slave device does not support the specific status code indication.

The slave device SHALL NOT respond any data bytes within the command containing the exception information (except the PCB itself). The slave device SHALL set the LEN field to 01h accordingly.

The slave device SHALL reset a pending exception after a Master to Slave data transmission command or after a Slave to Master transmission command has been answered without any further error.

The slave device MAY overwrite a pending exception in case a new exception is raised in addition.

The slave device SHALL discard all received Payload Chunks in case chaining using Master to Slave Data Transmission commands is active.

Note: The slave needs to discard all Payload Chunks received until the point it detects an exception to enable the master device to restart the communication.

14.1.2 Master Device detects an Exception

In case a protocol exception is raised within the master device it SHALL react according to one the following cases:

- In case chaining using Slave to Master Data Transmission commands is active the master device SHALL initiate a retransmission of the whole chain according to Chapter 13.10.4 for $N_{RETRY,DE}$ (number of retries data exchange) times before raising a Protocol Error.
- In case chaining using Slave to Master Data Transmission commands is not active the master device SHALL retransmit the previous command for $N_{RETRY,DE}$ (number of retries data exchange) times before raising a Protocol Error.

Note: In case the exception is the result of an exception returned by the slave device the previous command as used above is actually the command before the one returning the exception information,

14.2 Protocol Error Handling

In case a Protocol Error is raised within the master device it SHALL abort the communication with the slave device and it SHALL initiate either a Cold Reset or a Soft Reset to the slave device.

Afterwards the master device MAY restart the communication with the slave device.

Note: In this version of the specification there is no requirement for slave devices to raise Protocol Errors.

15. Answer to Reset

The Answer to Reset is returned by a slave device within a Read Answer to Reset command (see chapter 13.6).

The slave device SHALL return the following information contained in data objects within the data bytes as response to the Read Answer to Reset:

- Low level protocol,
- Protocol binding,
- Higher layer,
- Operating system and
- Identification

The master device SHALL be able to decode any kind of data object respectively tag values returned.

The master device SHALL ignore unknown data objects respectively tag values received independently if they are received before, in between or after the herein defined data objects.

The slave device SHALL provide the information as data objects encoded in TLV format (see chapter 1.4.6)

The slave device SHALL return the data objects in the fixed order as listed below in Table 22.

The slave device SHALL always return all REQUIRED data objects even though the actual data contained in the value field of the data objects MAY be empty, though.

The following table defines the data objects respectively tag values used:

Tag Value	Meaning	Remarks
B8h	Low level data object	REQUIRED tag
B9h	Protocol binding data object	REQUIRED tag
BAh	Higher layer data object	OPTIONAL tag
BBh	Operating system data object	OPTIONAL tag
BCh	Identification data object	OPTIONAL tag

Table 22 Read Answer to Reset – Data objects

Note: The slave device is not allowed to leave out any of the as REQUIRED defined data objects. This is done to allow for an optimized implementation on master device side. Anyhow the master device needs to be capable of ignoring data objects received that are unknown according to the rules set. In cases where the individual value field of data objects may be empty or in cases where the data object may not be present (only for OPTIONAL tags) the master device needs to respect the default values as defined for each element within the data objects.

15.1 Low Level Data Object

The low-level data object SHALL be encoded using the tag as defined in Table 22.

The length field SHALL be set to the number of contained bytes according to the number of bytes returned within the value field.

The value field MAY contain the following elements:

- Slave device protocol version
- Error detection code
- Frame waiting integer
- Communication speed

Table 23 defines the offset and size of the elements within the low level data object.

Offset (bytes)	Size (bytes)	Element	Remarks
0	1	Slave device protocol version	REQUIRED field
1	1	Error detection codes	REQUIRED field
2	1	Frame waiting integer	OPTIONAL field
3	1	Communication speed	OPTIONAL field

Table 23 Low Level Data Object – Data Elements

Any additional byte within the low-level data object is RFU.

Note: Slave devices are allowed to return only the maximum amount of bytes as defined in the according data objects. Anyhow, master devices need to ignore additional bytes returned by a slave device according to the RFU rule.

Note: In case a slave device returns additional elements within this data group based on a future version of this specification it will always need to return all values as defined in this version of the specification. This includes the REQUIRED and

OPTIONAL ones as otherwise the separation of the elements would not be possible without ambiguity.

15.1.1 Slave Device Protocol Version

The SDPV (Slave Device Protocol Version) indicates the specification version the slave device is compliant to. The protocol SHALL consist of one byte where the bits b4 to b7 SHALL indicate the major version number SDPVMA (SDPV MAjor), and the bits b0 to b3 SHALL indicate the minor version number SDPVMI (SDPV MInor).

The actual SDPV value is defined in chapter 17.

The slave device SHALL always return this data element.

15.1.2 Error Detection Codes

The slave device SHALL indicate its capability to support an error detection code to check the integrity of Payload Streams and Payload Chunks being exchanged within the data exchange in accordance with Table 24.

Value	Error Detection Codes
00000001b	LRC is supported

Table 24 Data Element – Error Detection Codes

Each bit codes a single error detection code supported. All not defined bits of the error detection code data element are RFU.

The slave device does not support the use of an error detection code in case no bit is set.

The slave device SHALL implement the error detection codes it indicates to support.

The slave device SHALL always return this data element.

15.1.2.1 Longitudinal Redundancy Code

The LRC (Longitudinal Redundancy Code) SHALL be 8 bits (1 byte) in size.

The LRC SHALL be implemented as an exclusive-oring of all data bytes of the command.

The LRC is verified by calculating an exclusive-oring of all data bytes of the command including the LRC value itself and the result SHALL be 00h.

15.1.3 Frame Waiting Integer

The slave device MAY indicate its maximum delay between two acknowledged commands. This maximum delay is called t_{FW} (time, Frame Waiting). The slave device does not indicate the t_{FW} directly but via a Frame Waiting Integer (FWI). The t_{FW} is calculated by the following formula:

$$t_{FW} = 10ms \times 2^{FWI}$$

The slave device SHALL only code a FWI within the range specified in Table 33. All other values are RFU The master device SHALL use the default value FWI_{DEF} for FWI as specified in Table 33 in case the slave device does not return an FWI value.

15.1.4 Communication Speed

The slave device SHALL indicate its capability to support a communication speed above BR_{INIT} called $BR_{S,MAX}$ in accordance with Table 25.

Value	Communication Speed
----0000b	A bit rate of up to 100 kbit/s is supported
----0001b	A bit rate of up to 150 kbit/s is supported
----0010b	A bit rate of up to 200 kbit/s is supported
----0011b	A bit rate of up to 300 kbit/s is supported
----0100b	A bit rate of up to 400 kbit/s is supported
----0101b	A bit rate of up to 1 Mbit/s is supported
----0110b	A bit rate of up to 3.4 Mbit/s is supported
----1111b	No information available

Table 25 Data Element – Communication Speed

The bits b0 to b3 code $BR_{S,MAX}$ (Bit Rate, Slave, MAXimum) supported by the slave device. All bit combinations not defined in Table 25 and the bits b4 to b7 are RFU.

The slave device does not support the use of communication speeds above BR_{INIT} in case the bits b0 to b3 are set to 0000b.

The slave device MAY return this data element.

The master device SHALL only use a communication speed above BR_{INIT} in case the slave device indicates support for $BR_{S,MAX}$ above BR_{INIT}.

The master device SHALL only use a communication speed of up to $BR_{S,MAX}$.

The master SHALL NOT use a communication speed above BR_{INIT} in case the slave returns 0000b as content of the bits b0 to b3.

The master device SHALL use an implementation specific bit rate as $BR_{S,MAX}$ value in case the slave device returns 1111b as content of the bits b0 to b3.

The master device MAY use a communication speed above BR_{INIT} in case the slave device returns 1111b as content of the bits b0 to b3.

The master device SHALL use the default value of 1111b in case the slave device does not return a $BR_{S,MAX}$ value.

Note: The communication speeds supported by the slave device for the coding of 1111b of the bits b0 to b3 are implementation specific. The master device will need to know the maximum bit rate of the specific slave device it's connected to.

15.2 Protocol Binding Data Object

The protocol binding data object SHALL be encoded using the tag as defined in Table 22. Its length field SHALL be set to 02h and the value field SHALL contain the following elements:

- Supported protocol bindings
- Default selected protocol binding

Table 26 defines the offset and size of the elements within the protocol binding and higher layer information data object.

Offset (bytes)	Size (bytes)	Element	Remarks
0	1	Supported protocol bindings	REQUIRED field
1	1	Default selected protocol binding	REQUIRED field

Table 26 Protocol Binding Data Object – Data Elements

Any additional byte within the protocol binding data object is RFU.

Note: Slave devices are defined to only return the amount of bytes as defined in the according data objects. Anyhow, master devices need to ignore additional bytes returned by a slave according to the RFU rule.

15.2.1 Supported Protocol Bindings

The slave device SHALL indicate its capability to support protocol bindings according to Table 27.

Value	Supported Protocol Bindings
00000001b	APDU [7816-3] mapping (see chapter 16)
10000000b	Proprietary mapping (out of scope of this specification)

Table 27 Protocol Binding – Supported Protocol Bindings

Each bit indicates a single binding supported. All other bit combinations not defined in Table 27 are RFU.

The slave device SHALL set at least a single bit indicating to support at least one protocol binding.

The master device SHALL only select a protocol binding indicated to be supported by the slave device.

Note: This version only specifies an APDU mapping in accordance with [7816-3] but thanks to the built-in feature of binding indication and selection future versions of this specification are open to define bindings e.g. for IP or other communication means very openly and with full backwards compatibility to system rolled out with only the APDU [7816-3] mapping.

15.2.2 Default Selected Protocol Binding

The slave device SHALL indicate its default selected protocol binding according to Table 28.

Value	Default Selected Protocol Binding
00000001b	APDU [7816-3] mapping (see chapter 16) by default selected and used
10000000b	Proprietary mapping by default selected and used

Table 28 Protocol Binding – Default Selected

The slave device SHALL set a single bit only indicating the default protocol binding selected by default and used by the slave device.

The master device SHALL use the indicated default selected protocol binding as long as not having switched the slave device to another supported protocol binding.

15.3 Higher Layer Data Object

The higher layer data object SHALL be encoded using the tag as defined in Table 22.

The length field SHALL be set to the number of contained bytes according to the number of bytes returned within the value field.

The value field SHALL contain information related to the default selected and used protocol binding (see Table 28).

The following chapter defines the actual elements returned based on the default selected and used protocol binding.

15.3.1 APDU [7816-3] Mapping related Elements

In case the APDU [7816-3] mapping is selected and used the following elements MAY be returned within the value field of the higher layer data object:

- APDUs supported

Table 26 defines the offset and size of the elements within the higher layer data object.

Offset (bytes)	Size (bytes)	Element	Remarks
0	1	APDUs Support	REQUIRED field

Table 29 Higher Layer APDU [7816-3] Mapping Data Object – Data Elements

Any additional byte within the higher layer data object is RFU.

The slave device SHALL only return the maximum amount of bytes as defined in the data object. Anyhow, master devices SHALL ignore additional bytes returned by a slave according to the RFU rule.

15.3.1.1 APDU Support

The slave device SHALL indicate its capability to support short and extended APDUs within the APDU support data element in accordance with Table 30.

Value	APDU Support
-------0b	Short APDUs supported and extended APDUs not supported
-------1b	Short and extended APDUs supported

Table 30 Data Element – Extended APDUs Supported

The bits b1 to b7 of the extended APDUs supported element are RFU.

The master device SHALL use the default value of 0b in case the slave device does not return an extended APDUs supported value.

15.4 Operating System Data Object

The operating system data object SHALL be encoded using the tag as defined in Table 22.

Its length field SHALL be set in the range from 0 to 15 in accordance to the number of historical bytes returned in the value field.

The value field of the operating system data object SHALL contain up to 15 historical bytes as defined in [7816-4].

The slave device SHALL return any number of historical bytes in the range from 0 to 15.

The master device SHALL be able to accept any number of historical bytes in the range from 0 to 15.

15.5 Identification Data Object

The identification data object SHALL be encoded using the tag as defined in Table 22.

Its length field SHALL be set in the range from 0 to 15 in accordance to the number of bytes returned in the value field.

The value field of the identification data object SHALL contain up to 15 bytes of identification.

The slave MAY return a human readable ASCII coded character string, a serial number or any other slave device specific and proprietary information in the Identification data object.

The slave device SHALL return any number of bytes in the range from 0 to 15.

The master device SHALL be able to accept any number of historical bytes in the range from 0 to 15.

The master device SHALL NOT assume to receive an ASCII coded NULL terminated character string.

Note: To avoid runtime issues a master devices implementation should never assume that a character string returned is NULL terminated. So it should always add a NULL termination when the data is copied and used as a string component.

16. Higher Layer Data Mapping

This chapter defines the mappings conveying a data block of the adjacent higher layer using the protocol as defined in this document.

The complete data block of the adjacent higher layer as a whole is called Payload Stream.

A Payload Stream MAY be split into multiple so-called Payload Chunks.

In case a complete Payload Stream does not fit into a single Data Transmission command (see chapter 13.10) under use of the OPTIONAL error detection code it will be split into multiple pieces of Payload Chunks and being transmitted in individual Data Transmission commands using the chaining feature (see chapter 13.10.1.3).

The master device indicates the use of an error detection code in the PCB of a Master to Slave Data Transmission command and in such case it SHALL add an EDC followed by an EDCCB as defined in chapter 16.2 and chapter 16.3.

The slave device responds in a Slave to Master Data Transmission using the identical error detection code as used by the master in its previous Master to Slave Data Transmission command by adding an EDC as defined in chapter 16.2.

The receiving entity removes the OPTIONAL error detection code from the received Payload Chunks and combines the contained higher layer data again.

The combination of all higher layer data contained in the received Payload Chunks results in the complete original Payload Stream again (see 13.10.1.3).

This version of the specification includes a mapping of APDUs according to [7816-3]. The actual structure of the APDUs itself is defined in [7816-4].

Based on the mapping indication and selection feature future versions of the specification are open to include other mappings like IP or other communication protocols.

The following chapters describe in full detail how to construct Payload Streams.

Note: The receiving entity is the master device in case of a Slave to Master Data Transmission and the slave device in case of a Master to Slave Transmission.

16.1 APDU [7816-3] Mapping

The APDU [7816-3] mapping SHALL use the “Command-response pair transmission by T=1” as defined in [7816-3] and map it to the higher layer data.

A complete C-APDU (Command APDU) stream SHALL be mapped with its first byte to data byte 0 of the higher layer data. The second byte SHALL be mapped to data byte 1 of the higher layer data and so on.

A complete R-APDU (Response APDU) stream SHALL be mapped with its first byte to data byte 0 of the higher layer data. The second byte SHALL be mapped to data byte 1 of the higher layer data and so on.

16.2 Error Detection Code Value

The EDCV (Error Detection Code Value) SHALL be calculated over all single bytes of a Payload Chunk as contained in a single Data Transmission command.

This SHALL be done starting with the data byte transmitted/received first up to the last data byte transmitted/received.

For the LRC error detection code the resulting 8-bit result value is called (EDC) Error Detection Code and SHALL be contained in the EDC field of the Data Transmission command.

The slave device MAY use byte level clock stretching after receiving the EDCV to be able to verify the EDCV.

16.3 Error Detection Code Check Byte

For the use of sending the error detection code from the master device to the slave device the actual result value of the calculation SHALL be appended by a so called EDCCB (Error Detection Code Check Byte).

The EDCCB SHALL always have a value of 00h and all other values of the EDCCB are RFU.

The EDCCB is acknowledged or not acknowledged by the slave device dependent if it was able to correctly verify or was not able to correctly verify the EDCV.

Note: According to the RFU rules the actual value of the EDCCB is not intended to be checked by the slave device.

17. Version Treatment

This chapter describes the version treatment of master devices and slave devices.

The MDPV (Master Device Protocol Version) indicates the specification version the master device is compliant to. The MDPV SHALL consist of one byte where the bits b4 to b7 SHALL indicate the major version number MDPVMA (MDPV MAjor), and the bits b0 to b3 SHALL indicate the minor version number MDPVMI (MDPV MInor).

Master devices compatible with version of the specification SHALL indicate a value of 0001b as major version number and 0000b as minor version number of its MDPV.

Slave devices compatible with version of the specification SHALL indicate a value of 0001b as major version number and 0000b as minor version number of its SDPV.

The master device SHALL identify the specification version the slave device implements based on the Answer to Select returned by the slave device (see chapter 13.6).

The master device SHALL be able to differentiate the cases as defined in Table 31 and the master device SHALL react as defined in the handling rules accordingly.

Case	Handling Rules	Remark
MDPVMA is equal to SDPVMA and MDPVMI is equal to SDPVMI	The master device SHALL accept to communicate with the slave device	The master device and slave device are fully compatible
MDPVMA is equal to SDPVMA and MDPVMI is bigger than SDPVMI	The master device SHALL accept to communicate with the slave device according to the implemented feature set of the slave device	The master device and slave device are fully compatible whereas the master device might implement additional features the slave device does not implement. The master device will only use the feature set of the slave device.
MDPVMA is equal to SDPVMA and MDPVMI is lower than SDPVMI	The master device SHALL accept to communicate with the slave device according to the implemented feature set of the master device	The master device and slave device are fully compatible whereas the slave device might implement additional features the master device does not implement. The master device will not be able to use any additional feature of the slave device
MDPVMA is smaller than SDPVMA	The master device SHALL NOT accept to communicate with the slave device	The master device and slave device are not compatible. The master device implements a previous incompatible version only
MDPVMA is bigger than SDPVMA	The master device MAY accept to communicate with the slave device according to the implemented feature set of the slave device	The master device and slave device are not compatible. The slave device implements a previous incompatible version only. Nevertheless, in case the master device implements an additional matching version of an older specification it is able to use the feature set of the slave device

Table 31 Version Treatment

Note: A minor version number increase assumes new features to be included or only minor changes to the overall specification have been applied that in no way harm the full backwards and forward compatibility between master devices and slave devices.

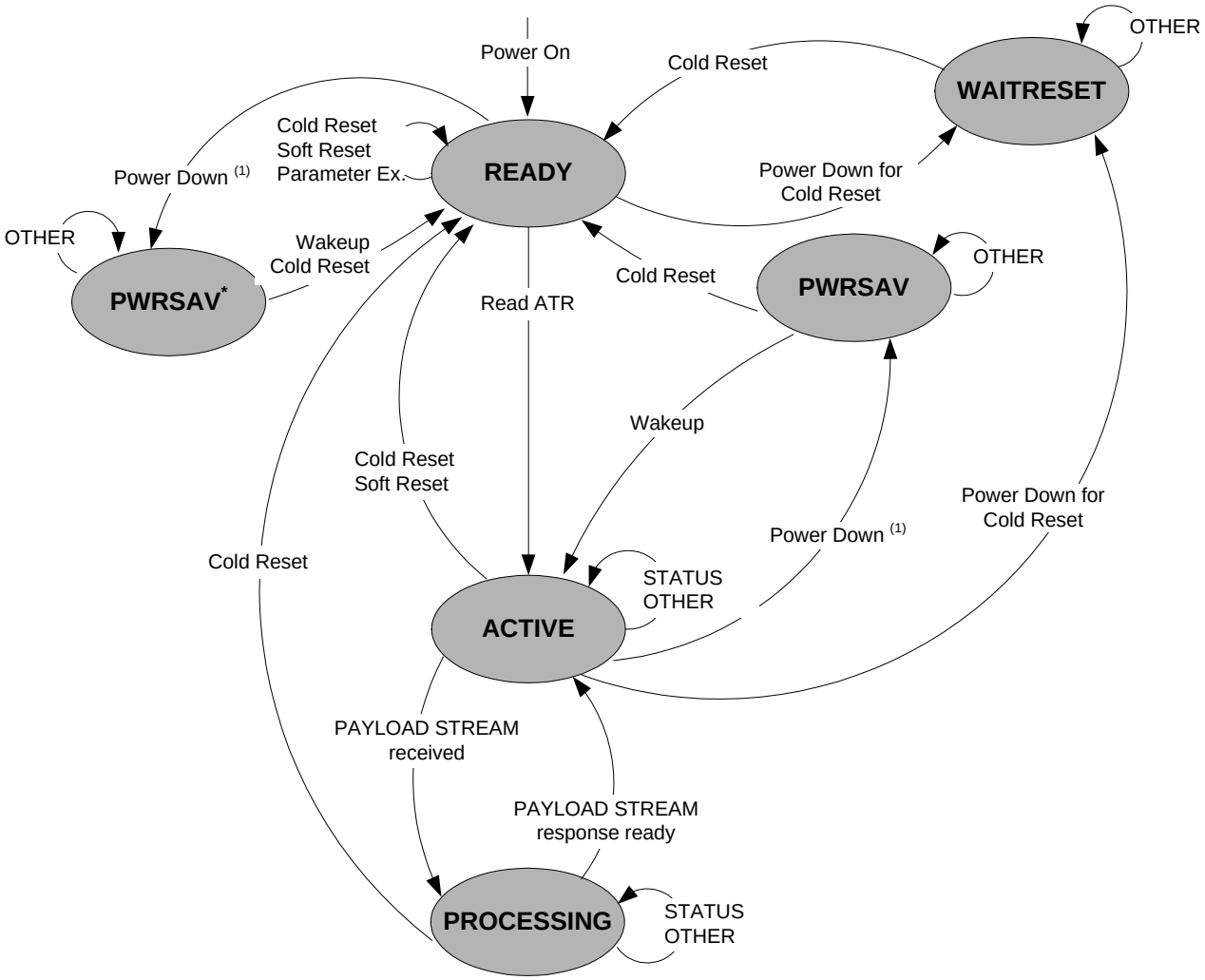
Note: A Major version number increase assumes new features to be included and major changes to the overall specification have been applied that do harm the full backwards compatibility between master devices and slave devices.

Note: For the definition of full backwards and forward compatibility both the compatibility between an older slave device talking to a newer slave device as well as a newer master device talking to an older slave device are to be taken into account.

18. State Diagram

The slave device SHALL maintain a state machine as defined by the state diagram below:

Figure 9 State Diagram – Slave Device



(1) Slave Devices implementing an automatic power down may enter the PWRSAV state or the PWRSAV * state without receiving a Power Down command.

The Slave device SHALL remain in its current State if not otherwise mentioned.

19. Protocol Control Byte Overview

Table 32 provides an overview of the defined PCB as defined within this specification. The PCB contains protocol block type indications, codes commands exchanged as well as its parameter (if any).

PCB	Block Type	Command	Remarks
-------0b	*Data Exchange*		*The bit b0 set to 0b is used to detect the block type.*
------00b	Data Transmission	Master to Slave	The bit b1 set to 0b codes the command.
------10b	Data Transmission	Slave to Master	The bit b1 set to 1b codes the command.
------01b	*Unused*		*The bits b0 and b1 set to 01b are used to detect the block type.*
------11b	*Protocol Data*		*The bits b0 and b1 set to 11b are used to detect the block type.*
--001111b	Protocol Data	Wakeup	The bits b2 to b5 set to 0011b code the command. The bits b6 and b7 are RFU.
--011111b	Protocol Data	Soft Reset	The bits b2, to b5 set to 0111b code the command. The bits b6 and b7 are RFU.
--101111b	Protocol Data	Read Answer to Reset	The bits b2, to b5 set to 1011b code the command. The bits b6 and b7 are RFU.
YY111111b	Protocol Data	Parameter Exchange	The bits b2 to b5 set to 1111b code the command. The bits b6 and b7 are used to code the supported block size.
YYYY0011b	Protocol Data	Protocol Binding Selection	The bits b2 and b3 set to 00b code the command. The bits b4 to b7 are used to code the protocol binding selected.
YYYY0111b	Protocol Data	Status	The bits b2 and b3 set to 01b code the command. The bits b4 to b7 are used to code the status code.
----1011b	Protocol Data	Unused	The value b2 and b3 set to 10b and the bits b4 to b7 are RFU.

Table 32 Protocol Control Byte – Overview

Note: The Power Down as well as the Power Down for Cold Reset commands are not covered as they are using the Quick command packet where the data direction bit is used to code the actual command a no PCB is send by the master device.

20. Data integrity, Security Recommendations

20.1 Managing data integrity

The protocol has been designed for the use in devices that are working on a bus that has typically quite short distances between the devices and is normally implemented on printed circuit boards where shielding between signals is fairly easily possible. As a result of that the probability of disturbances and bit errors is rather low. Using that assumption, the protocol has been designed following the design assumptions of [I2C-bus] and [SMBUS] to not secure the actual protocol data being exchanged.

Nevertheless, the protocol has been designed allowing an OPTIONAL integrity check of higher layer data being exchanged. The principle foreseen in [SMBUS] adds an integrity check per data packet exchanged. This seemed not efficient for the underlying use case of exchanging APDUs as mapping those to typically small data packets as defined within [SMBUS] results in the exchange of multiple data packets per single APDU. Besides that, the PEC as defined within [SMBUS] is based on a CRC8 which itself seemed not suitable for the big data packet sizes and the expected APDU sizes this protocol has been designed for.

So an OPTIONAL integrity check has been defined that is based on a CRC16 applied on whole APDU to be exchanged that itself MAY then be split over multiple data packets exchanged.

20.2 Managing the end of an application context

Once a transaction with an application on the slave device has been finished the adjacent higher layer needs to decide how far security relevant information is available within the slave device.

An application on master device side therefore needs to define in which way a slave device is left alone by

- just stop communicating,
- switch the device into the PWRSAV state, or
- reset the device

Note: Even though this specification does not enforce a certain behavior it's recommended for the master device to explicitly reset all non-persistent information inside the slave device context information by issuing a Cold Reset or Soft Reset after the end of a transaction.

20.3 Application influence of Power Saving Modes

The sequence of a Power Down command, slave device in PWRSAV state and a Wakeup command does not influence any non-persistent state inside the slave device. This means that any possibly open transaction, application communication as well as possibly open secure channel and security states are not influenced. Whereas using the sequence of a Power Down for Cold Reset command, slave device in PWRSAV state and followed by the thereafter only valid Code Reset clears all such non- persistent states inside the slave device. Nevertheless, it's not recommended to use the Power Down for Cold Reset to ensure that non persistent states are cleared after an application transaction ends as the actual clearing does only happen when the next application transaction starts and during that time the states are maintained. In such cases either the

Cold Reset sequence (if supported by the slave device) or a Soft Reset command should preferably be used by a master device implementation.

21. Values

Throughout the document symbols are used to identify the values of parameters. The actual values of the parameters are listed in this chapter. For some of the parameters, a minimum and maximum value is defined. Other parameters are defined by a single value.

Table 33 contains the definition of all values:

Parameter	Master Device			Slave Device		
	Min		Max	Min		Max
SADDR				1001000b		1001011b
BR_{INIT}	10 kbit/s		100 kbit/s	10 kbit/s		-
$BR_{S,MAX}$				-		3.4 Mbit/s
FWI				0		9
FWI_{DEF}					9	
t_{RSTL}	150 µs			100 µs		
t_{RSTG}	5 ms					4.9 ms
t_{CMDG}	180 µs					175 µs
t_{PDE}						150 µs
t_{FW}				10 ms		5120ms
$t_{FW,DEF}$					5120 ms	
t_{WNCMD}			200 ms			
t_{MD}	1 ms		50 ms			
t_{SD}				0 ms		
$CDB_{ATS,MAX}$						29
$CDB_{MS,DEF}$		32			32	
$CDB_{SM,DEF}$		29			29	
$CDB_{MS,MAX}$	32		255	32		255
$CDB_{SM,MAX}$	29		252	29		252
$N_{BUF,S}$				32		65535
$N_{RETRY,SRST}$	0		3			
$N_{RETRY,PD}$	0		3			
$N_{RETRY,PDCR}$	0		3			
$N_{RETRY,W}$	0		3			
$N_{RETRY,RATR}$	0		3			
$N_{RETRY,PE}$	0		3			
$N_{RETRY,PBS}$	0		3			
$N_{RETRY,DE}$	1		3			

Table 33 Values

22. Examples (INFORMATIVE)

This chapter contains informative examples of typical command exchanges. The examples are based on real data of a possible implementation. The Payload Streams exchanged in the data exchange example are based on the assumption of using C-APDUs as well as R-APDUs as higher layer data being exchanged.

The examples shown use 1001000b as the address of the slave device.

22.1 Master Device activates

In this example the master device will start the communicate issuing a Wakeup command, followed by a Soft Reset command and finally a Read Answer to Reset command.

After those commands have been exchanged successfully the master device is able to exchange a first Payload Stream.

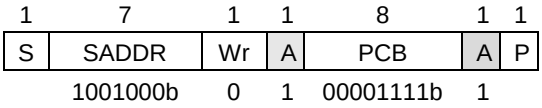


Figure 10 Example – Wakeup command

Note: In the above case the slave device does not acknowledge its SADDR. Anyhow according to the rules defined the master device assumes that the slave left the PWRSAV state.

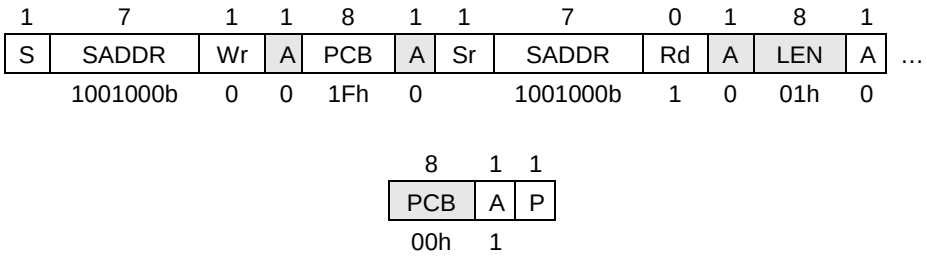
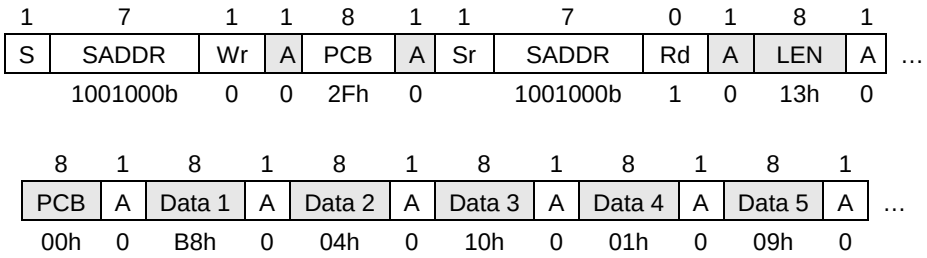


Figure 11 Example – Soft Reset command

Note: In case the slave device would not acknowledge its first SADDR of the Soft Reset Command the master device would retry sending the Wakeup command.

Note: Instead of a Wakeup command followed by a Soft Reset command the master device may also apply a Cold Reset sequence followed by issuing the Read Answer to Reset command thereafter.



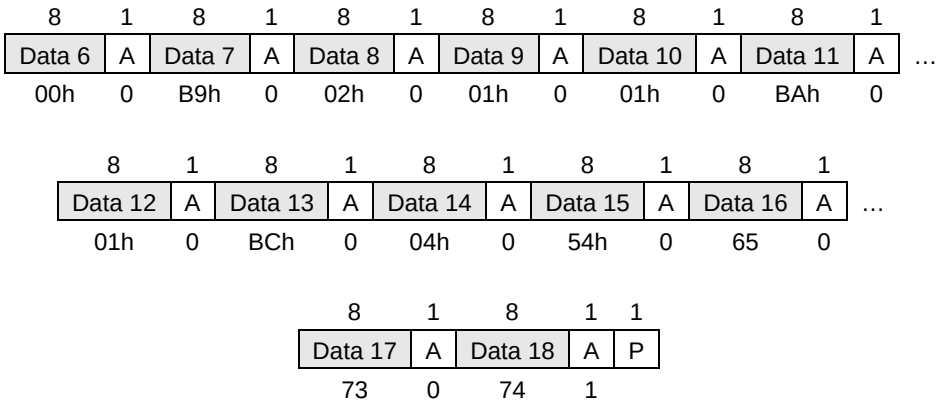


Figure 12 Example – Read Answer to Reset command

In the above example the slave device returns the following information:

- Low level information
    - o Slave device protocol version
      SDPVMA = 1
      SDPVMI = 0
    - o Error detection codes
      LRC is supported
    - o Frame Waiting Integer
      FWI = 09h
    - o Communication speed
      A bit rate of up to 100 kbit/s is supported
- Protocol binding and higher layer information
    - o Supported protocol bindings
      APDU [7816-3] mapping
    - o Default selected protocol binding
      APDU [7816-3] mapping
- Higher Layer Data information
    - o APDU [7816-3] mapping
      Short and extended APDUs supported
- Operating system information
    - o No information provided
- Identification information
    - o ASCII text “Test”
      (no NULL termination)

Note: In case the slave device would not acknowledge its first SADDR of the Read Answer to Reset Command the master device retries combination of Soft Reset and Read Answer to Reset commands.

22.2 Master Device exchanges Parameter

In this example the master device issued a Parameter Exchange command to indicate its maximum amount of data bytes supported in a single Data Transmission command and

at the same time read the maximum amount of data bytes supported in a single Data Transmission command by the slave device.

After this command has been exchanged successfully the master device as well as the slave device are both able to send the maximum number of blocks as indicated by the other device in following Data Transmission commands.

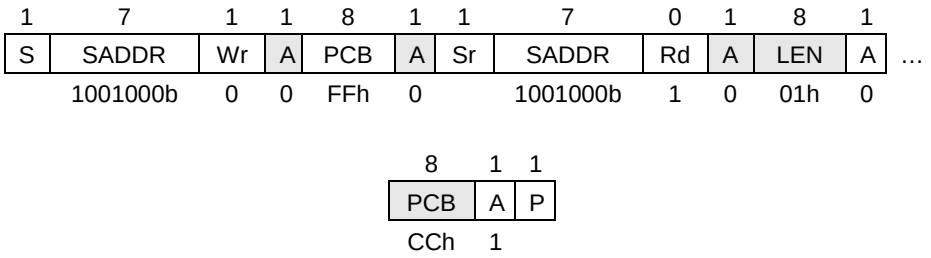


Figure 13 Example – Parameter Exchange command

In the above example the master device indicates support for a $CDB_{SM,MAX}$ of 252 bytes and the slave indicates support for a $CDB_{MS,MAX}$ of 255 bytes.

22.3 Master Device exchanges an APDU.

In this example the master device sends a C-APDU, checks the status until a response is received and finally reads the R-APDU out of the slave device.

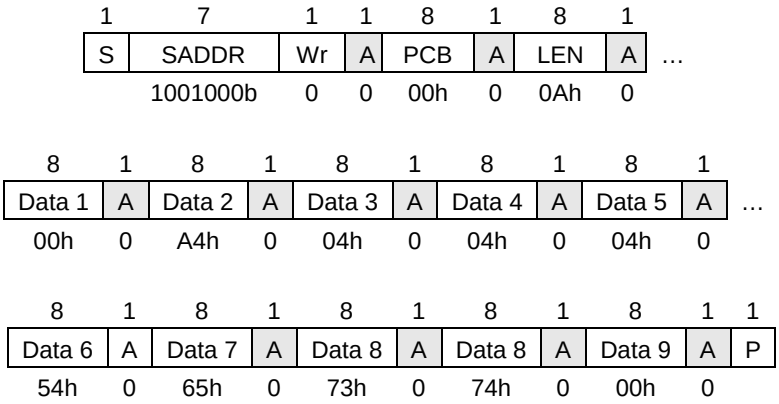


Figure 14 Example – Master to Slave Data Transmission command

The C-APDU as contained in the example above is a select by name command. Chaining is not used, no error detection code used and the sequence counter is 000b as it's the first command after activating the slave device.

- Select by Name
    - Header (CLA, INS, P1, P2): 00h A4h 04h 04h
    - Length in (Lc): 4
    - Data (AID): "Test"
    - Length out expected (Le): 00

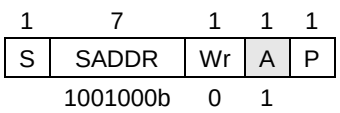


Figure 15 Example – Status command (not acknowledged)

In this example the slave device does not answer to the first Status command issued by the master device as the slave device is busy with the execution of the previously received C-APDU. Therefore, the master continues to address the slave device by re-issuing the last command again (possibly multiple times depending on the time of execution needed by the slave device and depending on the delay until re-issuing a command by the master device).

1	7	1	1	1	1	1	7	0	1	8	1	8	1	1
S	SADDR	Wr	A	PCB	A	Sr	SADDR	Rd	A	LEN	A	PCB	A	P
	1001000b	0	0	07h	0		1001000b	1	0	01h	0	07h	1	

Figure 16 Example – Status command (acknowledged)

Based on the returned status 0000b coded within the PCB returned the slave device indicated its successful executed of the C-APDU received and availability of an R-APDU.

As a result the master reads the R-APDU.

1	7	1	1	1	1	1	7	0	1	8	1	8	1	
S	SADDR	Wr	A	PCB	A	Sr	SADDR	Rd	A	LEN	A	PCB	A	…
	1001000b	0	0	02h	0		1001000b	1	0	03h	0	02h	0	

8	1	8	1	
Data 1	A	Data 2	A	P
6Ah	0	82h	1	

Figure 17 Example – Slave to Master Data Transmission command (acknowledged)

The slave device returns an R-APDU in the example above:

- Status word
  6482h (File or application not found)

22.4 Master Device deactivates

In this example the master device ends a communication with a slave device by initiating a Soft Reset command, followed by a Read Answer to Reset command and finally a Power Down command.

1	7	1	1	8	1	1	7	0	1	8	1	
S	SADDR	Wr	A	PCB	A	Sr	SADDR	Rd	A	LEN	A	…
	1001000b	0	0	1Fh	0		1001000b	1	0	01h	0	

8	1	1
PCB	A	P
00h	1	

Figure 18 Example – Soft Reset command

1	7	1	1	8	1	1	7	0	1	8	1	
S	SADDR	Wr	A	PCB	A	Sr	SADDR	Rd	A	LEN	A	…
	1001000b	0	0	2Fh	0		1001000b	1	0	11h	0	

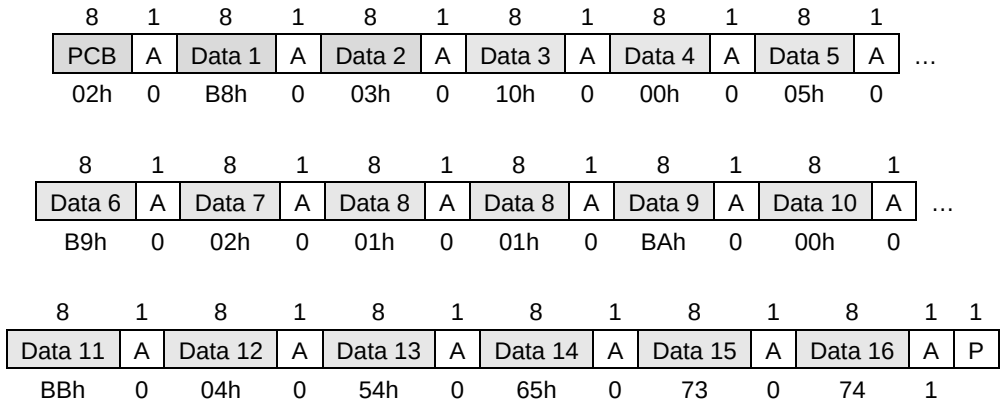


Figure 19 Example – Read Answer to Reset command

The content of the response from the slave device is don't care in this case as it's only issued to test that really a Soft Reset as issued before has been executed.

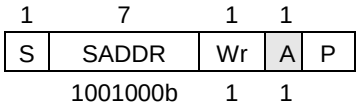


Figure 20 Example – Power Down command

23. Stack Layering (Informative)

This chapter contains informative figures on the stack layering of this specification and the relation to other specifications. They provide a quick overview and easy understanding of the typical defined layers as used within smart cards vs. this specification.

Figure 21 shows the layering compared to the well-known T=1 and T=CL protocols used within smart card systems.

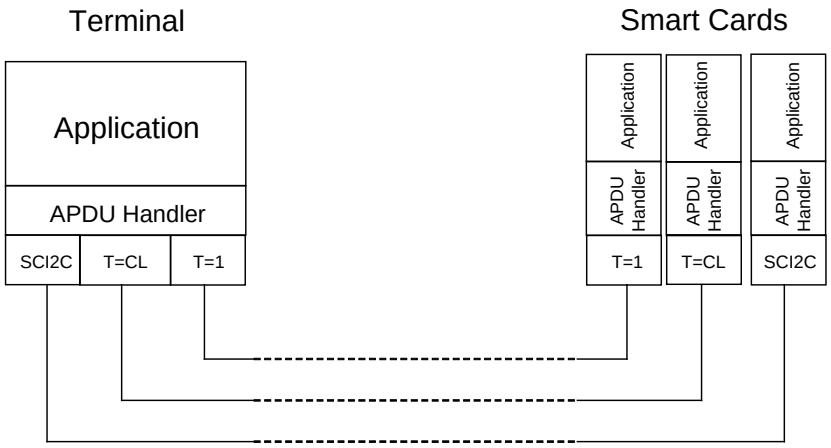


Figure 21 SCI2C vs. T=CL and T=1

The SCI2C protocol as specified in this specification is designed to work as a transparent data exchange protocol being able to convey any kind of information. It allows for protocol specific control data exchange for the following

- Lower layer
    - the physical interface and data link layer according to I2C
    - network layer according to SMBus
- Binding between lower layer and higher layer
- Higher layer
    - APDU [7816-3] mapping

This allows for a very flexible adjustment of layer dependent parameters and values and it also allows for an independency of layer specific indications.

Figure 22 shows the layering of the S2C protocol as specified within this specification including the defined APDU [7816-3] Mapping as well as an example for a possible but not yet defined IP mapping vs. other mappings possible.

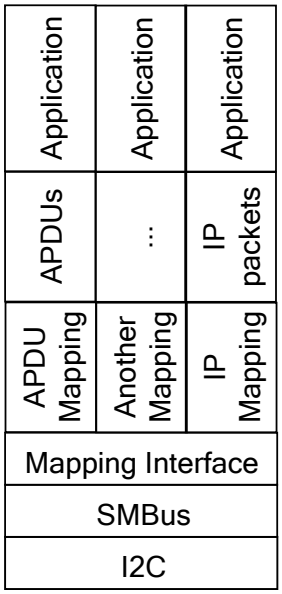


Figure 22 Layering of SCI2C

24. I2C address (Informative)

The I2C base address is always configured via the FABKEY during the production (EEPROM setting). By default, this address is set to 48h (1001000b).
Some devices MAY support an extended address selection via GPIO lines, e.g. the base address and the base_address+2 can be selected.

If another base address (e.g. 0x94) is needed, the EEPROM setting has to be modified during production.

25. Legal information

25.1 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.

25.2 Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at 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.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

25.3 Trademarks

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

I²C-bus — logo is a trademark of NXP B.V.

26. Contents

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

Date of release: 11 June 2018
Document identifier: AN12207

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

Date of release: 11 June 2018
Document identifier: AN12207