

AC784xx_DFP SPI

10.1.0

Generated by Doxygen 1.8.13

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	2
2.1	File List	2
3	Class Documentation	3
3.1	Spi_HalConfigType Struct Reference	3
3.1.1	Detailed Description	3
3.1.2	Member Data Documentation	4
3.1.2.1	BaudRate	4
3.1.2.2	Callback	4
3.1.2.3	ContinuousCs	4
3.1.2.4	Cpha	4
3.1.2.5	Cpol	4
3.1.2.6	CsHold	5
3.1.2.7	CsIdle	5
3.1.2.8	CsOutputEn	5
3.1.2.9	CsSetup	5
3.1.2.10	FrmSize	5
3.1.2.11	HreqEn	6
3.1.2.12	HreqPol	6
3.1.2.13	Mode	6
3.1.2.14	MsbFirst	6
3.1.2.15	PcsCfg	6

3.1.2.16	PcsPol	7
3.1.2.17	PinCfg	7
3.1.2.18	RxDmaChannel	7
3.1.2.19	TxDmaChannel	7
3.1.2.20	Width	7
3.2	Spi_StateType Struct Reference	8
3.2.1	Detailed Description	8
3.2.2	Member Data Documentation	8
3.2.2.1	Callback	8
3.2.2.2	FrmSize	8
3.2.2.3	InProgress	9
3.2.2.4	MsbFirst	9
3.2.2.5	RxBuffPtr	9
3.2.2.6	RxBuffPtr1	9
3.2.2.7	RxCount	9
3.2.2.8	RxDmaChannel	10
3.2.2.9	Status	10
3.2.2.10	TransferType	10
3.2.2.11	TxBuffPtr	10
3.2.2.12	TxCount	10
3.2.2.13	TxDmaChannel	10
3.2.2.14	XferCount	10

4 File Documentation	11
4.1 AC784xx_API_Reference_Manual_SPI.pdf File Reference	11
4.2 AC784xx_Spi_Reg.h File Reference	11
4.2.1 Detailed Description	13
4.2.2 Function Documentation	13
4.2.2.1 Spi_Reg_ClearRDMF()	13
4.2.2.2 Spi_Reg_ClearRxOF()	14
4.2.2.3 SPI_Reg_ClearStatusFlag()	14
4.2.2.4 Spi_Reg_ClearTxUF()	15
4.2.2.5 Spi_Reg_GetDataAddress()	15
4.2.2.6 Spi_Reg_GetIntMode()	16
4.2.2.7 Spi_Reg_GetRxOnly()	16
4.2.2.8 Spi_Reg_GetSckHighLow()	17
4.2.2.9 Spi_Reg_GetStatusFlag()	17
4.2.2.10 Spi_Reg_GetTxOnly()	18
4.2.2.11 SPI_Reg_IsCsContinuousMode()	18
4.2.2.12 Spi_Reg_IsMaster()	19
4.2.2.13 SPI_Reg_IsModuleEnabled()	19
4.2.2.14 Spi_Reg_ReadData()	20
4.2.2.15 Spi_Reg_ReleaseCS()	20
4.2.2.16 Spi_Reg_SetBaudRate()	21
4.2.2.17 Spi_Reg_SetContinuousCS()	21
4.2.2.18 Spi_Reg_SetCPHA()	22
4.2.2.19 Spi_Reg_SetCPOL()	22
4.2.2.20 Spi_Reg_SetCSHold()	23
4.2.2.21 Spi_Reg_SetCSIdle()	23
4.2.2.22 Spi_Reg_SetCSOE()	24
4.2.2.23 Spi_Reg_SetCSSetup()	25
4.2.2.24 Spi_Reg_SetDataWidth()	25
4.2.2.25 Spi_Reg_SetDebug()	26
4.2.2.26 Spi_Reg_SetDMIE()	26

4.2.2.27	Spi_Reg_SetEnable()	27
4.2.2.28	Spi_Reg_SetFrameSize()	27
4.2.2.29	Spi_Reg_SetHreq()	28
4.2.2.30	Spi_Reg_SetHreqPolarity()	28
4.2.2.31	Spi_Reg_SetIntMode()	29
4.2.2.32	SPI_Reg_SetMasterModeFaultDetectCs0()	29
4.2.2.33	SPI_Reg_SetMasterModeFaultDetectCs1()	30
4.2.2.34	SPI_Reg_SetMasterModeFaultDetectCs2()	30
4.2.2.35	SPI_Reg_SetMasterModeFaultDetectCs3()	31
4.2.2.36	Spi_Reg_SetMasterNoOverflowMode()	31
4.2.2.37	Spi_Reg_SetMasterSlaveMode()	31
4.2.2.38	Spi_Reg_SetMatchData()	32
4.2.2.39	Spi_Reg_SetModeFault()	32
4.2.2.40	Spi_Reg_SetPcsCfg()	33
4.2.2.41	Spi_Reg_SetPcsPolarity()	34
4.2.2.42	Spi_Reg_SetPinConfigMode()	34
4.2.2.43	Spi_Reg_SetROTRIG()	35
4.2.2.44	Spi_Reg_SetRxDmaCmd()	35
4.2.2.45	Spi_Reg_SetRxMSB()	36
4.2.2.46	Spi_Reg_SetRxOnly()	36
4.2.2.47	Spi_Reg_SetTxDmaCmd()	37
4.2.2.48	Spi_Reg_SetTxMSB()	37
4.2.2.49	Spi_Reg_SetTxOnly()	38
4.2.2.50	Spi_Reg_SoftwareReset()	38
4.2.2.51	Spi_Reg_WriteData()	39
4.3	Spi_Hal.c File Reference	39
4.3.1	Detailed Description	41
4.3.2	Macro Definition Documentation	41
4.3.2.1	SPI_1BYTE_MSK	41
4.3.2.2	SPI_2BYTES_MSK	42
4.3.2.3	SPI_BIT16_POS	42

4.3.2.4	SPI_BIT24_POS	42
4.3.2.5	SPI_BIT8_POS	42
4.3.2.6	SPI_TIMEOUT_FRAME	42
4.3.3	Enumeration Type Documentation	42
4.3.3.1	Spi_TransceiveType	42
4.3.4	Function Documentation	43
4.3.4.1	SPI0_IRQHandler()	43
4.3.4.2	SPI1_IRQHandler()	43
4.3.4.3	SPI2_IRQHandler()	44
4.3.4.4	Spi_Hal_AbortTransceive()	44
4.3.4.5	Spi_Hal_CheckBusy()	44
4.3.4.6	Spi_Hal_ConfigureBus()	45
4.3.4.7	Spi_Hal_ConfigureFrame()	45
4.3.4.8	Spi_Hal_DeInit()	46
4.3.4.9	Spi_Hal_GetBase()	46
4.3.4.10	Spi_Hal_GetBaudRate()	47
4.3.4.11	Spi_Hal_GetDefaultConfig()	47
4.3.4.12	Spi_Hal_GetTransceiveStatus()	48
4.3.4.13	Spi_Hal_Init()	48
4.3.4.14	Spi_Hal_SetBaudRate()	49
4.3.4.15	Spi_Hal_SetCsPin()	49
4.3.4.16	Spi_Hal_SetFrameSize()	50
4.3.4.17	Spi_Hal_SetMatchData()	50
4.3.4.18	Spi_Hal_SetPinMode()	51
4.3.4.19	Spi_Hal_SlaveGetReceiveLen()	51
4.3.4.20	Spi_Hal_SlaveReceiveNolimitLen()	52
4.3.4.21	Spi_Hal_SoftwareReset()	52
4.3.4.22	Spi_Hal_TransceiveDma()	53
4.3.4.23	Spi_Hal_TransceiveInt()	53
4.3.4.24	Spi_Hal_TransceivePoll()	54
4.4	Spi_Hal.h File Reference	54

4.4.1	Detailed Description	55
4.4.2	Function Documentation	55
4.4.2.1	Spi_Hal_AbortTransceive()	56
4.4.2.2	Spi_Hal_CheckBusy()	56
4.4.2.3	Spi_Hal_ConfigureBus()	57
4.4.2.4	Spi_Hal_ConfigureFrame()	57
4.4.2.5	Spi_Hal_DeInit()	58
4.4.2.6	Spi_Hal_GetBase()	58
4.4.2.7	Spi_Hal_GetBaudRate()	58
4.4.2.8	Spi_Hal_GetDefaultConfig()	59
4.4.2.9	Spi_Hal_GetTransceiveStatus()	60
4.4.2.10	Spi_Hal_Init()	60
4.4.2.11	Spi_Hal_SetBaudRate()	60
4.4.2.12	Spi_Hal_SetCsPin()	61
4.4.2.13	Spi_Hal_SetFrameSize()	61
4.4.2.14	Spi_Hal_SetMatchData()	62
4.4.2.15	Spi_Hal_SetPinMode()	62
4.4.2.16	Spi_Hal_SlaveGetReceiveLen()	63
4.4.2.17	Spi_Hal_SlaveReceiveNolimitLen()	63
4.4.2.18	Spi_Hal_SoftwareReset()	64
4.4.2.19	Spi_Hal_TransceiveDma()	64
4.4.2.20	Spi_Hal_TransceiveInt()	65
4.4.2.21	Spi_Hal_TransceivePoll()	65
4.5	Spi_Hal_Types.h File Reference	66
4.5.1	Detailed Description	67
4.5.2	Typedef Documentation	67
4.5.2.1	Spi_CallbackType	67
4.5.3	Enumeration Type Documentation	67
4.5.3.1	anonymous enum	67
4.5.3.2	Spi_CphaType	68
4.5.3.3	Spi_CpolType	68
4.5.3.4	Spi_HreqPolarityType	69
4.5.3.5	Spi_InterruptType	69
4.5.3.6	Spi_ModeType	69
4.5.3.7	Spi_PcsPolarityType	70
4.5.3.8	Spi_PcsType	70
4.5.3.9	Spi_PinCfgType	70
4.5.3.10	Spi_StatusFlagType	70
4.5.3.11	Spi_TransceiveStatusType	71
4.5.3.12	Spi_TransferType	71
4.5.3.13	Spi_TransferWidthType	72

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Spi_HalConfigType	User configuration structure for the hardware SPI driver	3
Spi_StateType	Runtime state structure for the device on the SPI bus	8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

AC784xx_API_Reference_Manual_SPI.pdf	11
AC784xx_Spi_Reg.h	
Spi driver register header file	11
Spi_Hal.c	
Spi driver hal api source file	39
Spi_Hal.h	
Spi driver hal api header file	54
Spi_Hal_Types.h	
Spi driver hal types header file	66

Chapter 3

Class Documentation

3.1 Spi_HalConfigType Struct Reference

User configuration structure for the hardware SPI driver.

```
#include <Spi_Hal_Types.h>
```

Public Attributes

- uint8 [CsSetup](#)
- uint8 [CsHold](#)
- uint8 [CsIdle](#)
- uint8 [FrmSize](#)
- uint8 [TxDmaChannel](#)
- uint8 [RxDmaChannel](#)
- boolean [MsbFirst](#)
- boolean [CsOutputEn](#)
- boolean [ContinuousCs](#)
- boolean [HreqEn](#)
- [Spi_ModeType](#) Mode
- [Spi_CpolType](#) Cpol
- [Spi_CphaType](#) Cpha
- [Spi_PcsType](#) PcsCfg
- [Spi_PcsPolarityType](#) PcsPol
- [Spi_HreqPolarityType](#) HreqPol
- [Spi_TransferWidthType](#) Width
- [Spi_PinCfgType](#) PinCfg
- [Spi_CallbackType](#) Callback
- uint32 [BaudRate](#)

3.1.1 Detailed Description

User configuration structure for the hardware SPI driver.

Definition at line 185 of file [Spi_Hal_Types.h](#).

3.1.2 Member Data Documentation

3.1.2.1 BaudRate

`uint32 Spi_HalConfigType::BaudRate`

Set the SPI Transfer baudrate

Definition at line 206 of file Spi_Hal_Types.h.

3.1.2.2 Callback

`Spi_CallbackType Spi_HalConfigType::Callback`

Calling the callback to transfer complete

Definition at line 205 of file Spi_Hal_Types.h.

3.1.2.3 ContinuousCs

`boolean Spi_HalConfigType::ContinuousCs`

Keeps PCS asserted until transfer complete

Definition at line 195 of file Spi_Hal_Types.h.

3.1.2.4 Cpha

`Spi_CphaType Spi_HalConfigType::Cpha`

Selects which phase of clock to capture data

Definition at line 199 of file Spi_Hal_Types.h.

3.1.2.5 Cpol

`Spi_CpolType Spi_HalConfigType::Cpol`

Selects clock polarity

Definition at line 198 of file Spi_Hal_Types.h.

3.1.2.6 CsHold

```
uint8 Spi_HalConfigType::CsHold
```

CS_HOLD, Timing between clock and chip select

Definition at line 188 of file Spi_Hal_Types.h.

3.1.2.7 Csidle

```
uint8 Spi_HalConfigType::CsIdle
```

CS_IDLE, Timing between chip select and chip select

Definition at line 189 of file Spi_Hal_Types.h.

3.1.2.8 CsOutputEn

```
boolean Spi_HalConfigType::CsOutputEn
```

The chip select output is enabled or not

Definition at line 194 of file Spi_Hal_Types.h.

3.1.2.9 CsSetup

```
uint8 Spi_HalConfigType::CsSetup
```

CS_SETUP, Timing between chip select and clock

Definition at line 187 of file Spi_Hal_Types.h.

3.1.2.10 FrmSize

```
uint8 Spi_HalConfigType::FrmSize
```

Number of bits per frame, support 4~32 bits

Definition at line 190 of file Spi_Hal_Types.h.

3.1.2.11 HreqEn

```
boolean Spi_HalConfigType::HreqEn
```

Set Hreq is enabled or not

Definition at line 196 of file Spi_Hal_Types.h.

3.1.2.12 HreqPol

```
Spi_HreqPolarityType Spi_HalConfigType::HreqPol
```

Hreq polarity

Definition at line 202 of file Spi_Hal_Types.h.

3.1.2.13 Mode

```
Spi_ModeType Spi_HalConfigType::Mode
```

Selects master or slave mode

Definition at line 197 of file Spi_Hal_Types.h.

3.1.2.14 MsbFirst

```
boolean Spi_HalConfigType::MsbFirst
```

Option to transmit MSB first

Definition at line 193 of file Spi_Hal_Types.h.

3.1.2.15 PcsCfg

```
Spi_PcsType Spi_HalConfigType::PcsCfg
```

Selects which PCS to use

Definition at line 200 of file Spi_Hal_Types.h.

3.1.2.16 PcsPol

`Spi_PcsPolarityType` `Spi_HalConfigType::PcsPol`

PCS polarity

Definition at line 201 of file `Spi_Hal_Types.h`.

3.1.2.17 PinCfg

`Spi_PinCfgType` `Spi_HalConfigType::PinCfg`

SPI pin (SOUT and SIN) configuration

Definition at line 204 of file `Spi_Hal_Types.h`.

3.1.2.18 RxDmaChannel

`uint8` `Spi_HalConfigType::RxDmaChannel`

DMA rx channel. It will be ignored if DMA mode isn't used.

Definition at line 192 of file `Spi_Hal_Types.h`.

3.1.2.19 TxDmaChannel

`uint8` `Spi_HalConfigType::TxDmaChannel`

DMA tx channel. It will be ignored if DMA mode isn't used.

Definition at line 191 of file `Spi_Hal_Types.h`.

3.1.2.20 Width

`Spi_TransferWidthType` `Spi_HalConfigType::Width`

Selects WIDTH

Definition at line 203 of file `Spi_Hal_Types.h`.

The documentation for this struct was generated from the following file:

- [Spi_Hal_Types.h](#)

3.2 Spi_StateType Struct Reference

Runtime state structure for the device on the SPI bus.

```
#include <Spi_Hal_Types.h>
```

Public Attributes

- uint8 [FrmSize](#)
- uint8 [TxDmaChannel](#)
- uint8 [RxDmaChannel](#)
- boolean [InProgress](#)
- boolean [MsbFirst](#)
- uint16 [TxCount](#)
- uint16 [RxCount](#)
- uint16 [XferCount](#)
- uint16 [Status](#)
- const uint8 * [TxBuffPtr](#)
- uint8 * [RxBuffPtr](#)
- uint8 * [RxBuffPtr1](#)
- [Spi_CallbackType](#) [Callback](#)
- [Spi_TransferType](#) [TransferType](#)

3.2.1 Detailed Description

Runtime state structure for the device on the SPI bus.

Definition at line 211 of file `Spi_Hal_Types.h`.

3.2.2 Member Data Documentation

3.2.2.1 Callback

```
Spi\_CallbackType Spi_StateType::Callback
```

Calling the callback to transfer complete

Definition at line 225 of file `Spi_Hal_Types.h`.

3.2.2.2 FrmSize

```
uint8 Spi_StateType::FrmSize
```

Number of bits per frame, support 4~32 bits

Definition at line 213 of file `Spi_Hal_Types.h`.

3.2.2.3 InProgress

```
boolean Spi_StateType::InProgress
```

True if transfer is active

Definition at line 216 of file Spi_Hal_Types.h.

3.2.2.4 MsbFirst

```
boolean Spi_StateType::MsbFirst
```

Option to transmit MSB first

Definition at line 217 of file Spi_Hal_Types.h.

3.2.2.5 RxBuffPtr

```
uint8* Spi_StateType::RxBuffPtr
```

Receive buffer base pointer

Definition at line 223 of file Spi_Hal_Types.h.

3.2.2.6 RxBuffPtr1

```
uint8* Spi_StateType::RxBuffPtr1
```

Receive buffer base pointer

Definition at line 224 of file Spi_Hal_Types.h.

3.2.2.7 RxCount

```
uint16 Spi_StateType::RxCount
```

Receive bytes counter

Definition at line 219 of file Spi_Hal_Types.h.

3.2.2.8 RxDmaChannel

```
uint8 Spi_StateType::RxDmaChannel
```

DMA rx channel. It will be ignored if DMA mode isn't used.

Definition at line 215 of file Spi_Hal_Types.h.

3.2.2.9 Status

```
uint16 Spi_StateType::Status
```

The current transfer status

Definition at line 221 of file Spi_Hal_Types.h.

3.2.2.10 TransferType

```
Spi_TransferType Spi_StateType::TransferType
```

Type of SPI transfer, DMA or interrupt

Definition at line 226 of file Spi_Hal_Types.h.

3.2.2.11 TxBuffPtr

```
const uint8* Spi_StateType::TxBuffPtr
```

Transmit buffer base pointer

Definition at line 222 of file Spi_Hal_Types.h.

3.2.2.12 TxCount

```
uint16 Spi_StateType::TxCount
```

Transmit bytes counter

Definition at line 218 of file Spi_Hal_Types.h.

3.2.2.13 TxDmaChannel

```
uint8 Spi_StateType::TxDmaChannel
```

DMA tx channel. It will be ignored if DMA mode isn't used.

Definition at line 214 of file Spi_Hal_Types.h.

3.2.2.14 XferCount

```
uint16 Spi_StateType::XferCount
```

Number of bytes to transfer

Definition at line 220 of file Spi_Hal_Types.h.

The documentation for this struct was generated from the following file:

- [Spi_Hal_Types.h](#)

Chapter 4

File Documentation

4.1 AC784xx_API_Reference_Manual_SPI.pdf File Reference

4.2 AC784xx_Spi_Reg.h File Reference

Spi driver register header file.

```
#include "Spi_Hal_Types.h"
```

Functions

- LOCAL_INLINE void [Spi_Reg_SoftwareReset](#) (SPI_Type *Base)
SPI Module softreset. And it only reset master engine/buffer/flag logic, slave buffer/flag logic, the configuration of control registers(CFG0/ CFG1/CFG2/CMD) will not be reset.
- LOCAL_INLINE void [Spi_Reg_SetCSSetup](#) (SPI_Type *Base, uint8 Delay)
Set CS_SETUP time.
- LOCAL_INLINE void [Spi_Reg_SetCSHold](#) (SPI_Type *Base, uint8 Delay)
Set CS_HOLD time.
- LOCAL_INLINE void [Spi_Reg_SetCSIdle](#) (SPI_Type *Base, uint8 Delay)
Set CS_IDLE time.
- LOCAL_INLINE void [Spi_Reg_SetPcsPolarity](#) (SPI_Type *Base, [Spi_PcsType](#) PcsCfg, [Spi_PcsPolarityType](#) PcsPolarity)
Configures the SPI PCS polarity.
- LOCAL_INLINE void [Spi_Reg_SetBaudRate](#) (SPI_Type *Base, uint8 SckLow, uint8 SckHigh)
Set SPI CS baudrate, 1/(low_time + high_time).
- LOCAL_INLINE void [Spi_Reg_GetSckHighLow](#) (const SPI_Type *Base, uint8 *SckLow, uint8 *SckHigh)
Get SPI SckLow and SckHigh.
- LOCAL_INLINE void [Spi_Reg_SetFrameSize](#) (SPI_Type *Base, uint8 FrameSize)
Set SPI frame size(4~32 bits per frame).
- LOCAL_INLINE void [Spi_Reg_SetEnable](#) (SPI_Type *Base, boolean Cmd)
Enable/Disable SPI module.
- LOCAL_INLINE boolean [Spi_Reg_IsModuleEnabled](#) (const SPI_Type *base)
Check if SPI module is enabled.
- LOCAL_INLINE void [Spi_Reg_SetMasterSlaveMode](#) (SPI_Type *Base, [Spi_ModeType](#) Mode)
Set SPI master or slave mode.
- LOCAL_INLINE boolean [Spi_Reg_IsMaster](#) (const SPI_Type *Base)

- Returns whether the SPI module is in master mode.*
- LOCAL_INLINE boolean [SPI_Reg_IsCsContinuousMode](#) (const SPI_Type *base)
- Returns whether the SPI module is in Cs Continuous mode.*
- LOCAL_INLINE boolean [Spi_Reg_GetStatusFlag](#) (const SPI_Type *Base, [Spi_StatusFlagType](#) StatusFlag)
- Gets the SPI status flag enable.*
- LOCAL_INLINE void [SPI_Reg_ClearStatusFlag](#) (SPI_Type *Base)
- Clear SPI Status Flag.*
- LOCAL_INLINE void [Spi_Reg_ClearTxUF](#) (SPI_Type *Base)
- Clear current SPI Tx underflow flag.*
- LOCAL_INLINE void [Spi_Reg_ClearRxOF](#) (SPI_Type *Base)
- Clear current SPI Rx overflow flag.*
- LOCAL_INLINE void [Spi_Reg_SetIntMode](#) (SPI_Type *Base, [Spi_InterruptType](#) InterruptSrc, boolean Cmd)
- Configures the SPI interrupts.*
- LOCAL_INLINE boolean [Spi_Reg_GetIntMode](#) (const SPI_Type *Base, [Spi_InterruptType](#) InterruptSrc)
- Returns if the SPI interrupt request is enabled or disabled.*
- LOCAL_INLINE void [Spi_Reg_SetTxDmaCmd](#) (SPI_Type *Base, boolean Cmd)
- Enable/disable the SPI Transmit Data DMA request.*
- LOCAL_INLINE void [Spi_Reg_SetRxDmaCmd](#) (SPI_Type *Base, boolean Cmd)
- Enable/disable the SPI Receive Data DMA request.*
- LOCAL_INLINE void [Spi_Reg_SetPinConfigMode](#) (SPI_Type *Base, [Spi_PinCfgType](#) PinCfg)
- Configures the SPI SOUT/SIN pin configuration mode.*
- LOCAL_INLINE void [Spi_Reg_SetCPHA](#) (SPI_Type *Base, [Spi_CphaType](#) ClkPhase)
- Set SPI CPHA.*
- LOCAL_INLINE void [Spi_Reg_SetCPOL](#) (SPI_Type *Base, [Spi_CpolType](#) ClkPolarity)
- Set SPI CPOL.*
- LOCAL_INLINE void [Spi_Reg_SetRxMSB](#) (SPI_Type *Base, boolean Cmd)
- Set SPI RX MSB first.*
- LOCAL_INLINE void [Spi_Reg_SetTxMSB](#) (SPI_Type *Base, boolean Cmd)
- Set SPI TX MSB first.*
- LOCAL_INLINE void [Spi_Reg_SetContinuousCS](#) (SPI_Type *Base, boolean Cmd)
- Set SPI CS continuous or disContinuous mode.*
- LOCAL_INLINE void [Spi_Reg_SetCSOE](#) (SPI_Type *Base, boolean Cmd)
- Set SPI CS output enable.*
- LOCAL_INLINE void [Spi_Reg_SetModeFault](#) (SPI_Type *Base, boolean Cmd)
- Enable/Disable SPI CS mode fault detect function.*
- LOCAL_INLINE void [Spi_Reg_ReleaseCS](#) (SPI_Type *Base)
- Release SPI CS, which will be changed to inactive enable.*
- LOCAL_INLINE void [Spi_Reg_SetPcsCfg](#) (SPI_Type *Base, [Spi_PcsType](#) PcsCfg)
- Sets the PCS flag to the value of the PcsCfg parameter.*
- LOCAL_INLINE void [Spi_Reg_SetDataWidth](#) (SPI_Type *Base, [Spi_TransferWidthType](#) DataWidth)
- Set SPI transfer data width.*
- LOCAL_INLINE void [Spi_Reg_WriteData](#) (SPI_Type *Base, uint32 Data)
- Writes data into the TX data register.*
- LOCAL_INLINE uint32 [Spi_Reg_ReadData](#) (const SPI_Type *Base)
- Reads data from the data register.*
- LOCAL_INLINE uint32 [Spi_Reg_GetDataAddress](#) (const SPI_Type *Base)
- Get address of the data register.*
- LOCAL_INLINE void [Spi_Reg_SetMasterNoOverflowMode](#) (SPI_Type *Base, boolean Cmd)
- Config Master no overflow mode.*
- LOCAL_INLINE void [Spi_Reg_SetTxOnly](#) (SPI_Type *Base, boolean Cmd)
- Enable/Disable TX Only mode.*
- LOCAL_INLINE boolean [Spi_Reg_GetTxOnly](#) (const SPI_Type *Base)
- Get TX Only mode.*

- LOCAL_INLINE void [Spi_Reg_SetRxOnly](#) (SPI_Type *Base, boolean Cmd)
Enable/Disable RX Only mode.
- LOCAL_INLINE boolean [Spi_Reg_GetRxOnly](#) (const SPI_Type *Base)
Get RX Only mode.
- LOCAL_INLINE void [Spi_Reg_SetHreqPolarity](#) (SPI_Type *Base, [Spi_HreqPolarityType](#) HreqPolarity)
Set SPI host request polarity.
- LOCAL_INLINE void [Spi_Reg_SetHreq](#) (SPI_Type *Base, boolean Cmd)
Enable or disable SPI host request.
- LOCAL_INLINE void [Spi_Reg_SetDebug](#) (SPI_Type *Base, boolean Cmd)
Enable or disable SPI in debug mode.
- LOCAL_INLINE void [SPI_Reg_SetMasterModeFaultDetectCs0](#) (SPI_Type *base, boolean enable)
Enable or disable SPI Master Mode Fault Detect in cs0.
- LOCAL_INLINE void [SPI_Reg_SetMasterModeFaultDetectCs1](#) (SPI_Type *base, boolean enable)
Enable or disable SPI Master Mode Fault Detect in cs1.
- LOCAL_INLINE void [SPI_Reg_SetMasterModeFaultDetectCs2](#) (SPI_Type *base, boolean enable)
Enable or disable SPI Master Mode Fault Detect in cs2.
- LOCAL_INLINE void [SPI_Reg_SetMasterModeFaultDetectCs3](#) (SPI_Type *base, boolean enable)
Enable or disable SPI Master Mode Fault Detect in cs3.
- LOCAL_INLINE void [Spi_Reg_SetMatchData](#) (SPI_Type *Base, uint32 Data)
Set SPI match data.
- LOCAL_INLINE void [Spi_Reg_SetDMIE](#) (SPI_Type *Base, boolean Cmd)
Set SPI match data.
- LOCAL_INLINE void [Spi_Reg_ClearRDMF](#) (SPI_Type *Base)
Clear current SPI RDMF flag.
- LOCAL_INLINE void [Spi_Reg_SetROTRIG](#) (SPI_Type *Base, boolean Cmd)
Toggle SPI RX Only.

4.2.1 Detailed Description

Spi driver register header file.

4.2.2 Function Documentation

4.2.2.1 Spi_Reg_ClearRDMF()

```
LOCAL_INLINE void Spi_Reg_ClearRDMF (
    SPI_Type * Base )
```

Clear current SPI RDMF flag.

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 808 of file AC784xx_Spi_Reg.h.

4.2.2.2 Spi_Reg_ClearRxOF()

```
LOCAL_INLINE void Spi_Reg_ClearRxOF (
    SPI_Type * Base )
```

Clear current SPI Rx overflow flag.

Note

Function ID: DES_SPI_API_312

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 312 of file AC784xx_Spi_Reg.h.

4.2.2.3 SPI_Reg_ClearStatusFlag()

```
LOCAL_INLINE void SPI_Reg_ClearStatusFlag (
    SPI_Type * Base )
```

Clear SPI Status Flag.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

none

Definition at line 286 of file AC784xx_Spi_Reg.h.

4.2.2.4 Spi_Reg_ClearTxUF()

```
LOCAL_INLINE void Spi_Reg_ClearTxUF (
    SPI_Type * Base )
```

Clear current SPI Tx underflow flag.

Note

Function ID: DES_SPI_API_311

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 299 of file AC784xx_Spi_Reg.h.

4.2.2.5 Spi_Reg_GetDataAddress()

```
LOCAL_INLINE uint32 Spi_Reg_GetDataAddress (
    const SPI_Type * Base )
```

Get address of the data register.

Note

Function ID: DES_SPI_API_330

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

The address of the data register.

Definition at line 604 of file AC784xx_Spi_Reg.h.

4.2.2.6 Spi_Reg_GetIntMode()

```
LOCAL_INLINE boolean Spi_Reg_GetIntMode (
    const SPI_Type * Base,
    Spi_InterruptType InterruptSrc )
```

Returns if the SPI interrupt request is enabled or disabled.

Note

Function ID: DES_SPI_API_314

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>InterruptSrc</i>	The interrupt source.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Return the interrupt source enable status.

- TRUE: the interrupt source is enabled.
- FALSE: the interrupt source is disabled.

Definition at line 350 of file AC784xx_Spi_Reg.h.

4.2.2.7 Spi_Reg_GetRxOnly()

```
LOCAL_INLINE boolean Spi_Reg_GetRxOnly (
    const SPI_Type * Base )
```

Get RX Only mode.

Note

Function ID: DES_SPI_API_335

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Returns SPI RX only mode is enabled/disabled.

- TRUE: SPI RX only mode is enabled.
- FALSE: SPI RX only mode is disabled.

Definition at line 679 of file AC784xx_Spi_Reg.h.

4.2.2.8 Spi_Reg_GetSckHighLow()

```
LOCAL_INLINE void Spi_Reg_GetSckHighLow (
    const SPI_Type * Base,
    uint8 * SckLow,
    uint8 * SckHigh )
```

Get SPI SckLow and SckHigh.

Parameters

in	<i>Base</i>	Module base pointer of type SPI_Type.
in	<i>SckLow</i>	SPI SCK Low count, low_time = (SckLow + 1) * Bus_Period.
in	<i>SckHigh</i>	SPI SCK High count, high_time = (SckHigh + 1) * Bus_Period.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 167 of file AC784xx_Spi_Reg.h.

4.2.2.9 Spi_Reg_GetStatusFlag()

```
LOCAL_INLINE boolean Spi_Reg_GetStatusFlag (
    const SPI_Type * Base,
    Spi_StatusFlagType StatusFlag )
```

Gets the SPI status flag enable.

Note

Function ID: DES_SPI_API_310

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>StatusFlag</i>	Select which status will be return status.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

State of the status flag.

- TRUE: the status flag is "1".
- FALSE: the status flag is "0".

Definition at line 275 of file AC784xx_Spi_Reg.h.

4.2.2.10 Spi_Reg_GetTxOnly()

```
LOCAL_INLINE boolean Spi_Reg_GetTxOnly (
    const SPI_Type * Base )
```

Get TX Only mode.

Note

Function ID: DES_SPI_API_333

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Returns SPI TX only mode is enabled/disabled.

- TRUE: SPI TX only mode is enabled.
- FALSE: SPI TX only mode is disabled.

Definition at line 649 of file AC784xx_Spi_Reg.h.

4.2.2.11 SPI_Reg_IsCsContinuousMode()

```
LOCAL_INLINE boolean SPI_Reg_IsCsContinuousMode (
    const SPI_Type * base )
```

Returns whether the SPI module is in Cs Continuous mode.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

Returns the status of SPI Continuous or DisContinuous mode -true: SPI work in Continuous mode -false: SPI work in DisContinuous mode

Definition at line 256 of file AC784xx_Spi_Reg.h.

4.2.2.12 Spi_Reg_IsMaster()

```
LOCAL_INLINE boolean Spi_Reg_IsMaster (
    const SPI_Type * Base )
```

Returns whether the SPI module is in master mode.

Note

Function ID: DES_SPI_API_309

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

Returns the status of SPI master or slave mode.

- TRUE: SPI work in master mode.
- FALSE: SPI work in slave mode.

Definition at line 243 of file AC784xx_Spi_Reg.h.

4.2.2.13 SPI_Reg_IsModuleEnabled()

```
LOCAL_INLINE boolean SPI_Reg_IsModuleEnabled (
    const SPI_Type * base )
```

Check if SPI module is enabled.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

SPI module is enabled or not -true: SPI module is enabled -false: SPI module is disabled

Definition at line 212 of file AC784xx_Spi_Reg.h.

4.2.2.14 Spi_Reg_ReadData()

```
LOCAL_INLINE uint32 Spi_Reg_ReadData (
    const SPI_Type * Base )
```

Reads data from the data register.

Note

Function ID: DES_SPI_API_329

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

The RX data read from the data register.

Definition at line 591 of file AC784xx_Spi_Reg.h.

4.2.2.15 Spi_Reg_ReleaseCS()

```
LOCAL_INLINE void Spi_Reg_ReleaseCS (
    SPI_Type * Base )
```

Release SPI CS, which will be changed to inactive enable.

Note

Function ID: DES_SPI_API_325

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 529 of file AC784xx_Spi_Reg.h.

4.2.2.16 Spi_Reg_SetBaudRate()

```
LOCAL_INLINE void Spi_Reg_SetBaudRate (
    SPI_Type * Base,
    uint8 SckLow,
    uint8 SckHigh )
```

Set SPI CS baudrate, 1/(low_time + high_time).

Note

Function ID: DES_SPI_API_305

Parameters

in	<i>Base</i>	Module base pointer of type SPI_Type.
in	<i>SckLow</i>	SPI SCK Low count, low_time = (SckLow + 1) * Bus_Period.
in	<i>SckHigh</i>	SPI SCK High count, high_time = (SckHigh + 1) * Bus_Period.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 152 of file AC784xx_Spi_Reg.h.

4.2.2.17 Spi_Reg_SetContinuousCS()

```
LOCAL_INLINE void Spi_Reg_SetContinuousCS (
    SPI_Type * Base,
    boolean Cmd )
```

Set SPI CS continuous or disContinuous mode.

Note

Function ID: DES_SPI_API_322

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Set the SPI CS output continuous mode. <ul style="list-style-type: none">• TRUE: CS continuous mode.• FALSE: CS discontinuous mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 481 of file AC784xx_Spi_Reg.h.

4.2.2.18 Spi_Reg_SetCPHA()

```
LOCAL_INLINE void Spi_Reg_SetCPHA (
    SPI_Type * Base,
    Spi_CphaType ClkPhase )
```

Set SPI CPHA.

Note

Function ID: DES_SPI_API_318

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>ClkPhase</i>	SPI clock CPHA <ul style="list-style-type: none"> • SPI_CPHA\leftarrow _0 • SPI_CPHA\leftarrow _1
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 410 of file AC784xx_Spi_Reg.h.

4.2.2.19 Spi_Reg_SetCPOL()

```
LOCAL_INLINE void Spi_Reg_SetCPOL (
    SPI_Type * Base,
    Spi_CpolType ClkPolarity )
```

Set SPI CPOL.

Note

Function ID: DES_SPI_API_319

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>ClkPolarity</i>	SPI cpol select. <ul style="list-style-type: none">• SPI_CPOL_LOW• SPI_CPOL_HIGH
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 426 of file AC784xx_Spi_Reg.h.

4.2.2.20 Spi_Reg_SetCSHold()

```
LOCAL_INLINE void Spi_Reg_SetCSHold (  
    SPI_Type * Base,  
    uint8 Delay )
```

Set CS_HOLD time.

Note

Function ID: DES_SPI_API_302

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Delay</i>	The 8-bit delay value 0x00 to 0xFF (255).
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 101 of file AC784xx_Spi_Reg.h.

4.2.2.21 Spi_Reg_SetCSIdle()

```
LOCAL_INLINE void Spi_Reg_SetCSIdle (  
    SPI_Type * Base,  
    uint8 Delay )
```

Set CS_IDLE time.

Note

Function ID: DES_SPI_API_303

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Delay</i>	The 8-bit delay value 0x00 to 0xFF (255).
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 115 of file AC784xx_Spi_Reg.h.

4.2.2.22 Spi_Reg_SetCSOE()

```
LOCAL_INLINE void Spi_Reg_SetCSOE (  
    SPI_Type * Base,  
    boolean Cmd )
```

Set SPI CS output enable.

Note

Function ID: DES_SPI_API_323

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Set the SPI CS hardware output. <ul style="list-style-type: none">• TRUE: enable the CS hardware output(CS control by hardware).• FALSE: disable the CS hardware output(CS control by software).
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 498 of file AC784xx_Spi_Reg.h.

4.2.2.23 Spi_Reg_SetCSSetup()

```
LOCAL_INLINE void Spi_Reg_SetCSSetup (
    SPI_Type * Base,
    uint8 Delay )
```

Set CS_SETUP time.

Note

Function ID: DES_SPI_API_301

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Delay</i>	The 8-bit delay value 0x00 to 0xFF (255).
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 87 of file AC784xx_Spi_Reg.h.

4.2.2.24 Spi_Reg_SetDataWidth()

```
LOCAL_INLINE void Spi_Reg_SetDataWidth (
    SPI_Type * Base,
    Spi_TransferWidthType DataWidth )
```

Set SPI transfer data width.

Note

Function ID: DES_SPI_API_327

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>DataWidth</i>	transfer data width. <ul style="list-style-type: none"> • SPI_DATA_WIDTH_1↔ BIT • SPI_DATA_WIDTH_2↔ BIT • SPI_DATA_WIDTH_4↔ BIT
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 564 of file AC784xx_Spi_Reg.h.

4.2.2.25 Spi_Reg_SetDebug()

```
LOCAL_INLINE void Spi_Reg_SetDebug (
    SPI_Type * Base,
    boolean Cmd )
```

Enable or disable SPI in debug mode.

Note

Function ID: DES_SPI_API_338

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable or disable SPI in debug mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 724 of file AC784xx_Spi_Reg.h.

4.2.2.26 Spi_Reg_SetDMIE()

```
LOCAL_INLINE void Spi_Reg_SetDMIE (
    SPI_Type * Base,
    boolean Cmd )
```

Set SPI match data.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

none

Definition at line 795 of file AC784xx_Spi_Reg.h.

4.2.2.27 Spi_Reg_SetEnable()

```
LOCAL_INLINE void Spi_Reg_SetEnable (
    SPI_Type * Base,
    boolean Cmd )
```

Enable/Disable SPI module.

Note

Function ID: DES_SPI_API_307

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable/disable SPI module.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 198 of file AC784xx_Spi_Reg.h.

4.2.2.28 Spi_Reg_SetFrameSize()

```
LOCAL_INLINE void Spi_Reg_SetFrameSize (
    SPI_Type * Base,
    uint8 FrameSize )
```

Set SPI frame size(4~32 bits per frame).

Note

Function ID: DES_SPI_API_306

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>FrameSize</i>	SPI frame size, support 4~32 bits.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 184 of file AC784xx_Spi_Reg.h.

4.2.2.29 Spi_Reg_SetHreq()

```
LOCAL_INLINE void Spi_Reg_SetHreq (
    SPI_Type * Base,
    boolean Cmd )
```

Enable or disable SPI host request.

Note

Function ID: DES_SPI_API_337

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable or disable SPI host request.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 709 of file AC784xx_Spi_Reg.h.

4.2.2.30 Spi_Reg_SetHreqPolarity()

```
LOCAL_INLINE void Spi_Reg_SetHreqPolarity (
    SPI_Type * Base,
    Spi_HreqPolarityType HreqPolarity )
```

Set SPI host request polarity.

Note

Function ID: DES_SPI_API_336

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>HreqPolarity</i>	Host request polarity. <ul style="list-style-type: none">• SPI_HREQ_POLARITY_LOW
Generated by Doxygen		<ul style="list-style-type: none">• SPI_HREQ_POLARITY_HIGH
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 695 of file AC784xx_Spi_Reg.h.

4.2.2.31 Spi_Reg_SetIntMode()

```
LOCAL_INLINE void Spi_Reg_SetIntMode (
    SPI_Type * Base,
    Spi_InterruptType InterruptSrc,
    boolean Cmd )
```

Configures the SPI interrupts.

Note

Function ID: DES_SPI_API_313

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>InterruptSrc</i>	The interrupt source, which will be enabled.
in	<i>Cmd</i>	Enable or disable the interrupt source.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 327 of file AC784xx_Spi_Reg.h.

4.2.2.32 SPI_Reg_SetMasterModeFaultDetectCs0()

```
LOCAL_INLINE void SPI_Reg_SetMasterModeFaultDetectCs0 (
    SPI_Type * base,
    boolean enable )
```

Enable or disable SPI Master Mode Fault Detect in cs0.

Parameters

in	<i>base</i>	SPI base pointer
in	<i>enable</i>	Enable or disable SPI Master Mode Fault Detect

Returns

none

Definition at line 737 of file AC784xx_Spi_Reg.h.

4.2.2.33 SPI_Reg_SetMasterModeFaultDetectCs1()

```
LOCAL_INLINE void SPI_Reg_SetMasterModeFaultDetectCs1 (
    SPI_Type * base,
    boolean enable )
```

Enable or disable SPI Master Mode Fault Detect in cs1.

Parameters

in	<i>base</i>	SPI base pointer
in	<i>enable</i>	Enable or disable SPI Master Mode Fault Detect

Returns

none

Definition at line 749 of file AC784xx_Spi_Reg.h.

4.2.2.34 SPI_Reg_SetMasterModeFaultDetectCs2()

```
LOCAL_INLINE void SPI_Reg_SetMasterModeFaultDetectCs2 (
    SPI_Type * base,
    boolean enable )
```

Enable or disable SPI Master Mode Fault Detect in cs2.

Parameters

in	<i>base</i>	SPI base pointer
in	<i>enable</i>	Enable or disable SPI Master Mode Fault Detect

Returns

none

Definition at line 761 of file AC784xx_Spi_Reg.h.

4.2.2.35 SPI_Reg_SetMasterModeFaultDetectCs3()

```
LOCAL_INLINE void SPI_Reg_SetMasterModeFaultDetectCs3 (
    SPI_Type * base,
    boolean enable )
```

Enable or disable SPI Master Mode Fault Detect in cs3.

Parameters

in	<i>base</i>	SPI base pointer
in	<i>enable</i>	Enable or disable SPI Master Mode Fault Detect

Returns

none

Definition at line 773 of file AC784xx_Spi_Reg.h.

4.2.2.36 Spi_Reg_SetMasterNoOverflowMode()

```
LOCAL_INLINE void Spi_Reg_SetMasterNoOverflowMode (
    SPI_Type * Base,
    boolean Cmd )
```

Config Master no overflow mode.

Note

Function ID: DES_SPI_API_331

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable or disable SPI no over flow mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 618 of file AC784xx_Spi_Reg.h.

4.2.2.37 Spi_Reg_SetMasterSlaveMode()

```
LOCAL_INLINE void Spi_Reg_SetMasterSlaveMode (
    SPI_Type * Base,
    Spi_ModeType Mode )
```

Set SPI master or slave mode.

Note

Function ID: DES_SPI_API_308

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Mode</i>	Set SPI master mode or not. <ul style="list-style-type: none">• SPI_MASTER: SPI master mode.• SPI_SLAVE: SPI slave mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 228 of file AC784xx_Spi_Reg.h.

4.2.2.38 Spi_Reg_SetMatchData()

```
LOCAL_INLINE void Spi_Reg_SetMatchData (  
    SPI_Type * Base,  
    uint32 Data )
```

Set SPI match data.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

none

Definition at line 784 of file AC784xx_Spi_Reg.h.

4.2.2.39 Spi_Reg_SetModeFault()

```
LOCAL_INLINE void Spi_Reg_SetModeFault (  
    SPI_Type * Base,  
    boolean Cmd )
```

Enable/Disable SPI CS mode fault detect function.

Note

Function ID: DES_SPI_API_324

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Set the SPI CS mode fault detect function. <ul style="list-style-type: none">• TRUE: enable the master mode fault detect function.• FALSE: disable the master mode fault detect function.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 515 of file AC784xx_Spi_Reg.h.

4.2.2.40 Spi_Reg_SetPcsCfg()

```
LOCAL_INLINE void Spi_Reg_SetPcsCfg (  
    SPI_Type * Base,  
    Spi_PcsType PcsCfg )
```

Sets the PCS flag to the value of the PcsCfg parameter.

Note

Function ID: DES_SPI_API_326

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>PcsCfg</i>	Select which CS is assert. <ul style="list-style-type: none">• SPI_PCS_0• SPI_PCS_1• SPI_PCS_2• SPI_PCS_3
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 547 of file AC784xx_Spi_Reg.h.

4.2.2.41 Spi_Reg_SetPcsPolarity()

```
LOCAL_INLINE void Spi_Reg_SetPcsPolarity (
    SPI_Type * Base,
    Spi_PcsType PcsCfg,
    Spi_PcsPolarityType PcsPolarity )
```

Configures the SPI PCS polarity.

Note

Function ID: DES_SPI_API_304

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>PcsCfg</i>	Select which PCS to configured. <ul style="list-style-type: none">• SPI_PCS_0• SPI_PCS_1• SPI_PCS_2• SPI_PCS_3
in	<i>PcsPolarity</i>	Set PCS as active high or low <ul style="list-style-type: none">• SPI_ACTIVE_LOW• SPI_ACTIVE_HIGH
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 136 of file AC784xx_Spi_Reg.h.

4.2.2.42 Spi_Reg_SetPinConfigMode()

```
LOCAL_INLINE void Spi_Reg_SetPinConfigMode (
    SPI_Type * Base,
    Spi_PinCfgType PinCfg )
```

Configures the SPI SOUT/SIN pin configuration mode.

Note

Function ID: DES_SPI_API_317

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>PinCfg</i>	Select configuration for the SOUT/SIN pins.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 394 of file AC784xx_Spi_Reg.h.

4.2.2.43 Spi_Reg_SetROTRIG()

```
LOCAL_INLINE void Spi_Reg_SetROTRIG (
    SPI_Type * Base,
    boolean Cmd )
```

Toggle SPI RX Only.

Parameters

in	<i>base</i>	SPI base pointer
----	-------------	------------------

Returns

none

Definition at line 819 of file AC784xx_Spi_Reg.h.

4.2.2.44 Spi_Reg_SetRxDmaCmd()

```
LOCAL_INLINE void Spi_Reg_SetRxDmaCmd (
    SPI_Type * Base,
    boolean Cmd )
```

Enable/disable the SPI Receive Data DMA request.

Note

Function ID: DES_SPI_API_316

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable or disable the RX DMA request.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 379 of file AC784xx_Spi_Reg.h.

4.2.2.45 Spi_Reg_SetRxMSB()

```
LOCAL_INLINE void Spi_Reg_SetRxMSB (
    SPI_Type * Base,
    boolean Cmd )
```

Set SPI RX MSB first.

Note

Function ID: DES_SPI_API_320

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Set the SPI RX MSB first. <ul style="list-style-type: none">• TRUE: RX MSB• FALSE: RX LSB
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 442 of file AC784xx_Spi_Reg.h.

4.2.2.46 Spi_Reg_SetRxOnly()

```
LOCAL_INLINE void Spi_Reg_SetRxOnly (
    SPI_Type * Base,
    boolean Cmd )
```

Enable/Disable RX Only mode.

Note

Function ID: DES_SPI_API_334

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	enable/disable SPI RX only mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 663 of file AC784xx_Spi_Reg.h.

4.2.2.47 Spi_Reg_SetTxDmaCmd()

```
LOCAL_INLINE void Spi_Reg_SetTxDmaCmd (
    SPI_Type * Base,
    boolean Cmd )
```

Enable/disable the SPI Transmit Data DMA request.

Note

Function ID: DES_SPI_API_315

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Enable or disable the TX DMA request.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 364 of file AC784xx_Spi_Reg.h.

4.2.2.48 Spi_Reg_SetTxMSB()

```
LOCAL_INLINE void Spi_Reg_SetTxMSB (
    SPI_Type * Base,
    boolean Cmd )
```

Set SPI TX MSB first.

Note

Function ID: DES_SPI_API_321

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	Set the SPI TX MSB first. • TRUE: TX MSB • FALSE: TX LSB
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 459 of file AC784xx_Spi_Reg.h.

4.2.2.49 Spi_Reg_SetTxOnly()

```
LOCAL_INLINE void Spi_Reg_SetTxOnly (
    SPI_Type * Base,
    boolean Cmd )
```

Enable/Disable TX Only mode.

Note

Function ID: DES_SPI_API_332

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Cmd</i>	enable/disable SPI TX only mode.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 633 of file AC784xx_Spi_Reg.h.

4.2.2.50 Spi_Reg_SoftwareReset()

```
LOCAL_INLINE void Spi_Reg_SoftwareReset (
    SPI_Type * Base )
```

SPI Module softreset. And it only reset master engine/buffer/flag logic, slave buffer/flag logic, the configuration of control registers(CFG0/ CFG1/CFG2/CMD) will not be reset.

Note

Function ID: DES_SPI_API_300

Parameters

in	<i>Base</i>	SPI base pointer.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 72 of file AC784xx_Spi_Reg.h.

4.2.2.51 Spi_Reg_WriteData()

```
LOCAL_INLINE void Spi_Reg_WriteData (
    SPI_Type * Base,
    uint32 Data )
```

Writes data into the TX data register.

Note

Function ID: DES_SPI_API_328

Parameters

in	<i>Base</i>	SPI base pointer.
in	<i>Data</i>	The data word to be sent.
in, out	<i>None</i>	
out	<i>None</i>	

Returns

None

Definition at line 578 of file AC784xx_Spi_Reg.h.

4.3 Spi_Hal.c File Reference

Spi driver hal api source file.

```
#include "AC784xx_Spi_Reg.h"
#include "Spi_Hal.h"
#include "Ckgen_Hal.h"
```

```
#include "Rcm_Hal.h"
#include "Dma_Hal.h"
#include "Core_Hal.h"
#include "OsIf_Time.h"
```

Macros

- `#define SPI_TIMEOUT_FRAME ((uint32)1320U)`
The timeout used to wait for TX/RX transmission to complete a frame. For OsIf counter, it is recommended to use 1320(us) which is enough for 2 frames in fifo at 32bit 50k bps. And in loop mode, there are some rough empirical value by counting test at 120MHz sys_clock: 0xFFFF - 170us, 0x1FFF - 341us, 0x4FFF - 853us, 0x5FFF - 1.024ms, 0xFFFF - 2.73ms.
- `#define SPI_1BYTE_MSK (0xFFU)`
SPI DATA register access mask and pos when it converts to different fram size.
- `#define SPI_2BYTES_MSK (0xFFFFU)`
- `#define SPI_BIT8_POS (8U)`
- `#define SPI_BIT16_POS (16U)`
- `#define SPI_BIT24_POS (24U)`

Enumerations

- enum `Spi_TransceiveType` { `SPI_HAL_TX` = 0x00U, `SPI_HAL_RX` = 0x01U, `SPI_HAL_TXRX` = 0x02U }
Type of SPI transfer(tx or rx)

Functions

- void `SPI0_IRQHandler` (void)
This function is the implementation of SPI0 handler named in startup code. It passes the instance to the shared SPI IRQ handler.
- void `SPI1_IRQHandler` (void)
This function is the implementation of SPI1 handler named in startup code. It passes the instance to the shared SPI IRQ handler.
- void `SPI2_IRQHandler` (void)
This function is the implementation of SPI2 handler named in startup code. It passes the instance to the shared SPI IRQ handler.
- void `Spi_Hal_ConfigureBus` (uint8 Instance, const `Spi_HalConfigType` *ConfigPtr)
This function initializes a SPI instance.
- void `Spi_Hal_ConfigureFrame` (uint8 Instance, uint8 FrmSize, boolean MsbFirst)
This function configures the SPI frame parameters.
- void `Spi_Hal_Init` (uint8 Instance, const `Spi_HalConfigType` *ConfigPtr)
This function initializes a SPI instance.
- void `Spi_Hal_DeInit` (uint8 Instance)
This function de-initializes a SPI instance.
- void `Spi_Hal_GetDefaultConfig` (`Spi_HalConfigType` *ConfigPtr)
This function gets default configuration for SPI hal driver.
- Hal_StatusType `Spi_Hal_TransceivePoll` (uint8 Instance, const uint8 *TxBuffPtr, uint8 *RxBuffPtr, uint16 ByteCount, uint32 Timeout)
This function transfers by polling mode, which will not return untill timeout or transfer finish.
- Hal_StatusType `Spi_Hal_TransceiveInt` (uint8 Instance, const uint8 *TxBuffPtr, uint8 *RxBuffPtr, uint16 ByteCount)
SPI transmission, reception by interrupt.
- Hal_StatusType `Spi_Hal_TransceiveDma` (uint8 Instance, const uint8 *TxBuffPtr, const uint8 *RxBuffPtr, uint16 ByteCount)
SPI transmission, reception by DMA.

- uint16 [Spi_Hal_SlaveGetReceiveLen](#) (uint8 Instance, uint8 **ReceiveBuffer)
This function get spi receive data when in rx buffer.
- Hal_StatusType [Spi_Hal_SlaveReceiveNolimitLen](#) (uint8 Instance, uint8 *ReceiveBuffer1, uint8 *ReceiveBuffer2, uint16 MaxreceiveByteCount)
SPI using interrupt receive by NolimitLen mode, which will return after start transfer. The user needs to check whether the receiveBuffer has valid data the Spi_Hal_GetTransceiveStatus function.
- Hal_StatusType [Spi_Hal_AbortTransceive](#) (uint8 Instance)
This function stop SPI Transceive.
- [Spi_TransceiveStatusType Spi_Hal_GetTransceiveStatus](#) (uint8 Instance)
This function get SPI Transceive Status.
- void [Spi_Hal_SoftwareReset](#) (uint8 Instance)
This function reset the SPI Status.
- void [Spi_Hal_SetPinMode](#) (uint8 Instance, [Spi_PinCfgType](#) PinCfg)
This function set the SPI pin mode.
- void [Spi_Hal_SetCsPin](#) (uint8 Instance, [Spi_PcsType](#) Pcs, [Spi_PcsPolarityType](#) Polarity)
This function select the SPI cs pin.
- Hal_StatusType [Spi_Hal_SetBaudRate](#) (uint8 Instance, uint32 BaudRate)
This function set the SPI BaudRate.
- uint32 [Spi_Hal_GetBaudRate](#) (uint8 Instance)
This function set the SPI BaudRate.
- void [Spi_Hal_SetMatchData](#) (uint8 Instance, uint32 Data, boolean Enable)
This function set the SPI Match Data.
- SPI_Type * [Spi_Hal_GetBase](#) (uint8 Instance)
Get the SPI base.
- void [Spi_Hal_SetFrameSize](#) (uint8 Instance, uint8 Framesize)
This function set the SPI FrameSize.
- boolean [Spi_Hal_CheckBusy](#) (uint8 Instance)
This function checks whether the status of SPI hal driver is busy.

4.3.1 Detailed Description

Spi driver hal api source file.

4.3.2 Macro Definition Documentation

4.3.2.1 SPI_1BYTE_MSK

```
#define SPI_1BYTE_MSK (0xFFU)
```

SPI DATA register access mask and pos when it converts to different fram size.

Definition at line 60 of file Spi_Hal.c.

4.3.2.2 SPI_2BYTES_MSK

```
#define SPI_2BYTES_MSK (0xFFFFU)
```

Definition at line 61 of file Spi_Hal.c.

4.3.2.3 SPI_BIT16_POS

```
#define SPI_BIT16_POS (16U)
```

Definition at line 63 of file Spi_Hal.c.

4.3.2.4 SPI_BIT24_POS

```
#define SPI_BIT24_POS (24U)
```

Definition at line 64 of file Spi_Hal.c.

4.3.2.5 SPI_BIT8_POS

```
#define SPI_BIT8_POS (8U)
```

Definition at line 62 of file Spi_Hal.c.

4.3.2.6 SPI_TIMEOUT_FRAME

```
#define SPI_TIMEOUT_FRAME ((uint32)1320U)
```

The timeout used to wait for TX/RX transmission to complete a frame. For Oslf counter, it is recommended to use 1320(us) which is enough for 2 frames in fifo at 32bit 50k bps. And in loop mode, there are some rough empirical value by counting test at 120MHz sys_clock: 0xFFFF - 170us, 0x1FFF - 341us, 0x4FFF - 853us, 0x5FFF - 1.024ms, 0xFFFF - 2.73ms.

Definition at line 57 of file Spi_Hal.c.

4.3.3 Enumeration Type Documentation

4.3.3.1 Spi_TransceiveType

```
enum Spi_TransceiveType
```

Type of SPI transfer(tx or rx)

Enumerator

SPI_HAL_TX	
SPI_HAL_RX	
SPI_HAL_TXRX	

Definition at line 70 of file Spi_Hal.c.

4.3.4 Function Documentation**4.3.4.1 SPI0_IRQHandler()**

```
void SPI0_IRQHandler (  
    void )
```

This function is the implementation of SPI0 handler named in startup code. It passes the instance to the shared SPI IRQ handler.

Note

Function ID: DES_SPI_API_221

Returns

void

Definition at line 1864 of file Spi_Hal.c.

4.3.4.2 SPI1_IRQHandler()

```
void SPI1_IRQHandler (  
    void )
```

This function is the implementation of SPI1 handler named in startup code. It passes the instance to the shared SPI IRQ handler.

Note

Function ID: DES_SPI_API_222

Returns

void

Definition at line 1875 of file Spi_Hal.c.

4.3.4.3 SPI2_IRQHandler()

```
void SPI2_IRQHandler (
    void )
```

This function is the implementation of SPI2 handler named in startup code. It passes the instance to the shared SPI IRQ handler.

Note

Function ID: DES_SPI_API_223

Returns

void

Definition at line 1886 of file Spi_Hal.c.

4.3.4.4 Spi_Hal_AbortTransceive()

```
Hal_StatusType Spi_Hal_AbortTransceive (
    uint8 Instance )
```

This function stop SPI Transceive.

Note

Function ID: DES_SPI_API_235

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

Hal_StatusType

Definition at line 1547 of file Spi_Hal.c.

4.3.4.5 Spi_Hal_CheckBusy()

```
boolean Spi_Hal_CheckBusy (
    uint8 Instance )
```

This function checks whether the status of SPI hal driver is busy.

Note

Function ID: DES_SPI_API_211

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

boolean

- TRUE: The SPI hal driver is busy.
- FALSE: The SPI hal driver is not busy.

Definition at line 1844 of file Spi_Hal.c.

4.3.4.6 Spi_Hal_ConfigureBus()

```
void Spi_Hal_ConfigureBus (
    uint8 Instance,
    const Spi_HalConfigType * ConfigPtr )
```

This function initializes a SPI instance.

Note

Function ID: DES_SPI_API_232

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ConfigPtr</i>	The data structure containing information about a device on the SPI bus.

Returns

void

Definition at line 1039 of file Spi_Hal.c.

4.3.4.7 Spi_Hal_ConfigureFrame()

```
void Spi_Hal_ConfigureFrame (
    uint8 Instance,
    uint8 FrmSize,
    boolean MsbFirst )
```

This function configures the SPI frame parameters.

Note

Function ID: DES_SPI_API_204

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>FrmSize</i>	Number of bits per frame, support 4~32 bits.
in	<i>MsbFirst</i>	Option to transmit MSB first.

Returns

void

Definition at line 1097 of file Spi_Hal.c.

4.3.4.8 Spi_Hal_DeInit()

```
void Spi_Hal_DeInit (
    uint8 Instance )
```

This function de-initializes a SPI instance.

Note

Function ID: DES_SPI_API_201

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

void

Definition at line 1165 of file Spi_Hal.c.

4.3.4.9 Spi_Hal_GetBase()

```
SPI_Type* Spi_Hal_GetBase (
    uint8 Instance )
```

Get the SPI base.

Note

Function ID: DES_SPI_API_231

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

SPI_Type: SPI Base Addr.

Definition at line 1799 of file Spi_Hal.c.

4.3.4.10 Spi_Hal_GetBaudRate()

```
uint32 Spi_Hal_GetBaudRate (
    uint8 Instance )
```

This function set the SPI BaudRate.

This function get the SPI BaudRate.

Note

Function ID: DES_SPI_API_240

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

uint32:SPI BaudRate

Definition at line 1744 of file Spi_Hal.c.

4.3.4.11 Spi_Hal_GetDefaultConfig()

```
void Spi_Hal_GetDefaultConfig (
    Spi_HalConfigType * ConfigPtr )
```

This function gets default configuration for SPI hal driver.

Note

Function ID: DES_SPI_API_202

Parameters

out	<i>ConfigPtr</i>	Pointer to configuration structure which is filled with default configuration.
-----	------------------	--

Returns

void

Definition at line 1189 of file Spi_Hal.c.

4.3.4.12 Spi_Hal_GetTransceiveStatus()

```
Spi_TransceiveStatusType Spi_Hal_GetTransceiveStatus (
    uint8 Instance )
```

This function get SPI Transceive Status.

Note

Function ID: DES_SPI_API_236

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

Spi_TransceiveStatusType:the Transceive Status.

Definition at line 1587 of file Spi_Hal.c.

4.3.4.13 Spi_Hal_Init()

```
void Spi_Hal_Init (
    uint8 Instance,
    const Spi_HalConfigType * ConfigPtr )
```

This function initializes a SPI instance.

Note

Function ID: DES_SPI_API_200

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ConfigPtr</i>	The data structure containing information about a device on the SPI bus.

Returns

None

Definition at line 1124 of file Spi_Hal.c.

4.3.4.14 Spi_Hal_SetBaudRate()

```
Hal_StatusType Spi_Hal_SetBaudRate (
    uint8 Instance,
    uint32 BaudRate )
```

This function set the SPI BaudRate.

Note

Function ID: DES_SPI_API_230

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>BaudRate</i>	To set the BaudRate value.

Returns

Hal_StatusType

Definition at line 1699 of file Spi_Hal.c.

4.3.4.15 Spi_Hal_SetCsPin()

```
void Spi_Hal_SetCsPin (
    uint8 Instance,
    Spi_PcsType Pcs,
    Spi_PcsPolarityType Polarity )
```

This function select the SPI cs pin.

Note

Function ID: DES_SPI_API_239

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Pcs</i>	SPI CS select.
in	<i>Polarity</i>	CS Polarity.

Returns

void

Definition at line 1671 of file Spi_Hal.c.

4.3.4.16 Spi_Hal_SetFrameSize()

```
void Spi_Hal_SetFrameSize (
    uint8 Instance,
    uint8 Framesize )
```

This function set the SPI FrameSize.

Note

Function ID: DES_SPI_API_242

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Framesize</i>	Number of bits per frame, support 4~32 bits.

Returns

void

Definition at line 1818 of file Spi_Hal.c.

4.3.4.17 Spi_Hal_SetMatchData()

```
void Spi_Hal_SetMatchData (
    uint8 Instance,
    uint32 Data,
    boolean Enable )
```

This function set the SPI Match Data.

Note

Function ID: DES_SPI_API_241

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Data</i>	Data to Match.
in	<i>Enable</i>	Enable or Disable Data Match.

Returns

void

Definition at line 1777 of file Spi_Hal.c.

4.3.4.18 Spi_Hal_SetPinMode()

```
void Spi_Hal_SetPinMode (
    uint8 Instance,
    Spi_PinCfgType PinCfg )
```

This function set the SPI pin mode.

Note

Function ID: DES_SPI_API_238

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>PinCfg</i>	SPI pin mode.

Returns

void

Definition at line 1652 of file Spi_Hal.c.

4.3.4.19 Spi_Hal_SlaveGetReceiveLen()

```
uint16 Spi_Hal_SlaveGetReceiveLen (
    uint8 Instance,
    uint8 ** ReceiveBuffer )
```

This function get spi receive data when in rx buffer.

Note

Function ID: DES_SPI_API_233

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ReceiveBuffer</i>	To set the BaudRate value.

Returns

uint16: SPI get receive data len.

Definition at line 1465 of file Spi_Hal.c.

4.3.4.20 Spi_Hal_SlaveReceiveNolimitLen()

```
Hal_StatusType Spi_Hal_SlaveReceiveNolimitLen (
    uint8 Instance,
    uint8 * ReceiveBuffer1,
    uint8 * ReceiveBuffer2,
    uint16 MaxreceiveByteCount )
```

SPI using interrupt receive by NolimitLen mode, which will return after start transfer. The user needs to check whether the receiveBuffer has valid data the Spi_Hal_GetTransceiveStatus function.

Parameters

in	<i>Instance</i>	SPI module instance
in	<i>ReceiveBuffer1</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ReceiveBuffer2</i>	Pointer to the buffer where the received bytes are stored.
in	<i>MaxreceiveByteCount</i>	The receiveBuffer max len.

Returns

STATUS_SUCCESS The transfer was successful, or STATUS_BUSY Cannot perform transfer because a transfer is already in progress

Definition at line 1503 of file Spi_Hal.c.

4.3.4.21 Spi_Hal_SoftwareReset()

```
void Spi_Hal_SoftwareReset (
    uint8 Instance )
```

This function reset the SPI Status.

Note

Function ID: DES_SPI_API_237

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

void

Definition at line 1634 of file Spi_Hal.c.

4.3.4.22 Spi_Hal_TransceiveDma()

```
Hal_StatusType Spi_Hal_TransceiveDma (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    const uint8 * RxBuffPtr,
    uint16 ByteCount )
```

SPI transmission, reception by DMA.

Note

Function ID: DES_SPI_API_209

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.

Returns

Hal_StatusType

Definition at line 1402 of file Spi_Hal.c.

4.3.4.23 Spi_Hal_TransceiveInt()

```
Hal_StatusType Spi_Hal_TransceiveInt (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    uint8 * RxBuffPtr,
    uint16 ByteCount )
```

SPI transmission, reception by interrupt.

Note

Function ID: DES_SPI_API_207

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.

Returns

Hal_StatusType

Definition at line 1318 of file Spi_Hal.c.

4.3.4.24 Spi_Hal_TransceivePoll()

```
Hal_StatusType Spi_Hal_TransceivePoll (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    uint8 * RxBuffPtr,
    uint16 ByteCount,
    uint32 Timeout )
```

This function transfers by polling mode, which will not return until timeout or transfer finish.

Note

Function ID: DES_SPI_API_205

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.
in	<i>Timeout</i>	A timeout count value for the transfer. If the transfer takes longer than this amount of time, the transfer is aborted and a STATUS_TIMEOUT error returned.

Returns

Hal_StatusType

Definition at line 1225 of file Spi_Hal.c.

4.4 Spi_Hal.h File Reference

Spi driver hal api header file.

```
#include "Spi_Hal_Types.h"
```

Functions

- void [Spi_Hal_Init](#) (uint8 Instance, const [Spi_HalConfigType](#) *ConfigPtr)
This function initializes a SPI instance.
- void [Spi_Hal_DeInit](#) (uint8 Instance)

This function de-initializes a SPI instance.

- void [Spi_Hal_GetDefaultConfig](#) ([Spi_HalConfigType](#) *ConfigPtr)

This function gets default configuration for SPI hal driver.

- void [Spi_Hal_ConfigureFrame](#) (uint8 Instance, uint8 FrmSize, boolean MsbFirst)

This function configures the SPI frame parameters.

- Hal_StatusType [Spi_Hal_TransceivePoll](#) (uint8 Instance, const uint8 *TxBuffPtr, uint8 *RxBuffPtr, uint16 ByteCount, uint32 Timeout)

This function transfers by polling mode, which will not return until timeout or transfer finish.

- Hal_StatusType [Spi_Hal_TransceiveInt](#) (uint8 Instance, const uint8 *TxBuffPtr, uint8 *RxBuffPtr, uint16 ByteCount)

SPI transmission, reception by interrupt.

- Hal_StatusType [Spi_Hal_TransceiveDma](#) (uint8 Instance, const uint8 *TxBuffPtr, const uint8 *RxBuffPtr, uint16 ByteCount)

SPI transmission, reception by DMA.

- uint16 [Spi_Hal_SlaveGetReceiveLen](#) (uint8 Instance, uint8 **ReceiveBuffer)

This function get spi receive data when in rx buffer.

- Hal_StatusType [Spi_Hal_SlaveReceiveNolimitLen](#) (uint8 Instance, uint8 *ReceiveBuffer1, uint8 *ReceiveBuffer2, uint16 MaxreceiveByteCount)

SPI using interrupt receive by NolimitLen mode, which will return after start transfer. The user needs to check whether the receiveBuffer has valid data the Spi_Hal_GetTransceiveStatus function.

- Hal_StatusType [Spi_Hal_AbortTransceive](#) (uint8 Instance)

This function stop SPI Transceive.

- [Spi_TransceiveStatusType](#) [Spi_Hal_GetTransceiveStatus](#) (uint8 Instance)

This function get SPI Transceive Status.

- Hal_StatusType [Spi_Hal_SetBaudRate](#) (uint8 Instance, uint32 BaudRate)

This function set the SPI BaudRate.

- uint32 [Spi_Hal_GetBaudRate](#) (uint8 Instance)

This function get the SPI BaudRate.

- void [Spi_Hal_SoftwareReset](#) (uint8 Instance)

This function reset the SPI Status.

- void [Spi_Hal_SetPinMode](#) (uint8 Instance, [Spi_PinCfgType](#) PinCfg)

This function set the SPI pin mode.

- void [Spi_Hal_SetCsPin](#) (uint8 Instance, [Spi_PcsType](#) Pcs, [Spi_PcsPolarityType](#) Polarity)

This function select the SPI cs pin.

- void [Spi_Hal_SetMatchData](#) (uint8 Instance, uint32 Data, boolean Enable)

This function set the SPI Match Data.

- void [Spi_Hal_SetFrameSize](#) (uint8 Instance, uint8 Framesize)

This function set the SPI FrameSize.

- SPI_Type * [Spi_Hal_GetBase](#) (uint8 Instance)

Get the SPI base.

- boolean [Spi_Hal_CheckBusy](#) (uint8 Instance)

This function checks whether the status of SPI hal driver is busy.

- void [Spi_Hal_ConfigureBus](#) (uint8 Instance, const [Spi_HalConfigType](#) *ConfigPtr)

This function initializes a SPI instance.

4.4.1 Detailed Description

Spi driver hal api header file.

4.4.2 Function Documentation

4.4.2.1 Spi_Hal_AbortTransceive()

```
Hal_StatusType Spi_Hal_AbortTransceive (
    uint8 Instance )
```

This function stop SPI Transceive.

Note

Function ID: DES_SPI_API_235

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

Hal_StatusType

Definition at line 1547 of file Spi_Hal.c.

4.4.2.2 Spi_Hal_CheckBusy()

```
boolean Spi_Hal_CheckBusy (
    uint8 Instance )
```

This function checks whether the status of SPI hal driver is busy.

Note

Function ID: DES_SPI_API_211

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

boolean

- TRUE: The SPI hal driver is busy.
- FALSE: The SPI hal driver is not busy.

Definition at line 1844 of file Spi_Hal.c.

4.4.2.3 Spi_Hal_ConfigureBus()

```
void Spi_Hal_ConfigureBus (
    uint8 Instance,
    const Spi_HalConfigType * ConfigPtr )
```

This function initializes a SPI instance.

Note

Function ID: DES_SPI_API_232

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ConfigPtr</i>	The data structure containing information about a device on the SPI bus.

Returns

void

Definition at line 1039 of file Spi_Hal.c.

4.4.2.4 Spi_Hal_ConfigureFrame()

```
void Spi_Hal_ConfigureFrame (
    uint8 Instance,
    uint8 FrmSize,
    boolean MsbFirst )
```

This function configures the SPI frame parameters.

Note

Function ID: DES_SPI_API_204

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>FrmSize</i>	Number of bits per frame, support 4~32 bits.
in	<i>MsbFirst</i>	Option to transmit MSB first.

Returns

void

Definition at line 1097 of file Spi_Hal.c.

4.4.2.5 Spi_Hal_DeInit()

```
void Spi_Hal_DeInit (
    uint8 Instance )
```

This function de-initializes a SPI instance.

Note

Function ID: DES_SPI_API_201

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

void

Definition at line 1165 of file Spi_Hal.c.

4.4.2.6 Spi_Hal_GetBase()

```
SPI_Type* Spi_Hal_GetBase (
    uint8 Instance )
```

Get the SPI base.

Note

Function ID: DES_SPI_API_231

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

SPI_Type: SPI Base Addr.

Definition at line 1799 of file Spi_Hal.c.

4.4.2.7 Spi_Hal_GetBaudRate()

```
uint32 Spi_Hal_GetBaudRate (
    uint8 Instance )
```

This function get the SPI BaudRate.

Note

Function ID: DES_SPI_API_240

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

uint32:SPI BaudRate

This function get the SPI BaudRate.

Note

Function ID: DES_SPI_API_240

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

uint32:SPI BaudRate

Definition at line 1744 of file Spi_Hal.c.

4.4.2.8 Spi_Hal_GetDefaultConfig()

```
void Spi_Hal_GetDefaultConfig (
    Spi_HalConfigType * ConfigPtr )
```

This function gets default configuration for SPI hal driver.

Note

Function ID: DES_SPI_API_202

Parameters

out	<i>ConfigPtr</i>	Pointer to configuration structure which is filled with default configuration.
-----	------------------	--

Returns

void

Definition at line 1189 of file Spi_Hal.c.

4.4.2.9 Spi_Hal_GetTransceiveStatus()

```
Spi_TransceiveStatusType Spi_Hal_GetTransceiveStatus (
    uint8 Instance )
```

This function get SPI Transceive Status.

Note

Function ID: DES_SPI_API_236

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

Spi_TransceiveStatusType:the Transceive Status.

Definition at line 1587 of file Spi_Hal.c.

4.4.2.10 Spi_Hal_Init()

```
void Spi_Hal_Init (
    uint8 Instance,
    const Spi_HalConfigType * ConfigPtr )
```

This function initializes a SPI instance.

Note

Function ID: DES_SPI_API_200

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ConfigPtr</i>	The data structure containing information about a device on the SPI bus.

Returns

None

Definition at line 1124 of file Spi_Hal.c.

4.4.2.11 Spi_Hal_SetBaudRate()

```
Hal_StatusType Spi_Hal_SetBaudRate (
    uint8 Instance,
    uint32 BaudRate )
```

This function set the SPI BaudRate.

Note

Function ID: DES_SPI_API_230

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>BaudRate</i>	To set the BaudRate value.

Returns

Hal_StatusType

Definition at line 1699 of file Spi_Hal.c.

4.4.2.12 Spi_Hal_SetCsPin()

```
void Spi_Hal_SetCsPin (
    uint8 Instance,
    Spi_PcsType Pcs,
    Spi_PcsPolarityType Polarity )
```

This function select the SPI cs pin.

Note

Function ID: DES_SPI_API_239

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Pcs</i>	SPI CS select.
in	<i>Polarity</i>	CS Polarity.

Returns

void

Definition at line 1671 of file Spi_Hal.c.

4.4.2.13 Spi_Hal_SetFrameSize()

```
void Spi_Hal_SetFrameSize (
    uint8 Instance,
    uint8 Framesize )
```

This function set the SPI FrameSize.

Note

Function ID: DES_SPI_API_242

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Framesize</i>	Number of bits per frame, support 4~32 bits.

Returns

void

Definition at line 1818 of file Spi_Hal.c.

4.4.2.14 Spi_Hal_SetMatchData()

```
void Spi_Hal_SetMatchData (
    uint8 Instance,
    uint32 Data,
    boolean Enable )
```

This function set the SPI Match Data.

Note

Function ID: DES_SPI_API_241

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>Data</i>	Data to Match.
in	<i>Enable</i>	Enable or Disable Data Match.

Returns

void

Definition at line 1777 of file Spi_Hal.c.

4.4.2.15 Spi_Hal_SetPinMode()

```
void Spi_Hal_SetPinMode (
    uint8 Instance,
    Spi_PinCfgType PinCfg )
```

This function set the SPI pin mode.

Note

Function ID: DES_SPI_API_238

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>PinCfg</i>	SPI pin mode.

Returns

void

Definition at line 1652 of file Spi_Hal.c.

4.4.2.16 Spi_Hal_SlaveGetReceiveLen()

```
uint16 Spi_Hal_SlaveGetReceiveLen (
    uint8 Instance,
    uint8 ** ReceiveBuffer )
```

This function get spi receive data when in rx buffer.

Note

Function ID: DES_SPI_API_233

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>ReceiveBuffer</i>	To set the BaudRate value.

Returns

uint16: SPI get receive data len.

Definition at line 1465 of file Spi_Hal.c.

4.4.2.17 Spi_Hal_SlaveReceiveNolimitLen()

```
Hal_StatusType Spi_Hal_SlaveReceiveNolimitLen (
    uint8 Instance,
    uint8 * ReceiveBuffer1,
    uint8 * ReceiveBuffer2,
    uint16 MaxreceiveByteCount )
```

SPI using interrupt receive by NolimitLen mode, which will return after start transfer.The user needs to check whether the receiveBuffer has valid data the Spi_Hal_GetTransceiveStatus function.

Parameters

in	<i>Instance</i>	SPI module instance
in	<i>ReceiveBuffer1</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ReceiveBuffer2</i>	Pointer to the buffer where the received bytes are stored.
in	<i>MaxreceiveByteCount</i>	The receiveBuffer max len.

Returns

STATUS_SUCCESS The transfer was successful, or STATUS_BUSY Cannot perform transfer because a transfer is already in progress

Definition at line 1503 of file Spi_Hal.c.

4.4.2.18 Spi_Hal_SoftwareReset()

```
void Spi_Hal_SoftwareReset (
    uint8 Instance )
```

This function reset the SPI Status.

Note

Function ID: DES_SPI_API_237

Parameters

in	<i>Instance</i>	SPI instance.
----	-----------------	---------------

Returns

void

Definition at line 1634 of file Spi_Hal.c.

4.4.2.19 Spi_Hal_TransceiveDma()

```
Hal_StatusType Spi_Hal_TransceiveDma (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    const uint8 * RxBuffPtr,
    uint16 ByteCount )
```

SPI transmission,reception by DMA.

Note

Function ID: DES_SPI_API_209

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.

Returns

Hal_StatusType

Definition at line 1402 of file Spi_Hal.c.

4.4.2.20 Spi_Hal_TransceiveInt()

```
Hal_StatusType Spi_Hal_TransceiveInt (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    uint8 * RxBuffPtr,
    uint16 ByteCount )
```

SPI transmission, reception by interrupt.

Note

Function ID: DES_SPI_API_207

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.

Returns

Hal_StatusType

Definition at line 1318 of file Spi_Hal.c.

4.4.2.21 Spi_Hal_TransceivePoll()

```
Hal_StatusType Spi_Hal_TransceivePoll (
    uint8 Instance,
    const uint8 * TxBuffPtr,
    uint8 * RxBuffPtr,
    uint16 ByteCount,
    uint32 Timeout )
```

This function transfers by polling mode, which will not return until timeout or transfer finish.

Note

Function ID: DES_SPI_API_205

Parameters

in	<i>Instance</i>	SPI instance.
in	<i>TxBuffPtr</i>	The pointer to the data buffer of the data to send.
in, out	<i>RxBuffPtr</i>	Pointer to the buffer where the received bytes are stored.
in	<i>ByteCount</i>	The number of bytes to send and receive.
in	<i>Timeout</i>	A timeout count value for the transfer. If the transfer takes longer than this amount of time, the transfer is aborted and a STATUS_TIMEOUT error returned.

Returns

Hal_StatusType

Definition at line 1225 of file Spi_Hal.c.

4.5 Spi_Hal_Types.h File Reference

Spi driver hal types header file.

```
#include "Device_Register.h"
#include "Platform_Types.h"
```

Classes

- struct [Spi_HalConfigType](#)
User configuration structure for the hardware SPI driver.
- struct [Spi_StateType](#)
Runtime state structure for the device on the SPI bus.

Typedefs

- typedef void(* [Spi_CallbackType](#)) (uint8 Instance, uint16 Status)
Callback for SPI.

Enumerations

- enum {
[SPI_STATUS_NONE](#) = 0U, [SPI_STATUS_RX_OVERFLOW_MASK](#), [SPI_STATUS_TX_UNDERFLOW_MASK](#),
[SPI_STATUS_RX_FINISH_MASK](#),
[SPI_STATUS_TX_FINISH_MASK](#), [SPI_STATUS_DATA_MATCH_MASK](#), [SPI_STATUS_MEM_BUSY_MASK](#), [SPI_STATUS_TIMEOUT_MASK](#),
[SPI_STATUS_ABORT_MASK](#), [SPI_STATUS_DMA_ERROR_MASK](#) }
SPI Event.
- enum [Spi_StatusFlagType](#) {
[SPI_TX_DATA_FLAG](#) = 0U, [SPI_RX_DATA_FLAG](#) = 1U, [SPI_TRANSMIT_ERROR](#) = 2U, [SPI_RECEIVE_ERROR](#)
= 3U,
[SPI_MODE_FAULT_ERROR](#) = 4U, [SPI_DATA_MATCH](#) = 5U, [SPI_MASTER_BUSY](#) = 7U, [SPI_MODULE_IDLE](#) =
8U,
[SPI_ALL_STATUS](#) = 0x0000002CU }

SPI status flags.

- enum `Spi_InterruptType` {
`SPI_INT_TX_DATA` = 8U, `SPI_INT_RX_DATA` = 9U, `SPI_INT_TRANSMIT_ERROR` = 10U, `SPI_INT_RECEIVE_ERROR` = 11U,
`SPI_INT_MODE_FAULT_ERROR` = 13U, `SPI_INT_DATA_MATCH` = 31U }

SPI interrupt source.

- enum `Spi_ModeType` { `SPI_SLAVE` = 0U, `SPI_MASTER` }

SPI master or slave configuration.

- enum `Spi_PinCfgType` { `SPI_SOUT_MOSI_SIN_MISO` = 0U, `SPI_SOUT_MISO_SIN_MOSI` }

SPI pin (SOUT and SIN) configuration.

- enum `Spi_TransferWidthType` { `SPI_DATA_WIDTH_1BIT` = 0U, `SPI_DATA_WIDTH_2BIT`, `SPI_DATA_WIDTH_4BIT` }

SPI transfer data width configuration.

- enum `Spi_HreqPolarityType` { `SPI_HREQ_POLARITY_HIGH` = 0U, `SPI_HREQ_POLARITY_LOW` = 1U }

SPI hreq polarity type selection.

- enum `Spi_CpolType` { `SPI_CPOL_LOW` = 0U, `SPI_CPOL_HIGH` }

SPI Clock (SCK) polarity configuration.

- enum `Spi_CphaType` { `SPI_CPHA_0` = 0U, `SPI_CPHA_1` }

SPI clock phase configuration.

- enum `Spi_PcsPolarityType` { `SPI_PCS_POLARITY_LOW` = 0U, `SPI_PCS_POLARITY_HIGH` }

SPI PCS polarity configuration.

- enum `Spi_PcsType` {
`SPI_PCS_0` = 0U, `SPI_PCS_1`, `SPI_PCS_2`, `SPI_PCS_3`,
`SPI_PCS_GPIO` }

SPI Peripheral Chip Select(PCS) configuration.

- enum `Spi_TransferType` { `SPI_USING_DMA` = 0x00U, `SPI_USING_POLL` = 0x01U, `SPI_USING_INTERRUPT` = 0x02U }

Type of SPI transfer(based on interrupts or DMA or Poll)

- enum `Spi_TransceiveStatusType` { `SPI_TRANSCEIVE_BUSY` = 0x00U, `SPI_TRANSCEIVE_ERROR` = 0x01U, `SPI_TRANSCEIVE_SUCCESS` = 0x02U, `SPI_TRANSCEIVE_TIMEOUT` = 0x03U }

Type of SPI transfer status type.

4.5.1 Detailed Description

Spi driver hal types header file.

4.5.2 Typedef Documentation

4.5.2.1 Spi_CallbackType

```
typedef void(* Spi_CallbackType) (uint8 Instance, uint16 Status)
```

Callback for SPI.

Definition at line 181 of file Spi_Hal_Types.h.

4.5.3 Enumeration Type Documentation

4.5.3.1 anonymous enum

```
anonymous enum
```

SPI Event.

Enumerator

SPI_STATUS_NONE	SPI None status.
SPI_STATUS_RX_OVERFLOW_MASK	SPI RX overflow status.
SPI_STATUS_TX_UNDERFLOW_MASK	SPI TX underflow status.
SPI_STATUS_RX_FINISH_MASK	SPI RX finsh status.
SPI_STATUS_TX_FINISH_MASK	SPI TX finsh status.
SPI_STATUS_DATA_MATCH_MASK	SPI Data match status.
SPI_STATUS_MEM_BUSY_MASK	SPI Meby busy status.
SPI_STATUS_TIMEOUT_MASK	SPI TX/RX timeout status.
SPI_STATUS_ABORT_MASK	SPI TX/RX abort status.
SPI_STATUS_DMA_ERROR_MASK	SPI Dma error status.

Definition at line 58 of file Spi_Hal_Types.h.

4.5.3.2 Spi_CphaType

enum [Spi_CphaType](#)

SPI clock phase configuration.

Enumerator

SPI_CPHA↔ _0	Data captured on SCK 1st edge, changed on 2nd.
SPI_CPHA↔ _1	Data changed on SCK 1st edge, captured on 2nd.

Definition at line 134 of file Spi_Hal_Types.h.

4.5.3.3 Spi_CpolType

enum [Spi_CpolType](#)

SPI Clock (SCK) polarity configuration.

Enumerator

SPI_CPOL_LOW	SCK is low when idle
SPI_CPOL_HIGH	SCK is high when idle

Definition at line 127 of file Spi_Hal_Types.h.

4.5.3.4 Spi_HreqPolarityType

enum `Spi_HreqPolarityType`

SPI hreq polarity type selection.

Enumerator

<code>SPI_HREQ_POLARITY_HIGH</code>	SPI host request active polarity is high
<code>SPI_HREQ_POLARITY_LOW</code>	SPI host request active polarity is low

Definition at line 120 of file `Spi_Hal_Types.h`.

4.5.3.5 Spi_InterruptType

enum `Spi_InterruptType`

SPI interrupt source.

Enumerator

<code>SPI_INT_TX_DATA</code>	TX data not full interrupt
<code>SPI_INT_RX_DATA</code>	RX data not empty interrupt
<code>SPI_INT_TRANSMIT_ERROR</code>	Transmit Error interrupt (TX underrun)
<code>SPI_INT_RECEIVE_ERROR</code>	Receive Error interrupt (RX overrun)
<code>SPI_INT_MODE_FAULT_ERROR</code>	Master mode fault error interrupt
<code>SPI_INT_DATA_MATCH</code>	Data match interrupt

Definition at line 87 of file `Spi_Hal_Types.h`.

4.5.3.6 Spi_ModeType

enum `Spi_ModeType`

SPI master or slave configuration.

Enumerator

<code>SPI_SLAVE</code>	SPI peripheral operates in slave mode.
<code>SPI_MASTER</code>	SPI peripheral operates in master mode.

Definition at line 98 of file `Spi_Hal_Types.h`.

4.5.3.7 Spi_PcsPolarityType

enum `Spi_PcsPolarityType`

SPI PCS polarity configuration.

Enumerator

<code>SPI_PCS_POLARITY_LOW</code>	PCS active low (idle is high).
<code>SPI_PCS_POLARITY_HIGH</code>	PCS active high (idle is low).

Definition at line 141 of file `Spi_Hal_Types.h`.

4.5.3.8 Spi_PcsType

enum `Spi_PcsType`

SPI Peripheral Chip Select(PCS) configuration.

Enumerator

<code>SPI_PCS_0</code>	PCS0
<code>SPI_PCS_1</code>	PCS1
<code>SPI_PCS_2</code>	PCS2
<code>SPI_PCS_3</code>	PCS3
<code>SPI_PCS_GPIO</code>	GPIO AS CS

Definition at line 148 of file `Spi_Hal_Types.h`.

4.5.3.9 Spi_PinCfgType

enum `Spi_PinCfgType`

SPI pin (SOUT and SIN) configuration.

Enumerator

<code>SPI_SOUT_MOSI_SIN_MISO</code>	SPI SOUT pin set as MOSI, SIN pin set as MISO
<code>SPI_SOUT_MISO_SIN_MOSI</code>	SPI SOUT pin set as MISO, SIN pin set as MOSI

Definition at line 105 of file `Spi_Hal_Types.h`.

4.5.3.10 Spi_StatusFlagType

enum `Spi_StatusFlagType`

SPI status flags.

Enumerator

SPI_TX_DATA_FLAG	TX data not full flag
SPI_RX_DATA_FLAG	RX data not empty flag
SPI_TRANSMIT_ERROR	Transmit Error flag (TX underrun)
SPI_RECEIVE_ERROR	Receive Error flag (RX overrun)
SPI_MODE_FAULT_ERROR	Master mode fault error flag
SPI_DATA_MATCH	Data match flag
SPI_MASTER_BUSY	SPI engine busy flag
SPI_MODULE_IDLE	Module idle flag
SPI_ALL_STATUS	Used for clearing all w1c status flags

Definition at line 73 of file Spi_Hal_Types.h.

4.5.3.11 Spi_TransceiveStatusType

enum [Spi_TransceiveStatusType](#)

Type of SPI transfer status type.

Enumerator

SPI_TRANSCEIVE_BUSY	The transfer is on going
SPI_TRANSCEIVE_ERROR	The transfer is error
SPI_TRANSCEIVE_SUCCESS	The transfer is success
SPI_TRANSCEIVE_TIMEOUT	The transfer is timeout

Definition at line 171 of file Spi_Hal_Types.h.

4.5.3.12 Spi_TransferType

enum [Spi_TransferType](#)

Type of SPI transfer(based on interrupts or DMA or Poll)

Enumerator

SPI_USING_DMA	The driver will use DMA to perform SPI transfer
SPI_USING_POLL	The driver will use POLL to perform SPI transfer
SPI_USING_INTERRUPT	The driver will use interrupt to perform SPI transfer

Definition at line 160 of file Spi_Hal_Types.h.

4.5.3.13 Spi_TransferWidthType

enum [Spi_TransferWidthType](#)

SPI transfer data width configuration.

Enumerator

SPI_DATA_WIDTH_1BIT	SPI data line 1 bit, data out on SOUT, in on SIN (standby SPI)
SPI_DATA_WIDTH_2BIT	SPI data line 2 bits, data out/in on SOUT, SIN
SPI_DATA_WIDTH_4BIT	SPI data line 4 bits, out on SDO/SDI/PCS[3:2] and in on SDO/SDI/PCS[3:2]

Definition at line 112 of file Spi_Hal_Types.h.

Index

AC784xx_API_Reference_Manual_SPI.pdf, 11

AC784xx_Spi_Reg.h, 11

- SPI_Reg_ClearStatusFlag, 14
- SPI_Reg_IsCsContinuousMode, 18
- SPI_Reg_IsModuleEnabled, 19
- SPI_Reg_SetMasterModeFaultDetectCs0, 29
- SPI_Reg_SetMasterModeFaultDetectCs1, 30
- SPI_Reg_SetMasterModeFaultDetectCs2, 30
- SPI_Reg_SetMasterModeFaultDetectCs3, 30
- Spi_Reg_ClearRDMF, 13
- Spi_Reg_ClearRxOF, 14
- Spi_Reg_ClearTxUF, 14
- Spi_Reg_GetDataAddress, 15
- Spi_Reg_GetIntMode, 15
- Spi_Reg_GetRxOnly, 16
- Spi_Reg_GetSckHighLow, 17
- Spi_Reg_GetStatusFlag, 17
- Spi_Reg_GetTxOnly, 18
- Spi_Reg_IsMaster, 19
- Spi_Reg_ReadData, 19
- Spi_Reg_ReleaseCS, 20
- Spi_Reg_SetBaudRate, 20
- Spi_Reg_SetCPHA, 22
- Spi_Reg_SetCPOL, 22
- Spi_Reg_SetCSHold, 23
- Spi_Reg_SetCSIdle, 23
- Spi_Reg_SetCSOE, 24
- Spi_Reg_SetCSSetup, 24
- Spi_Reg_SetContinuousCS, 21
- Spi_Reg_SetDMIE, 26
- Spi_Reg_SetDataWidth, 25
- Spi_Reg_SetDebug, 26
- Spi_Reg_SetEnable, 27
- Spi_Reg_SetFrameSize, 27
- Spi_Reg_SetHreq, 28
- Spi_Reg_SetHreqPolarity, 28
- Spi_Reg_SetIntMode, 29
- Spi_Reg_SetMasterNoOverflowMode, 31
- Spi_Reg_SetMasterSlaveMode, 31
- Spi_Reg_SetMatchData, 32
- Spi_Reg_SetModeFault, 32
- Spi_Reg_SetPcsCfg, 33
- Spi_Reg_SetPcsPolarity, 33
- Spi_Reg_SetPinConfigMode, 34
- Spi_Reg_SetROTRIG, 35
- Spi_Reg_SetRxDmaCmd, 35
- Spi_Reg_SetRxMSB, 36
- Spi_Reg_SetRxOnly, 36
- Spi_Reg_SetTxDmaCmd, 37
- Spi_Reg_SetTxMSB, 37
- Spi_Reg_SetTxOnly, 38
- Spi_Reg_SoftwareReset, 38

Spi_Reg_WriteData, 39

- BaudRate
 - Spi_HalConfigType, 4
- Callback
 - Spi_HalConfigType, 4
 - Spi_StateType, 8
- ContinuousCs
 - Spi_HalConfigType, 4
- Cpha
 - Spi_HalConfigType, 4
- Cpol
 - Spi_HalConfigType, 4
- CsHold
 - Spi_HalConfigType, 4
- CsIdle
 - Spi_HalConfigType, 5
- CsOutputEn
 - Spi_HalConfigType, 5
- CsSetup
 - Spi_HalConfigType, 5
- FrmSize
 - Spi_HalConfigType, 5
 - Spi_StateType, 8
- HreqEn
 - Spi_HalConfigType, 5
- HreqPol
 - Spi_HalConfigType, 6
- InProgress
 - Spi_StateType, 8
- Mode
 - Spi_HalConfigType, 6
- MsbFirst
 - Spi_HalConfigType, 6
 - Spi_StateType, 9
- PcsCfg
 - Spi_HalConfigType, 6
- PcsPol
 - Spi_HalConfigType, 6
- PinCfg
 - Spi_HalConfigType, 7
- RxBuffPtr
 - Spi_StateType, 9
- RxBuffPtr1
 - Spi_StateType, 9
- RxCount

- Spi_StateType, [9](#)
- RxDmaChannel
 - Spi_HalConfigType, [7](#)
 - Spi_StateType, [9](#)
- SPI0_IRQHandler
 - Spi_Hal.c, [43](#)
- SPI1_IRQHandler
 - Spi_Hal.c, [43](#)
- SPI2_IRQHandler
 - Spi_Hal.c, [43](#)
- SPI_1BYTE_MSK
 - Spi_Hal.c, [41](#)
- SPI_2BYTES_MSK
 - Spi_Hal.c, [41](#)
- SPI_BIT16_POS
 - Spi_Hal.c, [42](#)
- SPI_BIT24_POS
 - Spi_Hal.c, [42](#)
- SPI_BIT8_POS
 - Spi_Hal.c, [42](#)
- SPI_Reg_ClearStatusFlag
 - AC784xx_Spi_Reg.h, [14](#)
- SPI_Reg_IsCsContinuousMode
 - AC784xx_Spi_Reg.h, [18](#)
- SPI_Reg_IsModuleEnabled
 - AC784xx_Spi_Reg.h, [19](#)
- SPI_Reg_SetMasterModeFaultDetectCs0
 - AC784xx_Spi_Reg.h, [29](#)
- SPI_Reg_SetMasterModeFaultDetectCs1
 - AC784xx_Spi_Reg.h, [30](#)
- SPI_Reg_SetMasterModeFaultDetectCs2
 - AC784xx_Spi_Reg.h, [30](#)
- SPI_Reg_SetMasterModeFaultDetectCs3
 - AC784xx_Spi_Reg.h, [30](#)
- SPI_TIMEOUT_FRAME
 - Spi_Hal.c, [42](#)
- Spi_CallbackType
 - Spi_Hal_Types.h, [67](#)
- Spi_CphaType
 - Spi_Hal_Types.h, [68](#)
- Spi_CpolType
 - Spi_Hal_Types.h, [68](#)
- Spi_Hal.c, [39](#)
 - SPI0_IRQHandler, [43](#)
 - SPI1_IRQHandler, [43](#)
 - SPI2_IRQHandler, [43](#)
 - SPI_1BYTE_MSK, [41](#)
 - SPI_2BYTES_MSK, [41](#)
 - SPI_BIT16_POS, [42](#)
 - SPI_BIT24_POS, [42](#)
 - SPI_BIT8_POS, [42](#)
 - SPI_TIMEOUT_FRAME, [42](#)
 - Spi_Hal_AbortTransceive, [44](#)
 - Spi_Hal_CheckBusy, [44](#)
 - Spi_Hal_ConfigureBus, [45](#)
 - Spi_Hal_ConfigureFrame, [45](#)
 - Spi_Hal_DelInit, [46](#)
 - Spi_Hal_GetBase, [46](#)
 - Spi_Hal_GetBaudRate, [47](#)
 - Spi_Hal_GetDefaultConfig, [47](#)
- Spi_Hal_GetTransceiveStatus, [48](#)
- Spi_Hal_Init, [48](#)
- Spi_Hal_SetBaudRate, [48](#)
- Spi_Hal_SetCsPin, [49](#)
- Spi_Hal_SetFrameSize, [49](#)
- Spi_Hal_SetMatchData, [50](#)
- Spi_Hal_SetPinMode, [50](#)
- Spi_Hal_SlaveGetReceiveLen, [51](#)
- Spi_Hal_SlaveReceiveNolimitLen, [51](#)
- Spi_Hal_SoftwareReset, [52](#)
- Spi_Hal_TransceiveDma, [52](#)
- Spi_Hal_TransceiveInt, [53](#)
- Spi_Hal_TransceivePoll, [54](#)
- Spi_TransceiveType, [42](#)
- Spi_Hal.h, [54](#)
 - Spi_Hal_AbortTransceive, [55](#)
 - Spi_Hal_CheckBusy, [56](#)
 - Spi_Hal_ConfigureBus, [56](#)
 - Spi_Hal_ConfigureFrame, [57](#)
 - Spi_Hal_DelInit, [57](#)
 - Spi_Hal_GetBase, [58](#)
 - Spi_Hal_GetBaudRate, [58](#)
 - Spi_Hal_GetDefaultConfig, [59](#)
 - Spi_Hal_GetTransceiveStatus, [59](#)
 - Spi_Hal_Init, [60](#)
 - Spi_Hal_SetBaudRate, [60](#)
 - Spi_Hal_SetCsPin, [61](#)
 - Spi_Hal_SetFrameSize, [61](#)
 - Spi_Hal_SetMatchData, [62](#)
 - Spi_Hal_SetPinMode, [62](#)
 - Spi_Hal_SlaveGetReceiveLen, [63](#)
 - Spi_Hal_SlaveReceiveNolimitLen, [63](#)
 - Spi_Hal_SoftwareReset, [64](#)
 - Spi_Hal_TransceiveDma, [64](#)
 - Spi_Hal_TransceiveInt, [65](#)
 - Spi_Hal_TransceivePoll, [65](#)
- Spi_Hal_AbortTransceive
 - Spi_Hal.c, [44](#)
 - Spi_Hal.h, [55](#)
- Spi_Hal_CheckBusy
 - Spi_Hal.c, [44](#)
 - Spi_Hal.h, [56](#)
- Spi_Hal_ConfigureBus
 - Spi_Hal.c, [45](#)
 - Spi_Hal.h, [56](#)
- Spi_Hal_ConfigureFrame
 - Spi_Hal.c, [45](#)
 - Spi_Hal.h, [57](#)
- Spi_Hal_DelInit
 - Spi_Hal.c, [46](#)
 - Spi_Hal.h, [57](#)
- Spi_Hal_GetBase
 - Spi_Hal.c, [46](#)
 - Spi_Hal.h, [58](#)
- Spi_Hal_GetBaudRate
 - Spi_Hal.c, [47](#)
 - Spi_Hal.h, [58](#)
- Spi_Hal_GetDefaultConfig
 - Spi_Hal.c, [47](#)
 - Spi_Hal.h, [59](#)
- Spi_Hal_GetTransceiveStatus

- [Spi_Hal.c, 48](#)
 - [Spi_Hal.h, 59](#)
- [Spi_Hal_Init](#)
 - [Spi_Hal.c, 48](#)
 - [Spi_Hal.h, 60](#)
- [Spi_Hal_SetBaudRate](#)
 - [Spi_Hal.c, 48](#)
 - [Spi_Hal.h, 60](#)
- [Spi_Hal_SetCsPin](#)
 - [Spi_Hal.c, 49](#)
 - [Spi_Hal.h, 61](#)
- [Spi_Hal_SetFrameSize](#)
 - [Spi_Hal.c, 49](#)
 - [Spi_Hal.h, 61](#)
- [Spi_Hal_SetMatchData](#)
 - [Spi_Hal.c, 50](#)
 - [Spi_Hal.h, 62](#)
- [Spi_Hal_SetPinMode](#)
 - [Spi_Hal.c, 50](#)
 - [Spi_Hal.h, 62](#)
- [Spi_Hal_SlaveGetReceiveLen](#)
 - [Spi_Hal.c, 51](#)
 - [Spi_Hal.h, 63](#)
- [Spi_Hal_SlaveReceiveNoLimitLen](#)
 - [Spi_Hal.c, 51](#)
 - [Spi_Hal.h, 63](#)
- [Spi_Hal_SoftwareReset](#)
 - [Spi_Hal.c, 52](#)
 - [Spi_Hal.h, 64](#)
- [Spi_Hal_TransceiveDma](#)
 - [Spi_Hal.c, 52](#)
 - [Spi_Hal.h, 64](#)
- [Spi_Hal_TransceiveInt](#)
 - [Spi_Hal.c, 53](#)
 - [Spi_Hal.h, 65](#)
- [Spi_Hal_TransceivePoll](#)
 - [Spi_Hal.c, 54](#)
 - [Spi_Hal.h, 65](#)
- [Spi_Hal_Types.h, 66](#)
 - [Spi_CallbackType, 67](#)
 - [Spi_CphaType, 68](#)
 - [Spi_CpolType, 68](#)
 - [Spi_HreqPolarityType, 68](#)
 - [Spi_InterruptType, 69](#)
 - [Spi_ModeType, 69](#)
 - [Spi_PcsPolarityType, 69](#)
 - [Spi_PcsType, 70](#)
 - [Spi_PinCfgType, 70](#)
 - [Spi_StatusFlagType, 70](#)
 - [Spi_TransceiveStatusType, 71](#)
 - [Spi_TransferType, 71](#)
 - [Spi_TransferWidthType, 71](#)
- [Spi_HalConfigType, 3](#)
 - [BaudRate, 4](#)
 - [Callback, 4](#)
 - [ContinuousCs, 4](#)
 - [Cpha, 4](#)
 - [Cpol, 4](#)
 - [CsHold, 4](#)
 - [CsIdle, 5](#)
 - [CsOutputEn, 5](#)
 - [CsSetup, 5](#)
 - [FrmSize, 5](#)
 - [HreqEn, 5](#)
 - [HreqPol, 6](#)
 - [Mode, 6](#)
 - [MsbFirst, 6](#)
 - [PcsCfg, 6](#)
 - [PcsPol, 6](#)
 - [PinCfg, 7](#)
 - [RxDmaChannel, 7](#)
 - [TxDmaChannel, 7](#)
 - [Width, 7](#)
- [Spi_HreqPolarityType](#)
 - [Spi_Hal_Types.h, 68](#)
- [Spi_InterruptType](#)
 - [Spi_Hal_Types.h, 69](#)
- [Spi_ModeType](#)
 - [Spi_Hal_Types.h, 69](#)
- [Spi_PcsPolarityType](#)
 - [Spi_Hal_Types.h, 69](#)
- [Spi_PcsType](#)
 - [Spi_Hal_Types.h, 70](#)
- [Spi_PinCfgType](#)
 - [Spi_Hal_Types.h, 70](#)
- [Spi_Reg_ClearRDMF](#)
 - [AC784xx_Spi_Reg.h, 13](#)
- [Spi_Reg_ClearRxOF](#)
 - [AC784xx_Spi_Reg.h, 14](#)
- [Spi_Reg_ClearTxUF](#)
 - [AC784xx_Spi_Reg.h, 14](#)
- [Spi_Reg_GetDataAddress](#)
 - [AC784xx_Spi_Reg.h, 15](#)
- [Spi_Reg_GetIntMode](#)
 - [AC784xx_Spi_Reg.h, 15](#)
- [Spi_Reg_GetRxOnly](#)
 - [AC784xx_Spi_Reg.h, 16](#)
- [Spi_Reg_GetSckHighLow](#)
 - [AC784xx_Spi_Reg.h, 17](#)
- [Spi_Reg_GetStatusFlag](#)
 - [AC784xx_Spi_Reg.h, 17](#)
- [Spi_Reg_GetTxOnly](#)
 - [AC784xx_Spi_Reg.h, 18](#)
- [Spi_Reg_IsMaster](#)
 - [AC784xx_Spi_Reg.h, 19](#)
- [Spi_Reg_ReadData](#)
 - [AC784xx_Spi_Reg.h, 19](#)
- [Spi_Reg_ReleaseCS](#)
 - [AC784xx_Spi_Reg.h, 20](#)
- [Spi_Reg_SetBaudRate](#)
 - [AC784xx_Spi_Reg.h, 20](#)
- [Spi_Reg_SetCPHA](#)
 - [AC784xx_Spi_Reg.h, 22](#)
- [Spi_Reg_SetCPOL](#)
 - [AC784xx_Spi_Reg.h, 22](#)
- [Spi_Reg_SetCSHold](#)
 - [AC784xx_Spi_Reg.h, 23](#)
- [Spi_Reg_SetCSIdle](#)
 - [AC784xx_Spi_Reg.h, 23](#)
- [Spi_Reg_SetCSOE](#)
 - [AC784xx_Spi_Reg.h, 24](#)
- [Spi_Reg_SetCSSetup](#)

- AC784xx_Spi_Reg.h, [24](#)
- Spi_Reg_SetContinuousCS
 - AC784xx_Spi_Reg.h, [21](#)
- Spi_Reg_SetDMIE
 - AC784xx_Spi_Reg.h, [26](#)
- Spi_Reg_SetDataWidth
 - AC784xx_Spi_Reg.h, [25](#)
- Spi_Reg_SetDebug
 - AC784xx_Spi_Reg.h, [26](#)
- Spi_Reg_SetEnable
 - AC784xx_Spi_Reg.h, [27](#)
- Spi_Reg_SetFrameSize
 - AC784xx_Spi_Reg.h, [27](#)
- Spi_Reg_SetHreq
 - AC784xx_Spi_Reg.h, [28](#)
- Spi_Reg_SetHreqPolarity
 - AC784xx_Spi_Reg.h, [28](#)
- Spi_Reg_SetIntMode
 - AC784xx_Spi_Reg.h, [29](#)
- Spi_Reg_SetMasterNoOverflowMode
 - AC784xx_Spi_Reg.h, [31](#)
- Spi_Reg_SetMasterSlaveMode
 - AC784xx_Spi_Reg.h, [31](#)
- Spi_Reg_SetMatchData
 - AC784xx_Spi_Reg.h, [32](#)
- Spi_Reg_SetModeFault
 - AC784xx_Spi_Reg.h, [32](#)
- Spi_Reg_SetPcsCfg
 - AC784xx_Spi_Reg.h, [33](#)
- Spi_Reg_SetPcsPolarity
 - AC784xx_Spi_Reg.h, [33](#)
- Spi_Reg_SetPinConfigMode
 - AC784xx_Spi_Reg.h, [34](#)
- Spi_Reg_SetROTRIG
 - AC784xx_Spi_Reg.h, [35](#)
- Spi_Reg_SetRxDmaCmd
 - AC784xx_Spi_Reg.h, [35](#)
- Spi_Reg_SetRxMSB
 - AC784xx_Spi_Reg.h, [36](#)
- Spi_Reg_SetRxOnly
 - AC784xx_Spi_Reg.h, [36](#)
- Spi_Reg_SetTxDmaCmd
 - AC784xx_Spi_Reg.h, [37](#)
- Spi_Reg_SetTxMSB
 - AC784xx_Spi_Reg.h, [37](#)
- Spi_Reg_SetTxOnly
 - AC784xx_Spi_Reg.h, [38](#)
- Spi_Reg_SoftwareReset
 - AC784xx_Spi_Reg.h, [38](#)
- Spi_Reg_WriteData
 - AC784xx_Spi_Reg.h, [39](#)
- Spi_StateType, [8](#)
 - Callback, [8](#)
 - FrmSize, [8](#)
 - InProgress, [8](#)
 - MsbFirst, [9](#)
 - RxBuffPtr, [9](#)
 - RxBuffPtr1, [9](#)
 - RxCount, [9](#)
 - RxDmaChannel, [9](#)
 - Status, [10](#)
 - TransferType, [10](#)
 - TxBuffPtr, [10](#)
 - TxCount, [10](#)
 - TxDmaChannel, [10](#)
 - XferCount, [10](#)
- Spi_StatusFlagType
 - Spi_Hal_Types.h, [70](#)
- Spi_TransceiveStatusType
 - Spi_Hal_Types.h, [71](#)
- Spi_TransceiveType
 - Spi_Hal.c, [42](#)
- Spi_TransferType
 - Spi_Hal_Types.h, [71](#)
- Spi_TransferWidthType
 - Spi_Hal_Types.h, [71](#)
- Status
 - Spi_StateType, [10](#)
- TransferType
 - Spi_StateType, [10](#)
- TxBuffPtr
 - Spi_StateType, [10](#)
- TxCount
 - Spi_StateType, [10](#)
- TxDmaChannel
 - Spi_HalConfigType, [7](#)
 - Spi_StateType, [10](#)
- Width
 - Spi_HalConfigType, [7](#)
- XferCount
 - Spi_StateType, [10](#)