



**HAL**

**SPI Slave as DPM Interface**

**netX 100/500**

**V1.0.x.x**

**Hilscher Gesellschaft für Systemautomation mbH**

**[www.hilscher.com](http://www.hilscher.com)**

DOC120105HAL02EN | Revision 2 | English | 2012-01 | Released | Public

# Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>3</b>
1.1	About this Document.....	3
1.2	List of Revisions .....	3
1.3	Terms, Abbreviations and Definitions .....	3
1.4	References .....	3
1.5	Functional Overview.....	4
1.5.1	System Requirements .....	4
1.5.2	Intended Audience .....	4
1.5.3	Technical Data .....	4
1.5.4	Limitations .....	4
1.6	Legal Notes .....	5
1.6.1	Copyright.....	5
1.6.2	Important Notes.....	5
1.6.3	Exclusion of Liability.....	6
1.6.4	Export.....	6
<b>2</b>	<b>The Interface .....</b>	<b>7</b>
2.1	Overview of Service Classes .....	7
2.2	Control Service Class.....	8
2.2.1	SPISDPM_Deinit() - Deinitialize the SPI Slave.....	8
2.2.2	SPISDPM_GetParameter() - Get Parameter.....	9
2.2.3	SPISDPM_Init() - Init SPI Slave.....	10
2.2.4	SPISDPM_SetParameter() - Set Parameter.....	11
2.2.5	SPISDPM_Start() - Start SPI Slave.....	12
2.3	Status Service Class.....	13
2.3.1	SPISDPM_GetCounters() - Get Status Counters.....	13
2.4	Structure Definitions.....	14
2.4.1	SPISDPM_CMD_T - SPI Slave DPM Command .....	14
2.4.2	SPISDPM_STATUS_COUNTERS_T - Status Counters .....	15
2.5	Enumeration Definitions .....	16
2.5.1	SPISDPM_PARAMETER - Parameters ID that can be set/get by SPISDPM_SetParameter()/SPISDPM_GetParameter() .....	16
2.5.2	SPISDPM_RESULT_E - Result Codes for HAL Functions.....	17
<b>3</b>	<b>Appendix .....</b>	<b>18</b>
3.1	SPI Timing.....	18
3.2	Access Request Frame Format .....	19
3.3	Example for NXDB500-SYS Evaluation Board .....	20
3.4	List of Tables .....	21
3.5	List of Figures.....	22
3.6	Contacts .....	23

# 1 Introduction

## 1.1 About this Document

This manual describes the interface of the SPI Slave as DPM Interface with the aim to support and lead you during the integration process of the given unit into your application running under your own operating system.

It is a description of how to configure and to exchange data with it in general.

## 1.2 List of Revisions

Rev	Date	Name	Chapter	Revision
1	2012-01-02	AO		Created

Table 1: List of Revisions

## 1.3 Terms, Abbreviations and Definitions

Term	Description
DPM	Dual-Ported Memory
SPI	Serial Peripheral Interface
CPHA	SPI clock phase
CPOL	SPI clock polarity
CS	chip select
HAL	Hardware Abstraction Layer
MISO	master in slave out (data)
MOSI	master out slave in (data)

Table 2: Terms, Abbreviations and Definitions

All variables, parameters, and data used in this manual have the LSB/MSB ("Intel") data format. This corresponds to the convention of the Microsoft C Compiler.

All IP addresses in this document have host byte order.

## 1.4 References

This document based on the following specification:

Number	Document
1	Design Note #25, <a href="http://www.AVRfreaks.net">www.AVRfreaks.net</a>

Table 3: References

## 1.5 Functional Overview

You as a user are getting a capable and a general-purpose Software interface package with following features:

- Configuration of the SPI Slave as DPM interface
- Getting of Status Information of the SPI Slave as DPM interface

### 1.5.1 System Requirements

The software package has the following system requirements to its environment:

- netX 100/500 Chip as CPU hardware platform
- operating system independency

### 1.5.2 Intended Audience

This manual is suitable for software developers with the following background:

- Knowledge of the programming language C

### 1.5.3 Technical Data

- Up to 16 MHz SPI Clock rate
- SPI Modes 1 and 3 are supported (CPHA = 1, CPOL = 0/1)
- MSB first stream format is supported
- Access of netX memory dword-wise, byte and word accesses must be done via read-modify-write
- 3 mapping windows for translation of 16-bit SPI request address to 32-bit internal netX address

### 1.5.4 Limitations

- SPI Modes 0 and 2 (CPHA = 0) are not supported
- LSB first stream format is not supported
- No byte or word access to netX memory

## 1.6 Legal Notes

### 1.6.1 Copyright

© 2008-2010 Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying material (user manual, accompanying texts, documentation, etc.) are protected by German and international copyright law as well as international trade and protection provisions. You are not authorized to duplicate these in whole or in part using technical or mechanical methods (printing, photocopying or other methods), to manipulate or transfer using electronic systems without prior written consent. You are not permitted to make changes to copyright notices, markings, trademarks or ownership declarations. The included diagrams do not take the patent situation into account. The company names and product descriptions included in this document may be trademarks or brands of the respective owners and may be trademarked or patented. Any form of further use requires the explicit consent of the respective rights owner.

### 1.6.2 Important Notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

### 1.6.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

### 1.6.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different countries, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

## 2 The Interface

This section describes the data transfer services available to the SPI Slave User with their associated service primitives and parameters.

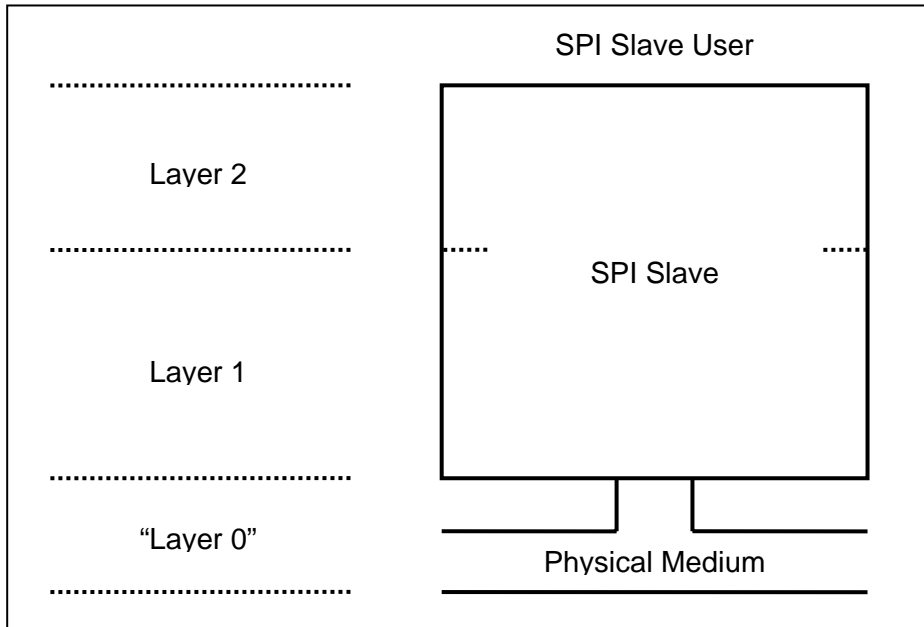


Figure 1: Interface between Interface User and Interface in Relation to Layer Model

### 2.1 Overview of Service Classes

The user of Layer 2 is provided with the following service classification:

#### Control

This service class defines the transfer of control commands from an SPI Slave User to the SPI Slave.

#### Status

This service class defines the transfer of status information from the SPI Slave to a SPI Slave MAC user.

## 2.2 Control Service Class

### 2.2.1 SPISDPM\_Deinit() - Deinitialize the SPI Slave

This function immediately stops the SPI Slave and inactivates the corresponding XC port.

#### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_Deinit( unsigned int  uPortNo )
```

#### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number

Table 4: SPISDPM\_Deinit() - Function Arguments

#### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified
SPISDPM_ERR_XC_INIT_FAILED	Initialization of XC-port failed

Table 5: SPISDPM\_Deinit() - Function Return Values



## 2.2.2 SPISDPM\_GetParameter( ) - Get Parameter

This function reads a SPI Slave parameter.

### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_GetParameter( unsigned int    uPortNo,  
                      unsigned long   ulPrmID,  
                      unsigned long*  pulPrmVal )
```

### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number
ulPrmID	[in]	Parameter ID
pulPrmVal	[in]	Pointer to parameter value

Table 6: SPISDPM\_GetParameter( ) - Function Arguments

### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified

Table 7: SPISDPM\_GetParameter( ) - Function Return Values

### 2.2.3 SPISDPM\_Init() - Init SPI Slave

This function loads the XC microcode into the XC unit and configures the SPI Mode.

#### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_Init( unsigned int  uPortNo,  
              int           fCPOL )
```

#### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number
fCPOL	[in]	SPI clock polarity CPOL (0 or 1)

Table 8: SPISDPM\_Init() - Function Arguments

#### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified
SPISDPM_ERR_XC_INIT_FAILED	Initialization of XC-port failed

Table 9: SPISDPM\_Init() - Function Return Values

## 2.2.4 SPISDPM\_SetParameter( ) - Set Parameter

This function writes a SPI Slave parameter.

### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_SetParameter( unsigned int    uPortNo,  
                      unsigned long   ulPrmID,  
                      unsigned long   ulPrmVal )
```

### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number
ulPrmID	[in]	Parameter ID
ulPrmVal	[in]	Parameter value

Table 10: SPISDPM\_SetParameter( ) - Function Arguments

### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified

Table 11: SPISDPM\_SetParameter( ) - Function Return Values

## 2.2.5 SPISDPM\_Start( ) - Start SPI Slave

This function starts the XC unit, which was initialized before.

### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_Start( unsigned int  uPortNo )
```

### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number

Table 12: SPISDPM\_Start( ) - Function Arguments

### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified
SPISDPM_ERR_XC_INIT_FAILED	Initialization of XC-port failed

Table 13: SPISDPM\_Start( ) - Function Return Values

## 2.3 Status Service Class

### 2.3.1 SPISDPM\_GetCounters( ) - Get Status Counters

This function gets the internal status counters from the SPI Slave.

#### Function Prototype

```
SPISDPM_RESULT_E  
SPISDPM_GetCounters( unsigned int      uPortNo,  
                     SPISDPM_STATUS_COUNTERS_T* ptCounters )
```

#### Function Arguments

Argument	Direction	Description
uPortNo	[in]	XC port number
ptCounters	[out]	Pointer to counter structure

Table 14: SPISDPM\_GetCounters( ) - Function Arguments

#### Function Return Values

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified

Table 15: SPISDPM\_GetCounters( ) - Function Return Values

## 2.4 Structure Definitions

### 2.4.1 SPISDPM\_CMD\_T - SPI Slave DPM Command

#### SPISDPM\_CMD\_T

Name	Type	Description
usAddr	unsigned short	DPM Address
fCmd	int	1/0: Read/Write command
bLen	unsigned char	Length in Bytes, must be between 4..124 and (Len mod 4==0)
bOutData[MAX_DATA_LEN]	unsigned char	Output Data bytes
bInData[MAX_DATA_LEN]	unsigned char	Input Data bytes
bCnf	unsigned char	Confirmation of previous request

Table 16: SPISDPM\_CMD\_T - Structure

## 2.4.2 SPISDPM\_STATUS\_COUNTERS\_T - Status Counters

Contains the Error counters of the SPI Slave DPM.

### SPISDPM\_STATUS\_COUNTERS\_T

Name	Type	Description
ulSPISDPM_WRITE_REQ_PROCESSED	unsigned long	Counter for number of processed write requests
ulSPISDPM_READ_REQ_PROCESSED	unsigned long	Counter for number of processed read requests
ulSPISDPM_CS_DEASSERT_ERRORS	unsigned long	Counter for chip-select deassert errors
ulSPISDPM_INTERNAL_ADDR_INVALID	unsigned long	Counter for number of occurrences that the translated internal address was invalid
ulSPISDPM_LENGTH_ERROR	unsigned long	Counter for rcvd length smaller 4 or greater 124 or not dword-aligned
ulSPISDPM_WR_REQ_RCVD_LEN_SM_HDR_LEN	unsigned long	Counter for write requests with received length smaller header length
ulSPISDPM_REQ_OVERRUN	unsigned long	Counter for number of occurrences that a new request (read or write) received before previous write request finished

Table 17: SPISDPM\_STATUS\_COUNTERS\_T - Structure

## 2.5 Enumeration Definitions

### 2.5.1 SPISDPM\_PARAMETER - Parameters ID that can be set/get by SPISDPM\_SetParameter()/SPISDPM\_GetParameter()

#### SPISDPM\_PARAMETER

Definition	Description
SPISDPM_PARAMETER_VALID_READ_DATA_COOKIE	8-bit value, MISO character content that signals valid read data follows
SPISDPM_PARAMETER_AREA1_DPM_BORDER	16-bit DPM offset address of Area 2 start, Note: Area 1 starts at DPM offset 0x0000
SPISDPM_PARAMETER_AREA2_DPM_BORDER	16-bit DPM offset address of Area 3 start
SPISDPM_PARAMETER_AREA3_DPM_BORDER	16-bit DPM offset address of Area 3 end plus 1
SPISDPM_PARAMETER_AREA1_INTERNAL_START_ADDR	32-bit Internal start address of Area 1 (netX address space)
SPISDPM_PARAMETER_AREA2_INTERNAL_START_ADDR	32-bit Internal start address of Area 2 (netX address space)
SPISDPM_PARAMETER_AREA3_INTERNAL_START_ADDR	32-bit Internal start address of Area 3 (netX address space)

Table 18: SPISDPM\_PARAMETER - Enumeration



## 2.5.2 SPISDPM\_RESULT\_E - Result Codes for HAL Functions

All functions return one of the following values after returning from the function call. Function return values shall always be evaluated by the calling function.

### SPISDPM\_RESULT\_E

Definition	Description
SPISDPM_OKAY	No error in function call
SPISDPM_ERR_INVALID_XC_PORT	Invalid XC-port number specified
SPISDPM_ERR_XC_INIT_FAILED	Initialization of XC-port failed
SPISDPM_ERR_INVALID_PARAM	Invalid parameter specified

Table 19: SPISDPM\_RESULT\_E - Enumeration

## 3 Appendix

### 3.1 SPI Timing

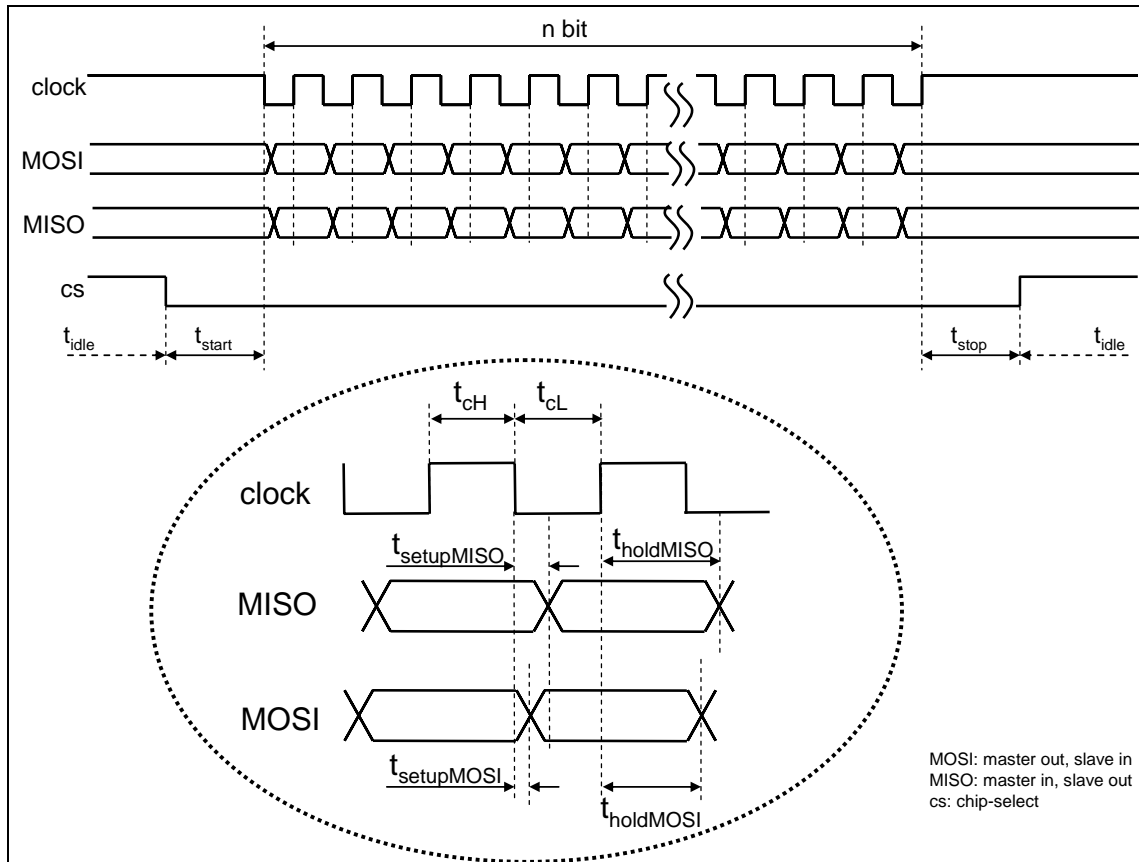


Figure 2: SPI timing

Parameter	Min.	Typ.	Max.	Unit
n		multiple of 8		bit
$t_{\text{start}}$	150			ns
$t_{\text{stop}}$		$t_{\text{cH}}$		ns
$t_{\text{idle}}$	150			ns
$t_{\text{cH}} + t_{\text{cL}}$	100			ns
$t_{\text{setupMISO}}$			0	ns
$t_{\text{holdMISO}}$	20	30	40	ns
$t_{\text{setupMOSI}}$			$t_{\text{cL}} - 10$	ns
$t_{\text{holdMOSI}}$	0			ns

Table 20: SPI timing parameters

## 3.2 Access Request Frame Format

### General frame header

The header is always generated by host device SPI Master interface via MOSI. The first 2 bytes contains address information (MSB first), followed by a control byte. The address must be a multiple of 4. Control byte MSB defines data transfer direction (nW/R). If this bit is set, the current transfer reads data from netX, if it is not set, data is written to netX. Proceeding 7 LSBs of the control byte contain transfer length information (length, bytes to be transferred). The length information must be a multiple between 4 and between 4 and 124. The SPI Slave interface maps the 16-bit address information into a 32-bit internal netX address using 3 configurable windows.

During the first address information byte the netX will send its ID character on MISO. The value of the ID for netX100/500 is 0x64 (MSB first). During the second address information byte the netX will send a confirmation byte about the previous request on MISO (MSB first). The byte has following meaning:

- Bit 0: the translated internal address was invalid
- Bit 1: chip-select deassert error
- Bit 2: the received length was smaller 4 or greater 124 or not a multiple of 4
- Bit 3: write request detected with received length smaller header length
- Bit 4: a new request (read or write) was received before previous write request finished

### Write Access

Address Byte (Bits 15-8)	Address Byte (Bits 7-0)	Bit 6-0: Length Bit 7: 0	Data Bytes
-----------------------------	----------------------------	-----------------------------	------------

Figure 3: SPI frame format for Write Request

The header is immediately followed by the data bytes written to the netX internal memory (on MOSI). As the length field is 7 bit in width, a single write access must not exceed a maximum of 124 data bytes. The chip select signal is active during the whole SPI frame transfer.

### Read Access

Address Byte (Bits 15-8)	Address Byte (Bits 7-0)	Bit 6-0: Length Bit 7: 1	Idle Bytes	0xA5 Marker Byte	Data Bytes
-----------------------------	----------------------------	-----------------------------	------------	---------------------	------------

Figure 4: SPI frame format for Read Request

Read access requires the SPI Master to pause a period of time after transferring the header. So the slave device is able to prepare the data to read from netX internal memory. The SPI master generates cyclic idle bytes immediately after the header as long as the SPI slave device does not signal its ready state with a special marker byte (0xA5). The SPI master in turn polls the first data byte (on MISO) by transferring a dummy byte to the SPI slave (on MOSI). As the length field is 7 bit in width and the interface only works dword-granular, a single read access must not exceed a maximum of 124 data bytes. The chip select signal is active during the whole SPI frame transfer.

### 3.3 Example for NXDB500-SYS Evaluation Board

The HAL comes with an example application running on the NXDB500-SYS board. The example uses the netX internal SPI master and the SPI Slave as DPM interface.

For running the example the NXDB500-SYS board must be prepared as follows:

- Remove jumper from J1A and J1B
- Connect SPI\_MISO (T5.3) to XM2\_TX (T1.29)
- Connect SPI\_MOSI (T5.1) to XM2\_RX (T1.30)
- Connect SPI\_CLK (T5.2) to XM2\_ECLK (J71C.2)
- Connect SPI\_CS0 (T5.4) to XM2\_IO0 (T1.31)

The application example cyclically writes/reads back test data via SPI Slave as DPM interface into/from the xPEC memory.

### 3.4 List of Tables

Table 1: List of Revisions .....	3
Table 2: Terms, Abbreviations and Definitions .....	3
Table 3: References .....	3
Table 4: SPISDPM_Deinit() - Function Arguments .....	8
Table 5: SPISDPM_Deinit() - Function Return Values .....	8
Table 6: SPISDPM_GetParameter() - Function Arguments .....	9
Table 7: SPISDPM_GetParameter() - Function Return Values .....	9
Table 8: SPISDPM_Init() - Function Arguments .....	10
Table 9: SPISDPM_Init() - Function Return Values .....	10
Table 10: SPISDPM_SetParameter() - Function Arguments .....	11
Table 11: SPISDPM_SetParameter() - Function Return Values .....	11
Table 12: SPISDPM_Start() - Function Arguments .....	12
Table 13: SPISDPM_Start() - Function Return Values .....	12
Table 14: SPISDPM_GetCounters() - Function Arguments .....	13
Table 15: SPISDPM_GetCounters() - Function Return Values .....	13
Table 16: SPISDPM_CMD_T - Structure .....	14
Table 17: SPISDPM_STATUS_COUNTERS_T - Structure .....	15
Table 18: SPISDPM_PARAMETER - Enumeration .....	16
Table 19: SPISDPM_RESULT_E - Enumeration .....	17
Table 20: SPI timing parameters .....	18

### 3.5 List of Figures

Figure 1: Interface between Interface User and Interface in Relation to Layer Model..... 7

Figure 2: SPI timing..... 18

Figure 3: SPI frame format for Write Request ..... 19

Figure 4: SPI frame format for Read Request..... 19

## 3.6 Contacts

### Headquarters

#### Germany

Hilscher Gesellschaft für  
Systemautomation mbH  
Rheinstrasse 15  
65795 Hattersheim  
Phone: +49 (0) 6190 9907-0  
Fax: +49 (0) 6190 9907-50  
E-Mail: [info@hilscher.com](mailto:info@hilscher.com)

#### Support

Phone: +49 (0) 6190 9907-99  
E-Mail: [de.support@hilscher.com](mailto:de.support@hilscher.com)

### Subsidiaries

#### China

Hilscher Ges.f.Systemaut. mbH  
Shanghai Representative Office  
200010 Shanghai  
Phone: +86 (0) 21-6355-5161  
E-Mail: [info@hilscher.cn](mailto:info@hilscher.cn)

#### Support

Phone: +86 (0) 21-6355-5161  
E-Mail: [cn.support@hilscher.com](mailto:cn.support@hilscher.com)

#### France

Hilscher France S.a.r.l.  
69500 Bron  
Phone: +33 (0) 4 72 37 98 40  
E-Mail: [info@hilscher.fr](mailto:info@hilscher.fr)

#### Support

Phone: +33 (0) 4 72 37 98 40  
E-Mail: [fr.support@hilscher.com](mailto:fr.support@hilscher.com)

#### India

Hilscher India Pvt. Ltd.  
New Delhi - 110 025  
Phone: +91 9810269248  
E-Mail: [info@hilscher.in](mailto:info@hilscher.in)

#### Italy

Hilscher Italia srl  
20090 Vimodrone (MI)  
Phone: +39 02 25007068  
E-Mail: [info@hilscher.it](mailto:info@hilscher.it)

#### Support

Phone: +39/02 25007068  
E-Mail: [it.support@hilscher.com](mailto:it.support@hilscher.com)

#### Japan

Hilscher Japan KK  
Tokyo, 160-0022  
Phone: +81 (0) 3-5362-0521  
E-Mail: [info@hilscher.jp](mailto:info@hilscher.jp)

#### Support

Phone: +81 (0) 3-5362-0521  
E-Mail: [jp.support@hilscher.com](mailto:jp.support@hilscher.com)

#### Korea

Hilscher Korea Inc.  
Suwon-Si, 443-810  
Phone: +82-31-204-6190  
E-Mail: [info@hilscher.kr](mailto:info@hilscher.kr)

#### Switzerland

Hilscher Swiss GmbH  
4500 Solothurn  
Phone: +41 (0) 32 623 6633  
E-Mail: [info@hilscher.ch](mailto:info@hilscher.ch)

#### Support

Phone: +49 (0) 6190 9907-99  
E-Mail: [ch.support@hilscher.com](mailto:ch.support@hilscher.com)

#### USA

Hilscher North America, Inc.  
Lisle, IL 60532  
Phone: +1 630-505-5301  
E-Mail: [info@hilscher.us](mailto:info@hilscher.us)

#### Support

Phone: +1 630-505-5301  
E-Mail: [us.support@hilscher.com](mailto:us.support@hilscher.com)