

Version: 1.0

Release date: 10 May 2017

© 2016 - 2017 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc. ("MediaTek") and/or its licensor(s). MediaTek cannot grant you permission for any material that is owned by third parties. You may only use or reproduce this document if you have agreed to and been bound by the applicable license agreement with MediaTek ("License Agreement") and been granted explicit permission within the License Agreement ("Permitted User"). If you are not a Permitted User, please cease any access or use of this document immediately. Any unauthorized use, reproduction or disclosure of this document in whole or in part is strictly prohibited. THIS DOCUMENT IS PROVIDED ON AN "AS-IS" BASIS ONLY. MEDIATEK EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF ANY KIND AND SHALL IN NO EVENT BE LIABLE FOR ANY CLAIMS RELATING TO OR ARISING OUT OF THIS DOCUMENT OR ANY USE OR INABILITY TO USE THEREOF. Specifications contained herein are subject to change without notice.



Document Revision History

Revision	Date	Description
1.0	10 May 2017	Initial release





Table of Contents

1.	SPIS		1
	1.1.	Overview	1
		Protocol	
	1.3.	Programming Sequence	3
	1.4.	Protocol Timing	5
	1.5.	Interrupt	8
	1.6.	Operation Register	8
	1.7.	Register Descriptions	8





Lists of Tables and Figures





1. SPIS

The SPI slave module translates the 16bits SPI serial protocol to create either Rbus or Pbus master transaction, for accessing DRAM data or configuring registers.

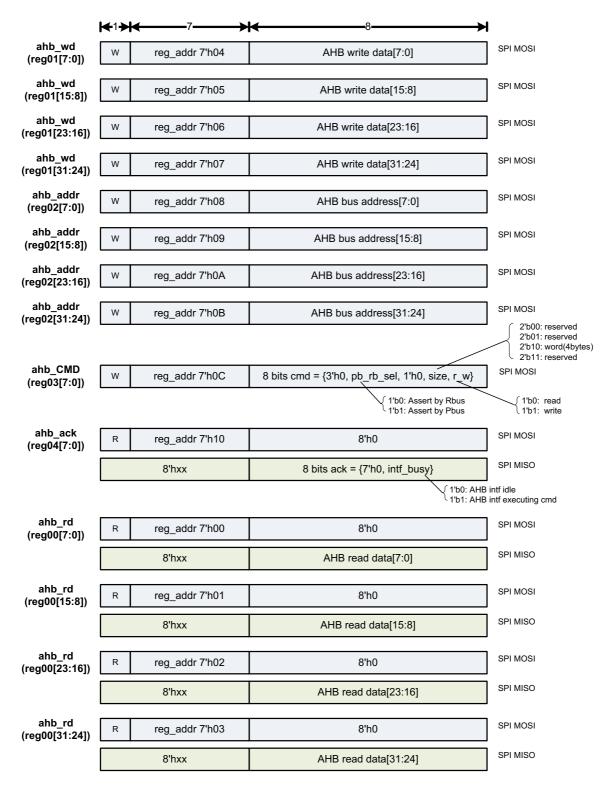
1.1. Overview

There is no software required to run on MediaTek MT7688 SPI slave. The only requirement is to setup pin-control for SPI slave, which means to disable Ethernet ports 1 to 4 (to set in IoT mode).

For SPI master to connect to MT7688 SPI slave, there is a protocol to follow (see section 1.2, "Protocol"), registers REG0 to REG4 in section 1.6.1, "Register of SPI Slave Interface" are used to get DRAM(Rbus) data from or configuring registers(Pbus) on MT7688.



1.2. Protocol



There are 5 registers in this module. Reg00 is the read data from AHB. Reg01 is the write data that programmers want to write to AHB. Reg02 is the address that programmers want to write/read to/from AHB. The configured value must be a **physical address**. Reg03 is the command that applies to AHB protocol. Reg04 is the status for polling to make sure AHB bus is idle or busy.



Before programming AHB/APB registers, programmers should check reg04 bit0 to see if AHB is idle. Programmers can set reg03 (cmd register) to kick SPIS2AHB module to write/read one byte/halfword/word/dword to/from AHB/APB.

Before SPI write/read to/from AHB, programmers should guarantee AHB bus is non-busy by checking spitoahb spi.reg04[0] to see if it equals to 1'b0.

1.3. Programming Sequence

1.3.1. Standard mode

1) Example 1: Write 0x0123 4567 data to the register at address 0x1013 0004

Step1.

Check if SPIS is idle. SPI master asserts following data on SPI MOSI:

```
{1'b0, 7'h10, 8'hxx} // Read SPIS status reg04
```

Wait until the SPI_MISO returned data bit[0] = 0, which indicates the Rbus/Pbus interface of SPIS is idle and ready to execute a new access command.

Step2.

Prepare commands for bus accessing. SPI master asserts following data on SPI MOSI respectively:

```
{1'b1, 7'h04, 8'h67} // put bus write data [7:0] into SPIS reg01[7:0] {1'b1, 7'h05, 8'h45} // put bus write data [15:8] into SPIS reg01[15:8] {1'b1, 7'h06, 8'h23} // put bus write data [23:16] into SPIS reg01[23:16] {1'b1, 7'h07, 8'h01} // put bus write data [31:24] into SPIS reg01[31:24] {1'b1, 7'h08, 8'h04} // put bus address [7:0] into SPIS reg02[7:0] {1'b1, 7'h08, 8'h00} // put bus address [15:8] into SPIS reg02[15:8] {1'b1, 7'h08, 8'h13} // put bus address [23:16] into SPIS reg02[23:16] {1'b1, 7'h08, 8'h10} // put bus address [31:24] into SPIS reg02[31:24] {1'b1, 7'h0C, {3'b0, 1'b0, 1'b0, 2'b10, 1'b1}} // Start the bus write access via Rbus master interface
```

Step3.

Wait for the bus accessing to be done. SPI master asserts following data on SPI MOSI:

```
{1'b0, 7'h10, 8'hxx} // Read SPIS bus interface status
Wait until the SPI MISO returned data bit[0] = 0, and make sure that either Rbus or Pbus finishes the bus access.
```

2) Example 2: Read 0x0123 4567 data from the register at address 0x1013 0004

Step1.

Check if SPIS is idle. SPI master asserts following data on SPI_MOSI:

```
{1'b0, 7'h10, 8'hxx} // Read SPIS status reg04
```

Wait until the SPI_MISO returned data bit[0] = 0, which indicates the Rbus/Pbus interface of SPIS is idle and ready to execute a new access command.



Step2.

Prepare commands for bus accessing. SPI master asserts following data on SPI MOSI respectively:

```
{1'b1, 7'h08, 8'h04} // put bus address [7:0] into SPIS reg02[7:0] {1'b1, 7'h09, 8'h00} // put bus address [15:8] into SPIS reg02[15:8] {1'b1, 7'h0A, 8'h13} // put bus address [23:16] into SPIS reg02[23:16] {1'b1, 7'h0B, 8'h10} // put bus address [31:24] into SPIS reg02[31:24] {1'b1, 7'h0C, {3'b0, 1'b0, 1'b1, 2'b10, 1'b0}} // Start the bus read access via Pbus master interface
```

Step3.

Wait for the bus accessing to be done. SPI master asserts following data on SPI_MOSI:

```
{1'b0, 7'h10, 8'hxx} // Read SPIS status
```

Wait until the SPI_MISO returned data bit[0] = 0, and make sure that either Rbus or Pbus finishes the bus access

Step4.

Get read data. SPI master asserts following data on SPI_MOSI respectively:

```
{1'b0, 7'h00, 8'hxx} // get bus read data [7:0] from SPIS reg00[7:0]

SPIS_MISO return data 16'hxx_67

{1'b0, 7'h01, 8'hxx} // get bus read data [15:8] from SPIS reg00[15:8]

SPIS_MISO return data 16'hxx_45

{1'b0, 7'h02, 8'hxx} // get bus read data [23:16] from SPIS reg00[23:16]

SPIS_MISO return data 16'hxx_23

{1'b0, 7'h03, 8'hxx} // get bus read data [31:24] from SPIS reg00[31:24]

SPIS_MISO return data 16'hxx_01
```

1.3.2. Sequential mode

1) Example 1: Write 0x0123_4567 data to the register at address 0x1013_0004

Step1.

Check if SPIS is idle. SPI master asserts following data on SPI_MOSI:

```
{1'b0, 7'h10, 8'h00} // Read SPIS bus interface status
```

Wait until the SPI_MISO returned data bit[0] = 0, which indicates the Rbus/Pbus interface of SPIS is idle and ready to execute a new access command.

Step2.

Prepare commands for bus accessing. SPI master asserts following data on SPI_MOSI.

```
{1'b1, 7'h04, 8'h67, 8'h45, 8'h23, 8'h01, 8'h04, 8'h00, 8'h13, 8'h10, {3'b0, 1'b0, 2'b10, 1'b1}}

// put bus write data[31:0] into SPIS reg01[31:0], put bus address
[31:0] into SPIS reg02[31:0]

// Start the bus write access via Rbus master interface
```

Step3.

Wait for the bus accessing to be done. SPI master asserts following data on SPI_MOSI:

```
{1'b0, 7'h10, 8'h00} // Read SPIS bus interface status
Wait until the SPI MISO returned data bit[0] = 0, make sure that either Rbus or Pbus finish the bus access.
```



2) Example 2: Read 0x0123_4567 data from the register at address 0x1013_0004

Step1.

Check if SPIS is idle. SPI master asserts following data on SPI MOSI:

```
{1'b0, 7'h10, 8'hxx} // Read SPIS status reg04
```

Wait until the SPI_MISO returned data bit[0] = 0, which indicates the Rbus/Pbus interface of SPIS is idle and ready to execute a new access command.

Step2.

Prepare commands for bus accessing. SPI master asserts following data on SPI_MOSI:

```
{1'b1, 7'h08, 8'h04, 8'h00, 8'h13, 8'h10, {3'b0, 1'b0, 1'b1, 2'b10, 1'b0}}
// put bus address [31:0] into SPIS reg02[31:0]
// Start the bus read access via Pbus master interface
```

Step3.

Wait for the bus accessing to be done. SPI master asserts following data on SPI_MOSI

```
{1'b0, 7'h10, 8'hxx} // Read SPIS status reg04
```

Wait until the SPI_MISO returned data bit[0] = 0, make sure that either Rbus or Pbus finish the bus access.

Step4.

Get read data. SPI master asserts following data on SPI_MOSI:

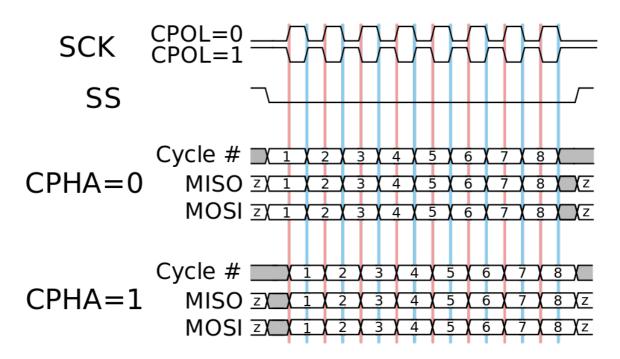
```
{1'b0, 7'h00, 8'hxx, 8'hxx, 8'hxx, 8'hxx} // get bus read data from SPIS reg00[31:0]
```

SPIS MISO return data {8'hxx, 8'h67, 8'h45, 8'h23, 8'h01}

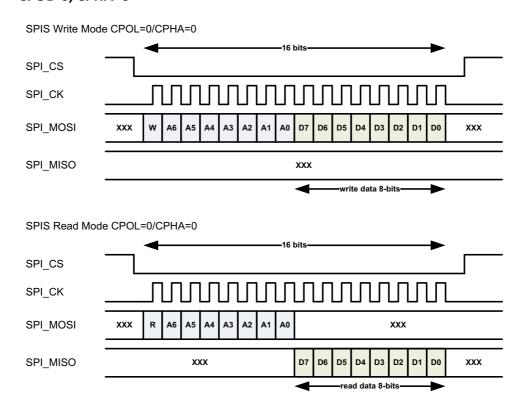
1.4. Protocol Timing

Limitation: Maximum clock frequency: 20MHz



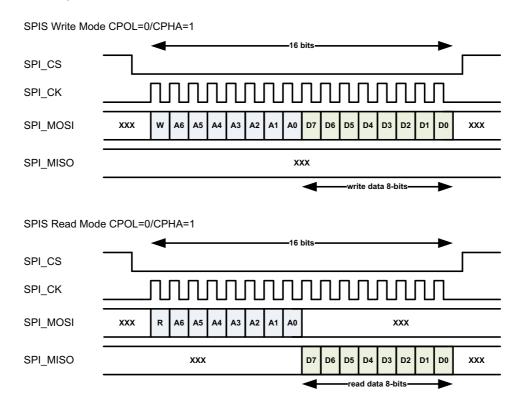


1.4.1. CPOL=0, CPHA=0

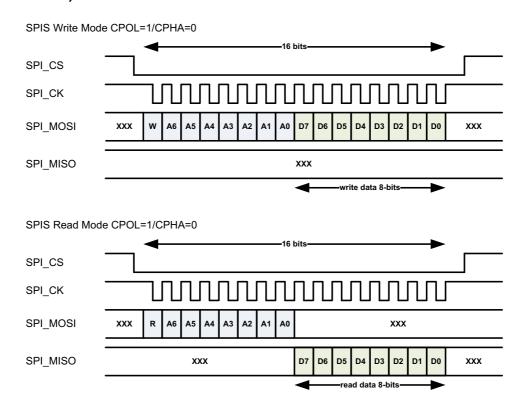




1.4.2. CPOL=0, CPHA=1

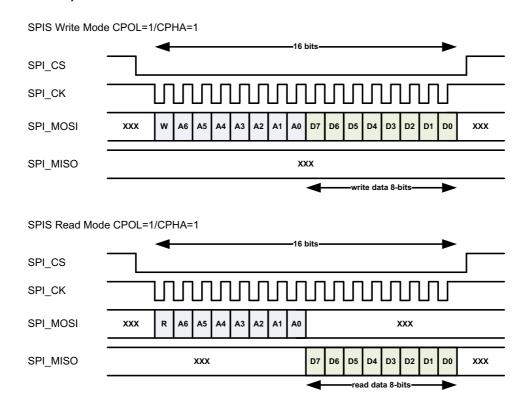


1.4.3. CPOL=1, CPHA=0





1.4.4. CPOL=1, CPHA=1



1.5. Interrupt

SPIS doesn't have dedicated hardware interrupt indication signal. SPI master could configure a software interrupt source of interrupt controller to inform the CPU of SPI slave side.

1.6. Operation Register

1.6.1. Register of SPI Slave Interface

Module name: SPIS Base address: (+0h)

Address	Name	Width	Register Function
00000000	REG00	32	SPI Slave Register 00
00000004	<u>REG01</u>	32	SPI Slave Register 01
00000008	REG02	32	SPI Slave Register 02
000000C	REG03	32	SPI Slave Register 03
00000010	REG04	32	SPI Slave Register 04

1.7. Register Descriptions

00000000 REG00 SPI Slave Register 00 00000000

Bit	31															
Name							bu	s_read_o	data[31::	16]						
Type								R	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	15	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0														
Name							bu	s_read_	data[15:	0]						
Type								R	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	bus_read_data	SPI Slave Register 00 for bus read data

00000004 <u>REG01</u> SPI Slave Register 01 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name							bus	_write_	data[31:	16]						
Type								R'	W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							bu	s_write_	data[15	:0]						
Type		RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	bus_write_data	SPI Slave Register 01 for bus write data

00000008 REG02 SPI Slave Register 02

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		bus address[31:16]														
Type								R\	N							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							t	us_addr	ess[15:0]						
Type		RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	bus_address	SPI Slave Register 02 for bus address
		This address must be physical address

0000000C <u>REG03</u> SPI Slave Register 03 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		reg03_31_5[26:11]														
Type		RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					reg0	3_31_5[:	10:0]					bus_pb _rb_sel	reg03_ 3	bus_	_size	bus_r_ w
Type		RW RW RW RW RW								RW						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:5	reg03_31_5	reg03[31:5] reserved bit
4	bus_pb_rb_sel	Bus interface selection
		0: Bus transaction is asserted by Rbus master interface, can access DRAM and peripheral registers 1: Bus transaction is asserted by Pbus master interface, can peripheral registers only.
3	reg03_3	reg03[3] reserved bit
2:1	bus_size	Bus access size
		00: reserved 01: reserved 10: word (4bytes) 11: reserved
0	bus_r_w	Bus access type



0: read 1: write

0000001	0	REG04			SPI Sla	ve Reg	ister 0	4							000	000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Туре																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																bus_bu
																sy
Type																RO
Reset																0

Bit(s)	Name	Description
0	bus_busy	Bus interface status 0: SPIS bus interface is idle for next access command
		1: SPIS bus interface is busy

1.7.1. Register of Pbus Slave Interface

The CPU at SPIS side could access this Pbus slave interface to probe the SPIS internal registers reg00~reg04 status, and control the SPI polarity/phase. Users can't configure SPIS reg via this pbus slave interface, and SPIS reg must be configured by SPI master.

Module name: spis_pbslv Base address: (+10000700h)

Address	Name	Width	Register Function
10000700	SPIS REGOO	32	SPI Slave Register 00
10000704	SPIS_REG01	32	SPI Slave Register 01
10000708	SPIS_REG02	32	SPI Slave Register 02
1000070C	SPIS_REG03	32	SPI Slave Register 03
10000710	SPIS REG04	32	SPI Slave Register 04
10000740	SPIS_CFG	32	SPI Slave Configuration

10000700 <u>SPIS_REG00</u> SPI Slave Register 00 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	spis_reg00[31:16]															
Type								R								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								spis_reg	00[15:0]							
Type		RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:0	spis_reg00	SPI Slave Register 00

10000704 <u>SPIS_REG01</u> SPI Slave Register 01 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	spis_reg01[31:16]															
Type								R	0							
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0										0					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								spis_reg	01[15:0]							
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit(s)	Name	е			Descrip	tion										
31:0	spis_r	eg01			SPI Slave	Registe	r 01									
1000070	8	SPIS F	REG02		SPI Sla	ve Reg	ister 0	2							00	000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name						•		spis_reg(02[31:16]							
Туре		1 _	1 -		1 -	T			0							
Reset	0	0	0	0	0	0	0	0	7	0 6	5	0	0	0	0	0
Bit Name	15	14	13	12	11	10	9	8	02[15:0]	ь	5	4	3	2	1	0
Type									02[13.0] 0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'																
Bit(s)	Name	e			Descrip	tion										
31:0	spis_r	reg02			SPI Slave		r 02									
	3P13_1				31 1 31440	перые	. 02									
1000070	_	SPIS F	ECU3		SPI Sla	vo Pog	ictor O	2							00	000000
											T	T		1		
Bit	31	30	29	28	27	26	25	. 24	23	22	21	20	19	18	17	16
Name Type							;		03[31:16] O							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								spis_reg	03[15:0]							
Туре									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit(s)	Name	е			Descrip	tion										
31:0	spis_r	-απΩ3														
		egos			SPI Slave	Registe	r 03									
		egos			SPI Slave	Registe	r 03									
1000071	0	SPIS F	REG04		SPI Slave			4							00	000000
-	0 31		REG04 29	28				4 24	23	22	21	20	19	18	00	000000
1000071		SPIS F			SPI Sla	ve Reg	ister 0 25	24	23 04[31:16]		21	20	19	18		
1000071 Bit Name Type	31	SPIS F	29	28	SPI Sla	ve Reg	ister 0	24 spis_reg(04[31:16 O						17	16
1000071 Bit Name Type Reset	31	SPIS F 30	29	28	SPI Sla	ve Reg	25 0	24 spis_reg(R	04[31:16] O 0	0	0	0	0	0	17	16
Bit Name Type Reset Bit	31	SPIS F	29	28	SPI Sla	ve Reg	25 0 9	24 spis_reg(R 0 8	04[31:16] O 0 7						17	16
Bit Name Type Reset Bit Name	31	SPIS F 30	29	28	SPI Sla	ve Reg	25 0 9	24 spis_regular R 0 8 spis_regular R	04[31:16] O 0	0	0	0	0	0	17	16
Bit Name Type Reset Bit	31	SPIS F 30	29	28	SPI Sla	ve Reg	25 0 9	24 spis_regular R 0 8 spis_regular R	04[31:16] O 0 7 04[15:0]	0	0	0	0	0	17	16
Bit Name Type Reset Bit Name Type	0 15	SPIS F 30 0 14	0 13	28 0 12	SPI Sla	ve Reg 26 0 10	25 0 9	24 spis_regular R 0 8 spis_regular	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Reset Reset	0 15	SPIS F 30 0 14 0	0 13	0 12	SPI Sla	ve Reg 26 0 10 0 0	25 0 9	24 spis_regular R 0 8 spis_regular	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset	0 15 0 Name	SPIS F 30 0 14 0 0	0 13	0 12	SPI Sla 27 0 11	ve Reg	25 0 9 0	24 spis_regular R 0 8 spis_regular	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Reset Reset	0 15	SPIS F 30 0 14 0 0	0 13	0 12	SPI Sla 27 0 111 0	ve Reg	25 0 9 0	24 spis_regular R 0 8 spis_regular	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Bit Same Type Reset Bit Same Type Reset Bit(s)	0 15 0 Name	SPIS F 30 30 14 0 14 0 0 eeeeg04	0 13	0 12	SPI Sla 27 0 11 0 Descript	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Sit Name Type Reset Type Reset Bit(s) 31:0	0 15 0 Name spis_r	0	0 13 0 CFG	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0 tion	04[31:16] 0	0 6	0 5	0 4	0 3	0 2	0 1 0 0 00	16
Bit Name Type Reset Bit Name Type Reset Sit Name Type Reset 1000074 Bit	0 15 0 Name	SPIS F 30 30 14 0 14 0 0 eeeeg04	0 13	0 12	SPI Sla 27 0 11 0 Descript	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0	04[31:16] 0 0 7 04[15:0]	0	0 5	0 4	0 3	0 2	0 1	0 0
Bit Name Type Reset Bit Name Type Reset Sit Name Type Reset 1000074 Bit Name	0 15 0 Name spis_r	0	0 13 0 CFG	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0 tion	04[31:16] 0	0 6	0 5	0 4	0 3	0 2	0 1 0 0 00	16 0 0
Bit Name Type Reset Bit Name Type Reset Sit Name Type Reset 1000074 Bit	0 15 0 Name spis_r	0	0 13 0 CFG	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0 tion	04[31:16] 0	0 6	0 5	0 4	0 3	0 2	0 1 0 0 00	16 0 0
Bit Name Type Reset Bit Name Type	0 15 0 Name spis_r	0	0 13 0 CFG	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave	ve Reg 26 0 10 0 tion Register	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0 0 tion	04[31:16] 0	0 6	0 5	0 4	0 3	0 2	0 1 0 0 00	16 0 0
Bit Name Type Reset Bit Name Type Reset Sit Name Type Reset Bit(s) 31:0 1000074 Bit Name Type Reset Bit Name Type Reset Name Type	0 15 Name spis_r	SPIS F 30 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 13 0 CFG 29	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave SPI Sla 27	ve Reg 26 0 10 0 tion Register ve Con 26	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O O O O O O O O O O O O O O O O O	0 6	0 5	0 4	0 3	0 2	00 17 00 17 1 spis	0000000 16 0 mode
Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit(s) 31:0 1000074 Bit Name Type Reset Bit Name Type Reset Bit	0 15 Name spis_r	SPIS F 30 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 13 0 CFG 29	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave SPI Sla 27	ve Reg 26 0 10 0 tion Register ve Con 26	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O O O O O O O O O O O O O O O O O	0 6	0 5	0 4	0 3	0 2	00 1 17 0 1 1 spis F	00000000000000000000000000000000000000
Bit Name Type Reset Bit Name Reset Bit Name	0 15 Name spis_r	SPIS F 30 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 13 0 CFG 29	0 12	SPI Sla 27 0 11 0 Descripi SPI Slave SPI Sla 27	ve Reg 26 0 10 0 tion Register ve Con 26	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O O O O O O O O O O O O O O O O O	0 6	0 5	0 4	0 3	0 2	00 17 00 17 1 spis	0000000 16 0 mode
Bit Name Type Reset Name Type Reset Bit Name Type Reset Bit Name Type Reset Reset	0 15 Name spis_r	SPIS F 30	0 13 0 CFG 29	0 12 0 28 28 12	SPI Sla 27 0 11 0 Description SPI Slave SPI Slave 127	ve Reg 26 0 10 tion Register 26 10	25 0 9 0 0	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O O O O O O O O O O O O O O O O O	0 6	0 5	0 4	0 3	0 2	00 1 17 0 1 1 spis F	00000000000000000000000000000000000000
Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit(s) 31:0 1000074 Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit Name	0 15 Name	SPIS F 30	0 13 0 CFG 29	0 12 0 28 28 12 12	SPI Sla 27 0 11 0 Descript SPI Slave SPI Slave 11 Descript	ve Reg 26 0 10 tion Register 26 10 tion	0 25 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O 7 O4[15:0] O O 7	0 6	0 5	0 4	0 3	0 2	00 1 17 0 1 1 spis F	00000000000000000000000000000000000000
Bit Name Type Reset Name Type Reset Bit Name Type Reset Bit Name Type Reset Reset	0 15 Name spis_r	SPIS F 30	0 13 0 CFG 29	28 0 12 0 28	SPI Sla 27 0 11 0 Descript SPI Slave SPI Slave SPI Slave SPI Slave SPI Slave	ve Reg 26 0 10 tion Register 26 10 tion clock pc	25	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O 7 O4[15:0] O O 7	0 6	0 5	0 4	0 3	0 2	00 1 17 0 1 1 spis F	00000000000000000000000000000000000000
Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit(s) 31:0 1000074 Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit Name Type Reset Bit	0 15 Name	SPIS F 30	0 13 0 CFG 29	0 12 0 28 12 12	SPI Sla 27 0 11 0 Descript SPI Slave SPI Slave 11 Descript	ve Reg 26 0 10 tion Register 26 10 clock pc POL=0, CD	25	24 spis_regg R 0 8 spis_regg R 0	D4[31:16] O O O 7 O4[15:0] O O 7	0 6	0 5	0 4	0 3	0 2	00 1 17 0 1 1 spis F	00000000000000000000000000000000000000



2'b10: CPOL=1, CPHA=0 2'b11: CPOL=1, CPHA=1