Smart Secure Platform (SSP);
SPI interface
(Release 15)
7.6.1 MCT LPDU structure .................................................................................................................. 26
7.6.2 MCT_DATA from master .............................................................................................................. 27
7.6.3 MCT_DATA from slave ................................................................................................................ 28
7.6.4 MCT activation procedure ......................................................................................................... 29
7.7 SHDLC LLC definition ......................................................................................................................... 30
7.7.1 SHDLC overview .......................................................................................................................... 30
7.7.2 Endpoints ..................................................................................................................................... 30
7.7.3 Flow control.................................................................................................................................. 30
7.7.3.1 Overview .................................................................................................................................. 30
7.7.3.2 Flow control based on SHDLC ................................................................................................ 30
7.8 Power management ......................................................................................................................... 30
7.8.1 Power saving mode ...................................................................................................................... 30
7.8.2 Conditions for entering power saving mode .............................................................................. 31
7.8.2.1 Slave entering power saving mode .......................................................................................... 31
7.8.2.2 Master entering power saving mode ......................................................................................... 31
7.8.3 Resuming from power saving mode ........................................................................................... 31
7.8.3.1 Resuming the slave from power saving mode ....................................................................... 31
7.8.3.2 Resuming the master from power saving mode .................................................................... 32

Annex A (informative): Slave SPI interface states electrical description .............................................. 33
A.1 Slave SPI interface for 5 wire interface .......................................................................................... 33
A.1.1 Slave SPI 5 wire interface diagram ............................................................................................... 33
A.1.2 Slave SPI 5 wire interface states electrical description ..................................................................... 33
A.1.3 Slave SPI 4 wire interface diagram ............................................................................................... 34
A.1.4 Slave SPI 4 wire interface states electrical description ..................................................................... 34

Annex B (informative): Change history .................................................................................................. 36
History ...................................................................................................................................................... 37
Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for ETSI members and non-members, and can be found in ETSI SR 000 314: “Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards”, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Card Platform (SCP).

The contents of the present document are subject to continuing work within TC SCP and may change following formal TC SCP approval. If TC SCP modifies the contents of the present document, it will then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x  the first digit:

0  early working draft;
1  presented to TC SCP for information;
2  presented to TC SCP for approval;
3  or greater indicates TC SCP approved document under change control.

y  the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z  the third digit is incremented when editorial only changes have been incorporated in the document.

Modal verbs terminology

In the present document "shall", "shall not", "should", "should not", "may", "need not", "will", "will not", "can" and "cannot" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"must" and "must not" are NOT allowed in ETSI deliverables except when used in direct citation.
1 Scope

The present document describes the SPI interface for the communication of an SSP, as defined in ETSI TS 103 666-1 [1] using the SCL protocol.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

- In the case of a reference to a TC SCP document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 666-1: "Smart Secure Platform (SSP); Part 1: General characteristics".
[2] ETSI TS 102 613: "Smart Cards; UICC - Contactless Front-end (CLF) Interface; Physical and data link layer characteristics".
[3] ISO/IEC 13239: "Information Technology -- Telecommunications and information exchange between systems -- High-level Data Link Control (HDLC) procedures".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

- In the case of a reference to a TC SCP document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 102 216: "Smart cards; Vocabulary for Smart Card Platform specifications".
3  Definition of terms, symbols and abbreviations

3.1  Terms

For the purposes of the present document, the terms given in ETSI TR 102 216 [i.1] and the following apply:

**data transfer**: information exchange during an SPI access between the master and the slave with SPI_MISO driven by the slave and SPI_MOSI driven by the master while the master is toggling the SPI_CLK signal

**flow control**: mechanism of the Data Link Layer that consists of methods applied by the transmitter in order to send at any time a number of logical data units that can be accepted by the receiver

**frame**: link layer data structure consisting of a prologue or frame header, payload and epilogue or trailer usually containing the CRC bytes

**MAC access request**: request from the slave to the master for a data transfer, i.e. a MAC phase initiated by the slave

**MAC phase**: initiation of a data transfer by the master and/or request for a data transfer by the slave

**SPI access**: SPI_NSS assertion by the master, if not already asserted in the MAC phase, followed by SPI_CLK start for transferring a certain number of bytes according to the SPI master configuration

**NOTE**: The number of bytes transferred during an SPI access is always the same in both directions on SPI_MISO and SPI_MOSI and is also referred to as access length.

**window size**: maximum number of logical data units that can be sent from the transmitter to the receiver without any link layer acknowledgements for any of these data units

**NOTE**: The length of a window size slot equals the Data Link Layer MTU.

3.2  Symbols

Void.

3.3  Abbreviations

For the purposes of the present document, the following abbreviations apply:

A  Asserted
AC  Alternating Current
ACT  Activation
CLF  ContactLess Frontend
CLT  ContactLess Tunnelling
CMD  Command
CPHA  Clock Phase
CPOL  Clock Polarity
CRC  Cyclic Redundancy Check
D  Driven (either Low Level or High Level)
DA  De-Asserted
DC  Direct Current
HiZ  High Impedance
II  Input Ignored
IL  Input Listened
IO  Input/Output
IOH  High Output Current (Output current corresponding to VOH)
IOL  Low Output Current (Output current corresponding to VOL)
LLC  Logical Link Control
LPDU  Link Protocol Data Unit
4 Introduction

The Serial Peripheral Interface (SPI) is a serial synchronous full-duplex communication interface between a single master and one or more slaves present on the same SPI bus, each slave being selected at one time by a dedicated SPI_NSS signal. This clause defines the physical, MAC and data link layers for the SPI interface.

In this clause the terms master and slave refer respectively to the terms master SPI and slave SPI.

5 SCL Under-Layers Protocol Stack

Figure 5.1 illustrates the protocol stack below the SCL supporting the SPI interface.
6 Electrical interfaces

6.1 Introduction

In the clauses below, different implementations of SPI interface are defined. These implementations allow bi-directional communication and the possibility for the slave to initiate communication with the master when it has data available thus avoiding the necessity for continuous polling to be performed by master.

Slave may initiate communication to send a command without a prior command from master.

6.2 Physical interface with 5 signals

Figure 6.1 illustrates the SPI electrical interface using 5 signals.
This SPI interface describes two sets of signals:

- The generic and legacy SPI interface using the 4 signals:
  - SPI_MOSI (Master Output Slave Input);
  - SPI_MISO (Master Input Slave Output);
  - SPI_CLK (clock);
  - SPI_NSS signal used for the selection of a Slave Endpoint among N slaves sharing the same bus;
  - SPI_MISO, SPI_MOSI and SPI_CLK can be shared between several SPI slaves present on the same SPI bus.

- The SPI_INT signal allows the slave to initiate a MAC access request in order to notify the master to start a data transfer.

SPI_INT is defined as an edge-triggered interrupt. It is asserted on the rising edge of the signal.

SPI_NSS is considered active or asserted at low voltage level.

### 6.3 Physical interface with 4 signals

Figure 6.2 illustrates the SPI interface using 4 signals, bi-directional SPI_NSS.

![SPI electrical interface with 4 signals, bi-directional SPI_NSS](image)

Figure 6.2: SPI electrical interface with 4 signals, bi-directional SPI_NSS

The SPI interface with 4 signals describes two sets of signals:

- The three generic and legacy SPI signals as SPI_MOSI (Master Output Slave Input), SPI_MISO (Master Input Slave Output) and SPI_CLK (clock). These signals can be shared between several SPI slaves as a bus.

- The SPI_NSS (Negative Slave Select) signal used for the selection of a slave endpoint among N slaves sharing the same bus and for the slave to initiate a MAC access request to notify the master to initiate a data transfer. SPI_NSS is considered active or asserted at low voltage level. SPI_NSS requires a bidirectional IO implementing an Open Drain (OD) interface for both master and slave. This configuration allows driving the SPI_NSS signal to low voltage level by both master and slave without electrical conflict.

A pull-up resistor keeps SPI_NSS at high state level (i.e. idle state) when SS_MO and SS_SO are not asserted. The SPI_NSS signal is at low state when either SS_MO or SS_SO are asserted.
NOTE: The current industry de-facto SPI specification defines SPI_NSS signal as unidirectional, driven by the master. However, in the present document the SPI_NSS in the 4 signals configuration is bidirectional.

### Table 6.1: Definition of the signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS_MO</td>
<td>Internal master output signal for SPI_NSS assertion. SS_MO is at high state level for generating a SPI_NSS signal assertion (i.e. low level state)</td>
</tr>
<tr>
<td>SS_SO</td>
<td>Internal slave output signal for SPI_NSS assertion. SS_SO is at high state level for generating a SPI_NSS signal assertion (i.e. low level state)</td>
</tr>
<tr>
<td>SS_MI</td>
<td>Internal master input signal indicating SPI_NSS status. SS_MI is at high state level when the SPI_NSS signal is not asserted</td>
</tr>
<tr>
<td>SS_SI</td>
<td>Internal slave input signal indicating SPI_NSS status. SS_SI is at high state level when the SPI_NSS signal is not asserted</td>
</tr>
<tr>
<td>SPI_NSS</td>
<td>SPI_NSS signal: low state level when asserted</td>
</tr>
</tbody>
</table>

### 6.4 Electrical characteristics

#### 6.4.1 DC characteristics

The SPI Electrical specification interface is defined for VDD operational voltage classes B and C as defined in ETSI TS 103 666-1 [1], clause 6.2.2.3. An implementation shall support at least one of these voltage classes.

NOTE: The negotiation of the voltage class between the master and the slave is not defined in the present document.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Note/Test condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input high voltage</td>
<td>VIH</td>
<td>$0.7 \times VDD$</td>
<td>VDD + 0.5 V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input low voltage</td>
<td>VIL</td>
<td>-0.5</td>
<td>0.3 $\times VDD$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output high voltage</td>
<td>VOH</td>
<td>$0.9 \times VDD$</td>
<td>V</td>
<td>IOH = -100 uA</td>
<td></td>
</tr>
<tr>
<td>Output low voltage</td>
<td>VOL</td>
<td>0.1 $\times VDD$</td>
<td>V</td>
<td>IOL = 1.0 mA</td>
<td></td>
</tr>
<tr>
<td>SPI_NSS Low Level Output current (see note)</td>
<td>IOL</td>
<td>-1</td>
<td>-</td>
<td>mA</td>
<td>VOL = 0.3 V</td>
</tr>
<tr>
<td>Maximal SPI_NSS line capacitance (see note)</td>
<td>Cl</td>
<td>-</td>
<td>20 pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Applicable for the physical interface with 4 signals.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
<th>Note/Test condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input high voltage</td>
<td>VIH</td>
<td>$0.7 \times VDD$</td>
<td>VDD + 0.3 V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input low voltage</td>
<td>VIL</td>
<td>-0.3</td>
<td>0.3 $\times VDD$</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output high voltage</td>
<td>VOH</td>
<td>$0.9 \times VDD$</td>
<td>V</td>
<td>IOH = -100 uA</td>
<td></td>
</tr>
<tr>
<td>Output low voltage</td>
<td>VOL</td>
<td>0.1 $\times VDD$</td>
<td>V</td>
<td>IOL = 1.0 mA</td>
<td></td>
</tr>
<tr>
<td>SPI_NSS Low Level Output current (see note)</td>
<td>IOL</td>
<td>-1</td>
<td>-</td>
<td>mA</td>
<td>VOL = 0.3 V</td>
</tr>
<tr>
<td>Maximal SPI_NSS line capacitance (see note)</td>
<td>Cl</td>
<td>-</td>
<td>20 pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Applicable for the physical interface with 4 signals.

The value of the resistor R in figure 6.2 shall be selected for a resultant maximum current lower than or equal to the minimum between the absolute IOL values of the master and the slave.

#### 6.4.2 Data transfer mode, AC characteristics

The SPI interface shall implement the SPI mode 0 according to the industry de-facto SPI specification.
SPI mode 0 is determined by CPOL = 0 and CPHA = 0 where:

- CPOL: defines the SPI_CLK idle state.
- CPOL = 0 implies that the SPI_CLK is at input low voltage while it is idle.
- CPHA: defines the data sampling time.
- CPHA = 0 implies that data sampling is done on the rising edges of the SPI_CLK for both SPI_MISO and SPI_MOSI.

SPI_NSS is considered active or asserted at low voltage level.

Data availability timings with reference to SPI_CLK are shown in figure 6.3.

![Figure 6.3: SPI timing diagram](image)

Table 6.4: AC characteristics for voltage classes B and C (SPI Slave, Mode 0: CPOL = 0, CPHA = 0)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Value (MIN values unless MAX specified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fCLK</td>
<td>SPI_CLK frequency</td>
<td>Max SPI_CLK is specified by Slave at initialization in MCT_READY</td>
</tr>
<tr>
<td>tCLKL</td>
<td>SPI_CLK low time</td>
<td>0,45 × tCLK</td>
</tr>
<tr>
<td>tCLKH</td>
<td>SPI_CLK high time</td>
<td>0,45 × tCLK</td>
</tr>
<tr>
<td>tsu</td>
<td>Data setup time to clock rising edge</td>
<td>5 ns</td>
</tr>
<tr>
<td>tH</td>
<td>SPI_MOSI hold time/Data hold time to clock rising edge</td>
<td>3 ns</td>
</tr>
<tr>
<td>tHO</td>
<td>SPI_MISO hold time/Output hold time to clock falling edge</td>
<td>0 ns</td>
</tr>
<tr>
<td>tCSS</td>
<td>SPI_NSS setup time (class C)</td>
<td>63 ns</td>
</tr>
<tr>
<td>tCSS</td>
<td>SPI_NSS setup time (class B)</td>
<td>33 ns</td>
</tr>
<tr>
<td>tCSSH</td>
<td>Hold time clock falling edge to SPI_NSS inactive (class C)</td>
<td>0,5 × tCLK</td>
</tr>
<tr>
<td>tCSS</td>
<td>SPI_NSS setup time (class B)</td>
<td>30 ns</td>
</tr>
<tr>
<td>tCSV</td>
<td>SPI_MISO valid delay time from SPI_NSS active (class C)</td>
<td>58 ns (MAX)</td>
</tr>
<tr>
<td>tCSV</td>
<td>SPI_MISO valid delay time from SPI_NSS active (class B)</td>
<td>28 ns (MAX)</td>
</tr>
<tr>
<td>tV</td>
<td>SPI_MISO valid delay time from clock falling edge (class C)</td>
<td>0 ns (MIN) 0,7 × tCLKL (MAX)</td>
</tr>
<tr>
<td>tCSDO</td>
<td>SPI_MISO Output disable time from SPI_NSS inactive (class C)</td>
<td>60 ns (MAX)</td>
</tr>
<tr>
<td>tV</td>
<td>SPI_MISO valid delay time from clock falling edge (class B)</td>
<td>0 ns (MIN) 0,7 × tCLKL (MAX)</td>
</tr>
<tr>
<td>tCSDO</td>
<td>SPI_MISO Output disable time from SPI_NSS inactive (class B)</td>
<td>30 ns (MAX)</td>
</tr>
</tbody>
</table>

The values indicated in table 6.4 are reference values for generic SPI slaves. If the concrete slave device supports better timing parameters, a system design may choose to configure the master for these timing values in order to achieve better performance.
6.5 Slave state

6.5.1 Slave state definitions

The slave shall be in one of the following states:

Initial state:

The slave enters this state as soon as it is powered on and VDD is valid or after a reset. In this state, the slave is not initialized and the SPI module is not ready to send or receive any data. The master shall not perform any SPI access while the slave is in this state.

After the POT time, following VDD valid or a reset, the slave shall transition to the state configured/de-selected.

Configured state:

The slave in configured state has two sub states:

- **De-selected sub-state:** In this state, the slave SPI module is enabled, i.e. it shall not ignore SPI_NSS assertion by master and shall be ready for an SPI access. SPI_NSS is not asserted by the master. As a consequence, the slave shall have SPI_MISO in high impedance and shall ignore both SPI_CLK and SPI_MOSI. In an implementation with 4 signal interface, SPI_NSS is not asserted by the slave.

  In this state, the slave is waiting for a master access. The slave will also be in this state after issuing a MAC access request, until the master generates an access for the data transfer.

  From this state, the slave shall enter into one of the states: configured/selected, pro-active or power saving mode; or initial in case of a reset.

- **Selected sub-state:** In this state, the master has asserted the SPI_NSS during the MAC phase and data transfer phase.

  The slave SPI module is enabled, i.e. it shall not ignore SPI_NSS assertion by the master and shall not ignore SPI_CLK and SPI_MOSI. SPI_MISO is not in high impedance.

  From this state, the slave shall enter into one of the states: configured/de-selected or pro-active; or initial in case of a reset.

Pro-active state:

This state is entered by the slave when data need to be sent. The slave issues a MAC access request by asserting the SPI_NSS or SPI_INT. The slave enters this state for the duration of T2, as described in clauses 7.2.3.2 or 7.2.4.3. The slave exits this state on its own after T2, regardless of the input signals states driven by the master.

- For the 4 signal interface:

  The slave shall have its SPI module disabled, i.e. all input signals shall be ignored and SPI_MISO shall be in high impedance. The slave asserts SPI_NSS.

  It is implementation-dependent if the slave SPI module keeps its internal configuration during T2 duration or requires a (partial) re-configuration when exiting this state, which has to be performed within T1 Slave Ready Time reported to the master.

- For the 5 signal interface:

  The slave SPI module needs not to be disabled.

  It is implementation-dependent if the slave SPI module keeps its internal configuration during T2 or needs a (partial) re-configuration when exiting the pro-active state, which has to be performed within T1 Slave Ready Time.

For both the 4 signal and 5 signal implementations, the slave shall transition from the pro-active state to:

- the state configured/de-selected if only the slave generated a MAC access request and the master did not initiate any MAC phase simultaneously;
- the state configured/selected when both the master and the slave generate a MAC phase simultaneously;
- the initial state in case of a reset.

Busy state:

This is an optional state applicable only for the case of the 4 wire interface. It is implementation-dependent if the slave SPI module keeps its internal configuration and/or stays enable during this state.

This state is entered by the slave when the slave performs the optional slave driven flow control by keeping SPI_NSS asserted following the configured/selected state, as described in clause 7.2.4.5.

From this state the slave shall enter the configured/de-selected state.

Power saving mode state:

This state is entered by the slave to reduce the power consumption. Entry and exit conditions are described in clause 7.8.

The slave shall transition from this state to the state configured/selected after the master has resumed the slave; or to the initial state in case of a reset.

6.5.2 Slave state diagram

Figure 6.4 shows the diagram of the slave states.

![Diagram of the slave states](image-url)

**Figure 6.4: Diagram of the slave states**

NOTE 1: For the sake of readability, the transitions to the initial state if a reset occurs are not shown.
NOTE 2: Additional information on slave state is provided in annex A.

7 Data Link Layer

7.1 Overview


7.2 MAC Layer

7.2.1 Overview

The MAC phase is the initial handshake phase between the SPI master and the SPI slave followed by the SPI data transfer phase.

7.2.2 Timing

7.2.2.1 Timing definitions

Table 7.1 describes the timing parameters of the MAC layer. In addition to these MAC timings, the timing requirements listed in AC electrical characteristics shall apply for all MAC diagrams in the following clauses.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Value (min)/Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Slave Ready Time</td>
<td>Reported in MCT_READY</td>
<td>MAC phase time prior to data transfer start. This time is needed for the SPI slave to be ready for the data transfer and is reported in MCT_READY.</td>
</tr>
<tr>
<td>T2</td>
<td>Slave Request Time</td>
<td>1 μs</td>
<td>SPI_NSS or SPI_INT assertion min pulse width.</td>
</tr>
<tr>
<td>T3</td>
<td>Slave Resume Time from power saving mode</td>
<td>Reported in MCT_READY</td>
<td>Time from SPI_NSS assertion by master for slave resume to data transfer start. T3 value shall be used by master in MAC phase instead of T1 when the slave is in power saving mode.</td>
</tr>
</tbody>
</table>

7.2.2.2 T1 = Slave Ready Time

T1 is the MAC phase time required by the slave to get configured and enabled at the end of the MAC phase, ready for the data transfer phase start (i.e. when the SPI_CLK can be started by master).

T1 is defined from the leading edge of the SPI_NSS or SPI_INT assertion by either master or slave to the data transfer phase start.

The data transfer phase at the end of T1 is started by the master by asserting SPI_NSS, if not already asserted depending on the prior MAC phase, followed by the SPI_CLK start.

T1 is slave implementation dependant.

T1 is determined by the slave and includes T2 and any slave-specific internal latencies. Slave shall provide a T1 value that covers the worst-case time it needs between the moment of sampling SPI_NSS to the time when slave becomes ready for the data transfer.

Master shall allow at least the time T1 requested by the slave between the start of the MAC phase initiated by either master or slave and the point in time when the data transfer phase starts. Master may use a higher value than T1, and it may vary T1 from one MAC phase to another.
7.2.2.3 T2 = Slave Request Time

T2 is the duration of an SPI_NSS or SPI_INT pulse generated by the slave for a MAC access request. The minimum value of T2 is 1 μs.

In order to sense the interrupts originating either from slave SPI_NSS or SPI_INT assertion, the master shall be configured for edge-triggered interrupts.

NOTE: The leading edges of the MAC access request signals of the slave should assert the internal interrupt of the master.

7.2.2.4 T3 = Slave resume time from power saving mode

T3 is the slave resume time from the power saving mode. It is slave implementation dependant. Slave reports at initialization in MCT_READY the minimum value of T3 required for the slave to become ready for the SPI access. Whenever master needs to resume the slave from power saving mode it should use during the MAC phase at least the time T3 instead of the MAC SLAVE_READY_TIME T1.

7.2.3 5 signals MAC layer

7.2.3.1 Initiation of the data transfer from the master

In this case at the start of a MAC phase master asserts the SPI_NSS. Slave may also assert the SPI_INT for making a MAC access request as described in clause 7.2.3.3.

Figure 7.1 illustrates the initiation of the data transfer by the master.

![Figure 7.1: Initiation of the data transfer from the Master](image)

The master shall run the following procedure:

1) Master asserts SPI_NSS then goes to the step 2.

2) Master waits for min T1 seconds then goes to the step 3. In use-cases when it is certain slave is not expected to initiate a MAC access request by SPI_INT assertion (e.g. at first access during SPI initialization) master may skip this step.

3) Master starts the bidirectional data transfer by toggling the SPI_CLK signal.

4) Master de-asserts SPI_NSS after data transfer completion i.e. SPI_CLK stopped.

7.2.3.2 Initiation of the data transfer from the slave

Figure 7.2 illustrates the initiation by the slave of a MAC access request for a data transfer to be performed by the master.
Figure 7.2: Initiation of the data transfer from the slave

The slave runs the following procedure:

1) If the slave determines that the SPI_NSS signal is de-asserted (i.e. at the high level state) by reading SPI_NSS_SI then the slave goes to the step 2.

2) The slave asserts SPI_INT by generating an SPI_INT pulse with a minimum width of T2 seconds then goes to the step 3.

3) Depending on its implementation, the slave configures its SPI module. The slave waits for data transfer from the master.

4) As a consequence of SPI_INT assertion by the slave at step 2, the master starts data transfer at a time greater than T1 following the leading edge of SPI_INT, by asserting SPI_NSS and then starts SPI_CLK. At data transfer completion, the master enters step 5.

5) After data transfer completion and SPI_CLK stop, the master de-asserts SPI_NSS.

7.2.3.3 Simultaneous initiation of a data transfer from both master and slave

Figure 7.3 illustrates a simultaneous initiation from the master and the slave.

Both endpoints request a data transfer and run simultaneously their respective procedures. From the master perspective the resulting procedure is equivalent to the initiation from the master.

The slave makes a MAC access request by asserting SPI_INT according to the procedure described in clause 7.2.3.2 and waits for the master to generate the access for data transfer.

The gap time T1 - T2 shall be long enough for the slave to prepare the SPI module for the data transfer.
7.2.3.4 MAC activation

The MAC activation procedure at power on shall be the following:

- Master shall set the SPI_NSS output to the de-asserted state.
- The master shall drive the VDD power line on.

When the power supply of the slave needs to be toggled independently from the power of other devices sharing the SPI bus and while VDD is off or not valid, the slave shall keep SPI_NSS, SPI_CLK, SPI_MOSI and SPI_MISO lines as inputs or in high impedance.

7.2.3.5 MAC deactivation

The MAC deactivation procedure for power off shall be the following:

- Master shall set the SPI_NSS output to the de-asserted state.
- The master shall drive the VDD power line off.

7.2.4 4 signals MAC layer

7.2.4.1 Introduction

As a slave SPI module is normally always ready for an access from the master when selected by SPI_NSS assertion, the slave configures its SPI module in disabled state before asserting SPI_NSS for a MAC access request. This is necessary in order to avoid self-selection e.g. driving MISO in contention with MISO signals of other slaves potentially selected at the same time on the same bus.

7.2.4.2 Initiation of the data transfer from the master

Figure 7.4 illustrates the initiation of the data transfer by the master.

![Figure 7.4: Initiation of the data transfer from the master](image)

The Master shall perform the following procedure:

1) The master checks the state of the SPI_NSS signal (by reading SS_MI) and if it is de-asserted (i.e. at the high state) the master goes to the step 2 otherwise loops on the step 1.

2) The master asserts SS_MO (to drive SPI_NSS signal to the asserted state) then goes to the step 3.

3) The master waits for T1 seconds then goes to the step 4. If the master is certain that the slave will not initiate a MAC access request by asserting SPI_NSS then the master may skip this step.
4) The master starts the bidirectional data transfer by toggling the SPI_CLK signal.
5) After data transfer completion i.e. SPI_CLK stop, the master de-asserts SPI_NSS.

### 7.2.4.3 Initiation of the data transfer from the slave

Figure 7.5 illustrates the initiation of the data transfer from the slave.

![Figure 7.5: Slave initiates a MAC access request](image)

The slave runs the following procedure:

1) If the SPI_NSS signal is found de-asserted i.e. at the high level state by reading SS_SI then the slave goes to the step 2.
2) The slave disables its SPI module then goes to the step 3.
3) The slave asserts SS_SO to drive SPI_NSS to the asserted state (i.e. low level) for at least T2 seconds then goes to step 4.
4) The slave enables its SPI module and waits for the master to initiate the data transfer.
5) The SS_MI signal generates an internal interrupt for the master, triggered on SPI_NSS falling edge. This interrupt initiates a data transfer procedure: the master asserts SS_MO after at least T1 following the SPI_NSS assertion by the slave for the MAC access request.
6) SPI_CLK starts after the SPI_NSS assertion by the master in the previous step, data transfer is performed.
7) After data transfer completion i.e. SPI_CLK stop, the master de-asserts SPI_NSS.

### 7.2.4.4 Simultaneous initiation of the data transfer from both master and slave

Figure 7.6 illustrates a simultaneous initiation from the master and the slave.
Both endpoints initiate a MAC phase for data transfers and run simultaneously their respective procedures. From a master perspective the resulting procedure is equivalent to the initiation from the master. The slave initiates a MAC access request and waits for the access start from the master as per the timings defined. The time $T_1 - T_2$ shall be long enough for the slave to enable the SPI module.

### 7.2.4.5 Slave-driven Flow Control

In certain use-cases and depending on the SPI module design, a slave may assert SPI_NSS (via its SS_SO) during a data transfer in progress for flow control e.g. when the slave needs to delay a subsequent master access as the slave estimates it may not be ready to receive or transmit data. It is assumed that the slave has already prepared data to be sent (if available) for the current data transfer before the data transfer started. Such an assertion of SS_SO while the master access is in progress is not a slave MAC access request and a slave shall not start to send a new frame in the middle of the current access, as a new frame shall be always aligned with the start of an SPI access.

As long as the SPI_NSS signal is asserted and even if the data transfer is completed, the master shall not run the MAC initiation by the master procedure as defined in clause 7.2.4.2.

Slave may assert SS_SO for flow control at any time between the start of the data transfer and SPI_NSS de-assertion by master at end of the data transfer.

NOTE: Detection by the slave of a data transfer for initiating flow control could be performed either by sensing the assertion of the SPI_NSS by master or by detection of the first bytes transferred.

If the SPI bus is shared among multiple slaves and the slave performing flow control has its SPI module enabled during this time, the master shall not initiate a data transfer to any other slaves on the shared SPI bus as long as the slave keeps the SPI_NSS asserted. The slave informs the master about the applicability of this requirement in bit 4 of the slave capabilities in MCT_DATA, as shown in table 7.9.

The slave should not use that mechanism longer than 500 µs; after that duration, the master may use its own recovery mechanisms.
Figure 7.7 illustrates the waveforms for this procedure.

7.2.4.6 MAC activation

The MAC activation procedure at power on shall be the following:

- The SS_MO shall be de-asserted (i.e. at low level) setting the SPI_NSS output to high impedance.
- The master shall drive the VDD power line on.

When the power supply of the slave needs to be toggled independently from the power of the other devices sharing the same SPI bus and while VDD is off or not valid, the slave shall keep SS_SO de-asserted and SPI_CLK, SPI_MOSI and SPI_MISO as inputs or in high impedance.

7.2.4.7 MAC deactivation

The MAC deactivation procedure for power off shall be the following:

- The SS_MO shall be de-asserted (i.e. at low level) setting the SPI_NSS output to high impedance.
- The master shall drive the VDD power line off.

7.3 Link Layer Frame

7.3.1 Overview

Master and slave exchange frames. The format of the frames generated by master and slave is determined by the link layer and it is shown in figure 7.8.
All bytes shall be transmitted with MSB first (most significant bit first). All frames are transferred with the bytes in the order shown in figure 7.8, starting with the LPDU Length. The LPDU field is transferred starting with the LLC Control byte followed by the data generated by the upper OSI layer.

The link layer frame shall contain the following fields:

- LPDU length: length of the LPDU, 1 byte.
- LPDU (Link Protocol Data Unit): LPDU includes the LLC control byte as defined in clause 7.4.
- CRC informs about the integrity of the whole frame i.e. Length and LPDU. Detection of errors in a frame shall be based on the 16-bit frame checking sequence as given in ISO/IEC 13239 [3]. The CRC polynomial is: \( X^{16} + X^{12} + X^{5} + 1 \). Its initial value is 'FFFF'.

Link Layer Frames are exchanged between SPI master and slave during SPI accesses. The SPI master determines the number of bytes exchanged in an SPI access. The maximum length of an access is MTU.

Link Layer frames (including header, LPDU and trailer) shall always be prepared with length less than or equal to MTU.

MTU values are negotiated at SPI interface initialization as described in clause 7.6. The resultant MTU shall be the smallest MTU value between the MTU of the master and the MTU of the slave. The MTU shall be the same irrespective of the transfer direction.

Non-Significant Data (NSD) may be appended at the end of a master or slave link layer frame until the end of the SPI access according to the rules described below, considering LPDU Length + 3 + NSD length ≤ MTU.

NSD shall consist of idle bytes set to the value 'FF' sent by:

- A master while retrieving a slave frame and not sending any frame.
- A slave while receiving a frame from master and not sending any frame.

Master frames, slave frames or remaining bytes of a slave frame (i.e. in a second SPI access for retrieving a slave frame) shall always start aligned on the first bytes transmitted on SPI_MOSI and SPI_MISO at SPI_CLK start.

The LPDU Length value shall be compliant with the values indicated in table 7.2.

### Table 7.2

<table>
<thead>
<tr>
<th>LPDU Length</th>
<th>LPDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00'</td>
<td>RFU</td>
</tr>
<tr>
<td>'01' to '1D'</td>
<td>With MTU = 32</td>
</tr>
<tr>
<td>'01' to '3D'</td>
<td>With MTU = 64</td>
</tr>
<tr>
<td>'01' to '7D'</td>
<td>With MTU = 128</td>
</tr>
<tr>
<td>'01' to 'FD'</td>
<td>With MTU = 256</td>
</tr>
<tr>
<td>'FE'</td>
<td>RFU</td>
</tr>
<tr>
<td>'FF'</td>
<td>Forbidden (The link layer frame is not present and only NSD bytes are present)</td>
</tr>
</tbody>
</table>

#### 7.3.2 Frames generation and transfer rules

The SPI master initiates an SPI access either to send a frame, retrieve a frame from the slave after a MAC access request or both.

If the SPI master has a frame to send, the SPI master shall send that frame in a single SPI access, however the SPI master may initiate a SPI access with a length higher than the length of the frame to send.

In case the SPI access is longer than the length of the frame being sent, SPI master and/or slave shall add Non-Significant Data (NSD) bytes following the CRC until the end of the SPI access.

A slave frame shall be retrieved in at most two SPI accesses if the number of bytes of the first SPI access is shorter than the slave frame. If the SPI master did not receive the entire slave frame in one SPI access, the master shall initiate a second SPI access with a length equal or greater than the number of remaining bytes of the frame to be retrieved from the slave.
In the second SPI access, the slave shall continue to send the same frame from the point where the previous SPI access stopped. The remaining part of a slave frame retrieved in a second access shall start on the first byte of the second access with the byte following the last byte retrieved in the prior access.

Master shall send only NSD bytes (i.e. bytes set to the value ‘FF’) during the second SPI access for retrieving the remaining bytes of a slave frame.

To retrieve a slave frame the SPI master may proceed with the following steps:

- When master does not have a frame to send and slave initiates a MAC access request asking for data transfer:
  - Step 1: master may generate an SPI access of minimum 1 byte to retrieve the slave frame length information.
  - Step 2: if the slave frame has not been entirely received in the first SPI access, the SPI master shall generate a second SPI access for retrieving the remaining bytes of the slave frame based on the information on the slave frame length received in the slave frame header during the first access.

- When master has a frame to send:
  - Step 1: master generates an SPI access with the length equal or higher than its frame length. At the same time master may receive on SPI_MISO part of or a full slave frame. When SPI access length is greater than the slave frame length, the slave sends NSD bytes at the end of its link layer frame, following the CRC bytes.
  - Step 2: if SPI access length is less than slave frame length, master shall generate a second SPI access for retrieving the remaining bytes of the slave frame based on the information on the slave frame length received in the slave frame header during the first SPI access.

In both cases above, master may generate a first SPI access with the length based on estimated optimal length for slave frame retrieval or may generate an SPI access of maximum length i.e. MTU in order to transfer the full slave frame during one SPI access.

The length byte of any frame shall always be the first byte sent in an SPI access, i.e. a new frame shall not be started in the same SPI access.

### 7.3.3 Data transfer cases

Some of the most representative data transfer cases based on the frames generation and transfer rules in clause 7.3.2 are described below. Any frame sent by master or slave shall be preceded by a MAC phase issued respectively by the master or slave. When two accesses are required for transferring a slave frame, master shall generate the second access at any time greater than or equal to the tCS value in clause 6.4.2.

**Case 1:** master initiates the MAC phase and then sends a frame. SPI access length is determined by the master frame length. No data is received from the slave.

<table>
<thead>
<tr>
<th>SPI_MOSI</th>
<th>SPI_MISO</th>
<th>LEN</th>
<th>LPDU</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO MAC</td>
<td></td>
<td></td>
<td>NSD</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{SPI access Length } \leq \text{ MTU}\]
Case 2: slave initiates a slave MAC access request to transfer a frame. Master performs a first access to retrieve the slave frame length followed by a second access to retrieve the remaining bytes of the slave frame considering the length information from the first access. 

![Diagram](Image)

\[\text{Total bytes transferred on MISO} \leq \text{MTU}\]

Case 3: slave initiates a MAC access request for sending a frame. Master generates an access with length equal to MTU to make sure the slave frame is transferred in a single access. Slave frame is shorter than the access length and slave appends NSD bytes after the end of its frame until the end of the access. 

![Diagram](Image)

\[\text{SPI access length} = \text{MTU}\]

Case 4: slave initiates a slave MAC access request for sending a frame. Consequently, master generates an access with length based on a best estimate. Master retrieves only part of the frame during the first access and will generate a second access to retrieve the remaining bytes of the slave frame. The length of the second access is based on the frame length information retrieved in the prior access. The total number of bytes transferred on MISO over both accesses is less than or equal to the MTU. 

![Diagram](Image)

\[\text{SPI access length} = \text{MTU}\]

Case 5: both master and slave have frames to transfer and MAC phase is initiated by both simultaneously. Master generates an access with length determined by its frame length. Master finds out that only a part of a slave frame was received and generates a second access to retrieve the remaining bytes of the slave frame. 

![Diagram](Image)

\[\text{SPI access length} = \text{MTU}\]

Case 6: both master and slave have frames to transfer and both initiate the MAC phase simultaneously. Master generates an access with length determined by its frame length. As the slave frame is shorter than the master frame, slave adds NSD bytes after the end of its frame up to access end. 

![Diagram](Image)

\[\text{SPI access length} = \text{MTU}\]

Case 7: both master and slave have frames to transfer and both initiate the MAC phase simultaneously. Master generates an access with the length equal to MTU to receive any slave frame occurring at the same time within a single access. Both master and slave may append NSD bytes after the end of their frames, up to access completion. 

![Diagram](Image)

\[\text{SPI access length} = \text{MTU}\]

The green boxes in the figures above indicate the initiator(s) of the MAC phase.
7.4 LLC layers

Three Logical Link Control (LLC) layers are defined in the present document:

- SHDLC: this is the generic LLC. SHDLC is defined in ETSI TS 102 613 [2], clause 10. Support of this LLC is mandatory for the master and the slave.

- CLT: this LLC is used for some proprietary protocol handling. CLT mode is defined in ETSI TS 102 613 [2], clause 11. Support of this LLC is optional for the master and the slave.

- MCT: this LLC consist of frames used during interface activation. Support of this LLC is mandatory for the master and the slave.

The control field is the first byte of the LPDU. Definition for the different LLC layers can be found in table 7.3.

<table>
<thead>
<tr>
<th>Frame types</th>
<th>Bit field</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFU</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>MCT</td>
<td>0 0 1</td>
</tr>
<tr>
<td>ACT (not used)</td>
<td>0 1 1</td>
</tr>
<tr>
<td>CLT</td>
<td>0 1 0</td>
</tr>
<tr>
<td>SHDLC</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7.3: LLC control field coding

The LPDUs shall be structured according to figures 7.9, 7.10 or 7.11, depending on the frame type.
7.5 Interworking of the LLC layers

After MAC activation, the SHDLC link shall not be established and no CLT session shall be open. Only the MCT LLC shall be used by the master and by the slave for the SPI interface initialization.

The master shall take the following action after a successful MCT LLC phase:

- If the master has data to be sent to the slave (e.g. due to a contactless transaction) that requires the use of the CLT LLC, it shall initiate a CLT LLC session.
- Otherwise it shall start the establishment of an SHDLC link as soon as possible.

After the slave and the master have established the SHDLC link or opened the CLT session, the slave and the master shall not send MCT LLC frames; received MCT LLC frames shall be ignored.

To enter the SHDLC LLC for the first time after MCT LLC, the link establishment procedure as described in clause 7.7.1 shall apply.

Once the SHDLC link is established, a CLT session shall not invalidate the SHDLC context and the endpoint capabilities negotiated during the SHDLC link establishment.

To enter the CLT LLC from MCT LLC or SHDLC LLC, the CLT session shall be opened as described in clause 11.6 of ETSI TS 102 613 [2]. The master shall open a CLT session only when all SHDLC I-Frames are acknowledged. SHDLC LLC frames received by the slave or by the master during a CLT session close the CLT session.

In case the slave or the master receives a corrupted frame, then the receiving entity shall use the error recovery procedure defined for the LLC of the last correctly received frame. Immediately after MAC activation, the error handling of the MCT LLC shall apply.

LPDU may be the SCL packet as defined in ETSI TS 103 666-1 [1], clause 8.3.2.

7.6 MCT LLC definition

7.6.1 MCT LPDU structure

The MCT LPDU shall be structured according to figure 7.12.

---

Figure 7.11: LPDU structure of the LLC layer of type SHDLC

---
The meaning of MCT_CTRL and MCT_DATA is given in table 7.4.

**Table 7.4: Meaning of MCT_CTRL and MCT_DATA**

<table>
<thead>
<tr>
<th>MCT_CTRL</th>
<th>Meaning</th>
<th>MCT_DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>MCT_READY</td>
<td>See table 7.5</td>
</tr>
<tr>
<td></td>
<td>Sent from Slave to Master</td>
<td></td>
</tr>
<tr>
<td>00010</td>
<td>MCT_MASTER_REQ</td>
<td>See table 7.7</td>
</tr>
<tr>
<td></td>
<td>Sent from Master to Slave</td>
<td></td>
</tr>
<tr>
<td>All other values (see note)</td>
<td>RFU</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** All other values are reserved for future use. These values shall not be set by the transmitting entity and shall be ignored by the receiving entity.

The MCT LPDU length shall be lower than or equal to 29 bytes.

**NOTE:** 29 bytes is equal to the smallest MTU size (32 bytes) minus the overhead of the link layer frame as described in clause 7.3.1.

### 7.6.2 MCT_DATA from master

Tables 7.5, 7.6 and 7.7 define the capabilities of the master.

**Table 7.5: Master-specific MCT_DATA field**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Info/parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Spec_Ver</td>
<td>Specification version to which master is compliant. Defined in table 7.6.</td>
</tr>
<tr>
<td>1</td>
<td>Capabilities</td>
<td>Defined in table 7.7.</td>
</tr>
<tr>
<td>2,3</td>
<td>T4</td>
<td>Inactivity period for power saving mode (ms) according to clause 7.8.2.1.</td>
</tr>
</tbody>
</table>

**NOTE:** All additional bytes (4 to 28) are reserved for future extensions of the protocol. They should not be sent by the master and shall be ignored by the slave.

**Table 7.6: Specification version**

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 to 4</td>
<td>00001</td>
<td>Major version of the SPI interface</td>
</tr>
<tr>
<td>3 to 1</td>
<td>000</td>
<td>Minor version of the SPI interface</td>
</tr>
</tbody>
</table>
Table 7.7: Master capabilities indication in MCT_DATA field

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 to 6</td>
<td>000</td>
<td>RFU (see note)</td>
</tr>
<tr>
<td>5, 4</td>
<td>11</td>
<td>Full Power Mode 3 supported</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Full Power Mode 2 supported</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>Full Power Mode 1 e.g. 10 mA supported</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>Low Power Mode e.g. 5 mA supported</td>
</tr>
<tr>
<td>3, 2</td>
<td>11</td>
<td>MTU 256 bytes</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>MTU 128 bytes</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>MTU 64 bytes</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>MTU 32 bytes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RFU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Flow control SHDLC-based (default)</td>
</tr>
</tbody>
</table>

NOTE: These bits shall not be set by the master and shall be ignored by the slave.

After the SPI activation as defined in clause 7.2.3.4 or in clause 7.2.4.6, the master shall send the MCT_MASTER_REQ frame and the slave shall respond with the MCT_READY frame.

The MCT phase shall be performed with default SPI_CLK = 1MHz and T1 ≥ 255 μs.

The MTU negotiation between the master and slave shall be between the MTU sent by the master in the MCT_MASTER_REQ frame and the MTU sent by the slave in the MCT_READY. The lower of the MTU values will be used by both master and slave for all frames.

Master shall indicate in MCT_MASTER_REQ its highest power source capability (i.e. Low Power, Full Power Mode 1, Full Power Mode 2 or Full Power Mode 3) available for the respective slave. The power levels are defined as increasing respectively for Low Power Mode, Full Power Mode 1, Full Power Mode 2, Full Power Mode 3.

A slave able to limit its maximum current according to the received master power capabilities shall:

- support Low Power Mode;
- support Full Power Mode 1 if the slave requires a maximum current corresponding to Full Power Mode 1 or a higher current;
- start in Low Power Mode following VDD ON and may switch to a Full Power Mode, depending on the power mode capabilities received from the master in MCT_MASTER_REQ.

Low Power, Full Power Mode 1, Full Power Mode 2 and Full Power Mode 3 current values are out of scope of the present document. These power modes are described for general reference for the master and the slave when applicable and are not mandatory. The master and the slave may agree at the system design time on the implementation specific power requirements, on the power modes to be supported and specific current levels may be defined for the power modes. In a particular case the master includes in its power budget the maximum current required by the slave and in this case the slave may ignore the power modes information sent by master in MCT_DATA.

7.6.3 MCT_DATA from slave

Tables 7.8 and 7.9 define the capabilities of the slave.
Table 7.8: Slave-specific MCT_DATA field (bytes 0…7)

<table>
<thead>
<tr>
<th>Byte</th>
<th>Info/parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Spec_Ver</td>
<td>Specification version to which slave is compliant. Defined in table 7.6.</td>
</tr>
<tr>
<td>1</td>
<td>Capabilities</td>
<td>Defined in table 7.9.</td>
</tr>
<tr>
<td>2</td>
<td>SPI_CLK</td>
<td>Max SPI_CLK value supported by SPI slave (MHz).</td>
</tr>
<tr>
<td>3</td>
<td>T1</td>
<td>Slave MAC Ready Time (μs).</td>
</tr>
<tr>
<td>4</td>
<td>T3</td>
<td>Slave resume time from power saving mode (us) (see note 1).</td>
</tr>
<tr>
<td>5,6</td>
<td>T4</td>
<td>Slave supported Inactivity period for entering power saving mode T4 (ms) according to clause 7.8.2.1.</td>
</tr>
<tr>
<td>7</td>
<td>POT</td>
<td>Power-ON Time: time after slave VDD valid when master can send MCT_MASTER_REQ (ms) (see note 2).</td>
</tr>
</tbody>
</table>

NOTE 1: In case a slave may “self-resume” (e.g. due to activities on another interface) and it has T3 < T1 slave shall report T1 value for T3.

NOTE 2: This value should be used by master at next power on. The initial value used by the master should be 1 s.

NOTE 3: All additional bytes (8 to 28) are reserved for future extensions of the protocol. They should not be sent by the slave and shall be ignored by the master.

Table 7.9: Slave capabilities indication in MCT_DATA field byte 0

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 to 5</td>
<td>0000</td>
<td>RFU (see note 1)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Slave-driven flow control with SPI module enabled (see note 2)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>No slave-driven flow control master restrictions</td>
</tr>
<tr>
<td>3, 2</td>
<td>11</td>
<td>MTU 256 bytes</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>MTU 128 bytes</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>MTU 64 bytes</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>MTU 32 bytes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RFU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Flow control SHDLC-based (default)</td>
</tr>
</tbody>
</table>

NOTE 1: These bits shall not be set by the slave and shall be ignored by the master.

NOTE 2: The master shall postpone any transfers to other slaves on the shared bus as long as the slave is performing slave-driven flow control.

7.6.4 MCT activation procedure

Slave start-up time following power-on is defined as POT and has an initial value of 1 s for the first power on, when the slave reported MCT_READY parameters are not yet available. After POT time (from the time VDD is valid after power-on) slave shall be ready to receive the MCT_MASTER_REQ from master. Shorter POT values may be reported by slave in MCT_READY. Master shall use the POT value reported by slave or a higher value in subsequent power-up sequences.

Master shall wait for MCT_READY from slave after sending MCT_MASTER_REQ. In case slave did not send MCT_READY response (no MAC access request) within MCT_SLAVE_TIMEOUT or if MCT_READY is corrupted, master shall retry the MCT activation by sending another MCT_MASTER_REQ frame. Master shall retry at least two times i.e. shall re-send MCT_MASTER_REQ at least twice without power toggle. Specific recovery procedure further steps after these retries are implementation specific and out of scope of the present document.

After power-up, if slave gets a corrupted frame or any other frame instead of the MCT_MASTER_REQ, slave shall discard the data and remain in receive state. If slave receives three corrupted or invalid frames instead of MCT_MASTER_REQ, slave should enter power saving mode. If the MCT activation has not been successfully performed or master did not initiate an SPI access within MCT_MASTER_TIMEOUT following power-on or for the subsequent retries in case of errors, slave should enter power saving mode.

MCT_MASTER_TIMEOUT is the maximum time within which the master shall send the MCT_MASTER_REQ after power-on or for the retries in case of errors. The value defined is 1 s.

MCT_SLAVE_TIMEOUT is the maximum time within which the slave shall send the MCT_READY response to MCT_MASTER_REQ. The default value defined is 200 ms. A slave may send MCT_READY faster according to certain applications requirements.
7.7  SHDLC LLC definition

7.7.1  SHDLC overview

The provisions of ETSI TS 102 613 [2], clause 10.1 shall apply. The SWP SHDLC layer is replaced by the SPI SHDLC layer defined in the present document.

The SHDLC layer shall ensure that data passed up to the next layer has been received exactly as transmitted i.e. error free, without loss and in the correct order. Also, the SHDLC layer manages the flow control, which ensures that data is transmitted only as fast as the receiver may receive it.

The provisions of ETSI TS 102 613 [2] clauses from 10.3 to 10.8 shall apply. Additional SHDLC rules are defined below.

7.7.2  Endpoints

SHDLC communication occurs between two endpoints. Those endpoints may be either the master endpoint or the slave endpoint. There is no priority of traffic.

![Figure 7.13: Endpoints](image)

In ETSI TS 102 613 [2], clause 10, the term CLF refers to the master endpoint and the term UICC to the slave endpoint.

7.7.3  Flow control

7.7.3.1  Overview

Flow control is performed by a transmitter in order to avoid corruption or loss of data. It consists of methods applied by the transmitter and receiver in order to send a maximum number of SHDLC frames that can be accepted by the receiver, after which it shall stop sending data until the receiver sends at least an acknowledgement (e.g. SHDLC I-frame or SHDLC S-frame) for one of the received SHDLC frames.

NOTE: The number of free window size slots is not to be decremented for the second access for retrieving the remaining bytes of a slave frame.

The maximum number of SHDLC frames that can be sent by a transmitter is determined by the negotiated window size.

7.8  Power management

7.8.1  Power saving mode

In order to optimize the power consumption, both the master and the slave may enter into power saving mode. This clause defines the conditions for the slave to enter into power saving mode and the procedures for the master for resuming the slave from power saving mode.
The master and the slave shall both resume in the same LLC context following the master or the slave resumption. The master and the slave may then initiate the switch to a different LLC context according to clause 7.5.

7.8.2 Conditions for entering power saving mode

7.8.2.1 Slave entering power saving mode

The slave shall not enter into power saving mode, if the slave has issued a MAC access request and is waiting for data transfer from the master.

The slave may enter into power saving mode in one of the cases below, assuming there is no pending activity:

1) All frames have been transmitted by the slave, acknowledged successfully by the master and the slave detects an inactivity time $T_4$ with no assertion of SPI_NSS by the master. Entering into power saving mode based on the inactivity period may be disabled by the master thru the inactivity period value negotiated at interface initialization as described below.

2) The slave informs that it does not require or expect any further activities e.g. through EVT_LINK_END_OF_OPERATION sent from its Link Application Gate to the Link Service Gate of the Network Controller Host, as defined in ETSI TS 103 666-1 [1]. The slave may enter into power saving mode after it receives the SHDLC link layer acknowledgement for EVT_LINK_END_OF_OPERATION.

3) Following the power-up or during MCT activation procedure as described in clause 7.6.4, when the slave does not detect any activity for more than the default inactivity period MCT_MASTER_TIMEOUT.

The inactivity period $T_4$ is negotiated by the master and the slave at interface initialization as described in clause 7.6. The master shall provide an inactivity period in MCT_MASTER_REQ and the slave shall send back an MCT_READY with the same $T_4$ value for acceptance, or a different value if it cannot support the value received from the master. If the master sends $T_4='FFFF'$ in MCT_MASTER_REQ, the slave shall disable the entering power saving mode on detection of an inactivity period and the slave shall send an MCT_READY with the same $T_4$ value. If the slave sends $T_4='FFFF'$ in MCT_READY it indicates to the master that the slave will not enter into power saving mode based on the detection of inactivity time, regardless the value sent by the master in the MCT_MASTER_REQ and the master should not resume the slave on this condition.

A slave which supports a resume time $T_3$ lower than $T_1$ and is either not able to systematically enter into power saving mode (i.e. it may remain active) or may resume independently of the activity on the SPI interface, shall report in MCT_READY a $T_3$ value equal to $T_1$ (or higher).

In power saving mode, the slave shall maintain its SPI interface as for the case when SPI_NSS is de-asserted with the slave not in power saving mode.

7.8.2.2 Master entering power saving mode

The master may enter into an implementation-specific power saving mode at any time, without informing the slave. The slave power source status and capabilities indicated by the master at interface initialization shall not change when the master enters into power saving mode, is in power saving mode, is resuming from power saving mode or after the master has resumed. SPI_NSS shall be maintained de-asserted by the master when the master is in power saving mode and when the master is resuming.

7.8.3 Resuming from power saving mode

7.8.3.1 Resuming the slave from power saving mode

Resuming the slave from power saving mode shall be performed by the master when any of the conditions above for the slave to enter into power saving mode has been previously met. The master shall ensure that all signals it drives are in the idle state corresponding to SPI mode 0 before initiating the resuming of the slave. The leading edge of the SPI_NSS assertion shall trigger the slave to resume.
The master shall perform the following procedure to resume the slave:

1) In the case of a 4 signals SPI interface, the master checks the state of the SPI_NSS signal and if it is de-asserted, it goes to the step 2, otherwise loops on step 1. In the case of a 5 signals SPI interface, the master starts with step 2.

2) The master asserts SPI_NSS and waits for at least T3, then goes to step 3.

3) The master considers the slave as resumed from the power saving mode and starts the data transfer. The slave shall support the resumption up to the data transfer phase with SPI_NSS continuously asserted by the master during T3.

7.8.3.2 Resuming the master from power saving mode

If the master has entered into power saving mode, it shall resume when the slave initiates a MAC access request. The slave initiates the MAC access request with the procedures as described in clause 7, regardless the power management status of the master. Following a resume by a MAC access request during T2 as described in clause 7.2.2.3, the master shall start an SPI access after T1 or later, from the leading edge of the slave MAC access request pulse, i.e. clauses 7.2.3.2 and 7.2.4.3 apply.

The leading edge of the MAC access request (SPI_NSS or SPI_INT assertion) should trigger the resuming of the master.
Annex A (informative):
Slave SPI interface states electrical description

A.1 Slave SPI interface for 5 wire interface

A.1.1 Slave SPI 5 wire interface diagram

![Figure A.1: SPI electrical interface with 5 signals](image)

SPI_INT is generated separately from the SPI module which does not include the SPI_INT by default. SPI_INT is an output of the slave and an input to the master.

A.1.2 Slave SPI 5 wire interface states electrical description

The signals SPI_MISO, SPI_MOSI and SPI_CLK may be shared between multiple slaves. In that case each slave has a dedicated SPI_NSS. Shared SPI_INT is out of the scope of the present document.

The slave allows SPI shared bus and states according to the table A.1.

<table>
<thead>
<tr>
<th>slave state</th>
<th>SPI_INT slave</th>
<th>SPI_NS slave</th>
<th>SPI_CLK slave</th>
<th>SPI_MISO slave</th>
<th>SPI_MOSI slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial state (see note 1)</td>
<td>HiZ or DA</td>
<td>IL</td>
<td>IL or HiZ</td>
<td>II or HiZ</td>
<td>Idle or active (see note 2)</td>
</tr>
<tr>
<td>configured/de-selected</td>
<td>HiZ or DA</td>
<td>IL</td>
<td>IL</td>
<td>DA</td>
<td>II or IL (see note 2)</td>
</tr>
<tr>
<td>configured/selected</td>
<td>HiZ or DA</td>
<td>IL or IL</td>
<td>IL</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>pro-active (see note 3)</td>
<td>A</td>
<td>IL</td>
<td>DA</td>
<td>II</td>
<td>Idle or active (see note 2)</td>
</tr>
</tbody>
</table>

Table A.1: 5 wire SPI interface states
Table A.2: 4 wire SPI interface states

<table>
<thead>
<tr>
<th>slave state</th>
<th>SPI_NSS</th>
<th>SPI_CLK</th>
<th>SPI_MISO</th>
<th>SPI_MOSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>slave</td>
<td>master</td>
<td>slave</td>
<td>master</td>
<td>slave</td>
</tr>
<tr>
<td>initial state (see note 1)</td>
<td>II or HiZ</td>
<td>DA</td>
<td>II or HiZ</td>
<td>DA</td>
</tr>
<tr>
<td>configured/de-selected</td>
<td>DA, IL (see note 3)</td>
<td>DA, IL (see note 3)</td>
<td>II</td>
<td>Idle or active (see note 2)</td>
</tr>
</tbody>
</table>

NOTE 1: State after VDD being valid until the slave internal configuration is completed.
NOTE 2: Master may be on an ongoing communication with another slave.
NOTE 3: Applies only during T2.

A.1.3 Slave SPI 4 wire interface diagram

Slave asserts the SPI_NSS signal independently from the SPI module in which the slave select signal is an input.

A.1.4 Slave SPI 4 wire interface states electrical description

The signals SPI_MISO, SPI_MOSI and SPI_CLK may be shared between multiple slaves. In that case each slave have a dedicated SPI_NSS.

Slave allows SPI shared bus and states according to the table A.2.
<table>
<thead>
<tr>
<th>slave state</th>
<th>SPI_NSS</th>
<th>SPI_CLK</th>
<th>SPI_MISO</th>
<th>SPI_MOSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>configured/selected</td>
<td>DA, A or IL (see notes 3 and 8)</td>
<td>A</td>
<td>IL</td>
<td>D</td>
</tr>
<tr>
<td>busy (see note 4)</td>
<td>A</td>
<td>DA, IL (see note 3)</td>
<td>IL or II (see note 7)</td>
<td>Idle or active (see notes 2 and 7)</td>
</tr>
<tr>
<td>pro-active (see note 5)</td>
<td>A</td>
<td>IL or A (see note 3 or 6)</td>
<td>II</td>
<td>Idle or active (see note 2)</td>
</tr>
<tr>
<td>power saving mode</td>
<td>DA, IL (see note 3)</td>
<td>DA, IL (see note 3)</td>
<td>II</td>
<td>Idle or active (see note 2)</td>
</tr>
</tbody>
</table>

NOTE 1: State after VDD being valid until the slave internal configuration is completed.
NOTE 2: Master may be on an ongoing communication with another slave.
NOTE 3: The state is not enforced (SS_MO and SS_SO not asserted), the line is sampled (SS_MI and SS_SI).
NOTE 4: State SPI_NSS is asserted only by slave after data transfer according to clause 7.2.4.5.
NOTE 5: Applies only during the time T2.
NOTE 6: Applies also during simultaneous assertion of SPI_NSS by master and slave for MAC phase.
NOTE 7: According to Busy state description in clause 6.5.1 and slave capabilities information in table 7.9 bit 4.
NOTE 8: Slave may assert its SPI_NSS at the same time with master SPI_NSS to enter into the busy state.
Annex B (informative):
Change history

The table below indicates all changes that have been incorporated into the present document since it was placed under change control.

<table>
<thead>
<tr>
<th>Date</th>
<th>Meeting</th>
<th>Plenary Doc</th>
<th>CR</th>
<th>Rev</th>
<th>Cat</th>
<th>Document</th>
<th>Subject/Comment</th>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/10/2019</td>
<td>SCP#89</td>
<td>SCP(19)000211</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Version 15.0.0 first publication</td>
<td></td>
<td>15.0.0</td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000247</td>
<td>001</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 MCT LLC RFU additions</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000248</td>
<td>002</td>
<td>-</td>
<td>D</td>
<td>CR 103 713 Rel-15 Clarification of the naming of the signal 'master internal interrupt'</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000249</td>
<td>003</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 Data Transfer definition</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000250</td>
<td>004</td>
<td>-</td>
<td>C</td>
<td>CR 103 713 Rel-15 AC characteristics clarifications</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000251</td>
<td>005</td>
<td>-</td>
<td>C</td>
<td>CR 103 713 Rel-15 DC characteristics clarifications</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>07/01/2020</td>
<td>SCP#90</td>
<td>SCP(19)000252</td>
<td>006</td>
<td>-</td>
<td>B</td>
<td>CR 103 713 Rel-15 Power Management</td>
<td>15.0.0</td>
<td>15.1.0</td>
<td></td>
</tr>
<tr>
<td>01/04/2020</td>
<td>SCP #91</td>
<td>SCP(20)091023</td>
<td>007</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 SPI Resume Time</td>
<td>15.1.0</td>
<td>15.2.0</td>
<td></td>
</tr>
<tr>
<td>01/04/2020</td>
<td>SCP #91</td>
<td>SCP(20)091024</td>
<td>008</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 SPI_INT description</td>
<td>15.1.0</td>
<td>15.2.0</td>
<td></td>
</tr>
<tr>
<td>01/04/2020</td>
<td>SCP #91</td>
<td>SCP(20)091025r1</td>
<td>009r1</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 MAC SPI interface states definition</td>
<td>15.1.0</td>
<td>15.2.0</td>
<td></td>
</tr>
<tr>
<td>01/04/2020</td>
<td>SCP #91</td>
<td>SCP(20)091026</td>
<td>010</td>
<td>-</td>
<td>D</td>
<td>CR 103 713 Rel-15 Editorial Changes</td>
<td>15.1.0</td>
<td>15.2.0</td>
<td></td>
</tr>
<tr>
<td>01/04/2020</td>
<td>SCP #91</td>
<td>SCP(20)091027</td>
<td>011</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 SPI slave flow control clarification</td>
<td>15.1.0</td>
<td>15.2.0</td>
<td></td>
</tr>
<tr>
<td>28/07/2020</td>
<td>SCP#94</td>
<td>SCP(20)000046r1</td>
<td>12r1</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 Slave interface states</td>
<td>15.2.0</td>
<td>15.3.0</td>
<td></td>
</tr>
<tr>
<td>28/07/2020</td>
<td>SCP#94</td>
<td>SCP(20)000047r1</td>
<td>13r1</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 MAC activation</td>
<td>15.2.0</td>
<td>15.3.0</td>
<td></td>
</tr>
<tr>
<td>23/09/2020</td>
<td>SCP#95</td>
<td>SCP(20)000106</td>
<td>14</td>
<td>-</td>
<td>F</td>
<td>CR 103 713 Rel-15 Power Modes updates</td>
<td>15.2.0</td>
<td>15.3.0</td>
<td></td>
</tr>
<tr>
<td>16/12/2020</td>
<td>SCP#97</td>
<td>SCP(20)000161</td>
<td>015</td>
<td>-</td>
<td>F</td>
<td>CR 103_713_Rel-15_AC_DC_Characteristics_Updates</td>
<td>15.3.0</td>
<td>15.4.0</td>
<td></td>
</tr>
</tbody>
</table>
## History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>V15.0.0</td>
<td>November 2019</td>
<td>Publication</td>
</tr>
<tr>
<td>V15.1.0</td>
<td>February 2020</td>
<td>Publication</td>
</tr>
<tr>
<td>V15.2.0</td>
<td>May 2020</td>
<td>Publication</td>
</tr>
<tr>
<td>V15.3.0</td>
<td>October 2020</td>
<td>Publication</td>
</tr>
<tr>
<td>V15.4.0</td>
<td>February 2021</td>
<td>Publication</td>
</tr>
</tbody>
</table>