# Module -- Spi Driver

# High-Level Description

This module provides the following Autosar API:

1. Spi\_SetupEB()
2. Spi\_Init()
3. Spi\_AsyncTransmit()
4. Spi\_GetSequenceResult()

This module provides the following TI Halcogen API:

1. mibspiSetCtrlData() – *Note: this is a modified form of the standard mibspiSetData() API*
2. mibspiTransfer()
3. mibspiGetData()
4. mibspiSetData()

The Autosar API naming has been altered from the Autosar standard to allow co-existence of this module and a Third Party Spi driver implementation in the same project. SWC’s operating within the Rte can be mapped to either the Spi service ports offered by this BSW or the Third Parties Spi driver service ports by only changing the Rte service port mapping.

This driver exists to provide configurations/use cases that cannot provided by the third party Spi driver due to limitations in the design of the module.

The subset of the Texas Instruments Halcogen mibspi API is provided specifically to support the Turns Counter Flash Programming SWC and the Digital MSB SWC. If the Turns Counter Flash Programming SWC design and the Digital MSB design changed to use the standard Autosar API, then the provided mibspi API could be changed to module internal functions or removed. However, the requirements of the Digital MSB component are such that its SPI usage does not fit easily into the Autosar API definition (more detail provided in section 9).

## References

1. PIC16(L)F1847 Data Sheet – DS41453B (41453B.pdf)
2. Turns Counter Column Position Sensor FDD 20C ([FDD 20C Turns Counter Column Position Sensor (BMW EA3) Rev 03.doc](http://misagweb01.nexteer.com/eRoomReq/Files/erooms8/NextGeneration/0_138945/FDD%2020C%20Turns%20Counter%20Column%20Position%20Sensor%20(BMW-EA3)%20Rev%20003.doc))
3. TMS570LS31x/21x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual – September 2011 (spnu499.pdf)
4. Specification of the SPI Handler/Driver v3.0.0 (AUTOSAR\_SWS\_SPIHandlerDriver.pdf)
5. Turns Counter Flash Programming FDD 98 Rev 002
6. Digital MSB FDD ES50ARev005 7-Feb-14
7. Allegro A1331 Data Sheet Addendum – Programming Reference A1331-ADD1

# Figures

## Diagram – Function Data Sharing

This diagram shows all data that is shared between functions within the module.



### Diagram – Function (Name)

This diagram describes the functional characteristics and data flow of a given function.

(Note – This is not mandatory, only used where a graphical representation helps explain the function. It is left to the author’s discretion. New headers of this level (Level 3) should be created for each function.

# Variable Data Dictionary

For details on module input / output variable, refer to the Data Dictionary for the application. Input / output variable names are listed here for reference.

(Note: Full variable names required in table.)

(Note: All global variables including End Of Line data used should be shown here)

|  |  |  |
| --- | --- | --- |
| Module Inputs | Module Outputs | |
| None | | None |

## Module Internal Variables

This section identifies the name, range and resolutions for module specific data created by this module. If there are no range restrictions on the variable, the term “FULL” is placed into the table for legal range.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Resolution | Legal Range  (min) | Legal Range  (max) | Software Segment |
| ExtBufCfg\_Str  [D\_SPINXTNUMCHAN\_CNT\_U16] | See structure definition | See structure definition | See structure definition | SPINXT\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| SeqResult\_Enum  [D\_SPINXTNUMSEQ\_CNT\_U16] | Enum | SPI\_SEQ\_OK (= 0) | SPI\_SEQ\_CANCELLED (=4) | SPINXT\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |

### User defined typedef definition/declaration

This section documents any user types uniquely used for the module.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Typedef Name | Element Name | User Defined Type | Legal Range  (min) | Legal Range  (max) |
| EbCfg\_Type | SrcDataBufferPtr | Spi\_DataType | N/A | N/A |
|  | DesDataBufferPtr | Spi\_DataType | N/A | N/A |
|  | Length | Spi\_NumberOfDataType | N/A | N/A |

# Constant Data Dictionary

## Calibration Constants

This section lists the calibrations used by the module. For details on calibration constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
| <None> |
|  |

## Program(fixed) Constants

### Embedded Constants

All embedded constants whose values are provided in Eng units will be evaluated to the equivalent counts by using the FPM\_InitFixedPoint\_m() macro within the #define statement.

#### Local

| Constant Name | Resolution | Units | Value when D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXTUSEWITHTC | Value when D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXTUSEWITHDIGMSB |
| --- | --- | --- | --- | --- |
| D\_SPINXTUSEWITHTC | Count | Unitless | 0 | 0 |
| D\_SPINXTUSEWITHDIGMSB | Count | Unitless | 1 | 1 |
| D\_SPINXTUSEWITH\_CNT\_ENUM | Count | Unitless | 0 | 1 |
| SPI\_TCDATA\_CH | Count | Unitless | 0 | Not Defined |
| SPI\_TCDATA\_SEQ | Count | Unitless | 0 | Not Defined |
| SPI\_DIE1DATA\_CH | Count | Unitless | Not Defined | 0 |
| SPI\_DIE2DATA\_CH | Count | Unitless | Not Defined | 1 |
| SPI\_DIE1DATA\_SEQ | Count | Unitless | Not Defined | 0 |
| SPI\_DIE2DATA\_SEQ | Count | Unitless | Not Defined | 1 |
| D\_SPINXTNUMCHAN\_CNT\_U16 | Count | Unitless | 1 | 2 |
| D\_SPINXTNUMSEQ\_CNT\_U16 | Count | Unitless | 1 | 2 |
| D\_TGSIZE\_CNT\_U16 | Count | Unitless | Not Defined | 3 |
| MIBSPI3\_GCR1 | Count | Unitless | 3 | 3 |
| MIBSPI5\_GCR1 | Count | Unitless | 3 | 3 |
| MIBSPI3\_DELAY | Count | Unitless | 0xFFFF0000 | 0x04010000 |
| MIBSPI5\_DELAY | Count | Unitless | 0 | 0x04010000 |
| MIBSPI3\_FMT0 | Count | Unitless | 0xFF00FF08 | 0x04020710 |
| MIBSPI5\_FMT0 | Count | Unitless | 0x5E148206 | 0x04020710 |
| MIBSPI5\_FMT1 | Count | Unitless | 0x5E148210 | Not Defined |
| MIBSPI5\_FMT2 | Count | Unitless | 0x0014820B | Not Defined |
| MIBSPI3\_TGCNTRL0 | Count | Unitless | 0x403F0000 | 0x001C0000 |
| MIBSPI3\_TGCNTRL1 | Count | Unitless | 0x00000A00 | 0x402C0300 |
| MIBSPI3\_TGCTRL2 | Count | Unitless | Not Defined | 0x00700600 |
| MIBSPI3\_TOTALTGLENGTH | Count | Unitless | 10 | 6 |
| MIBSPI5\_TGCNTRL0 | Count | Unitless | 0x40700000 | 0x001E0000 |
| MIBSPI5\_TGCNTRL1 | Count | Unitless | 0x40700300 | 0x402E0300 |
| MIBSPI5\_TGCNTRL2 | Count | Unitless | 0x40700600 | 0x00700600 |
| MIBSPI5\_TGCNTRL3 | Count | Unitless | 0x40701000 | Not Defined |
| MIBSPI5\_TGCNTRL4 | Count | Unitless | 0x40701400 | Not Defined |
| MIBSPI5\_TGCNTRL5 | Count | Unitless | 0x40707500 | Not Defined |
| MIBSPI5\_TOTALTGLENGTH | Count | Unitless | 117 | 6 |
| MIBSPI3\_BUFRAMCTRLINIT | Count | Unitless | 0x84F7 | 0x8CFE |
| MIBSPI5\_BUFRAMCTRLINIT | Count | Unitless | Not Defined | 0x8CFE |
| MIBSPI3\_INTCFG | Count | Unitless | Defined | Not Defined |
| MIBSPI3\_LVL | Count | Unitless | 0 | Not Defined |
| MIBSPI3\_INT0 | Count | Unitless | 0 | Not Defined |
| MIBSPI3\_TICKCNT | Count | Unitless | 0x80000010 | 0 |
| MIBSPI5\_TICKCNT | Count | Unitless | Not Defined | 0 |
| MIBSPI3\_TG0\_NOTIF | Count | Unitless | Defined | Not Defined |
| MIBSPI3\_DMACTRL0 | Count | Unitless | Not Defined | 0x02108000 when BC\_SPINXT\_USEDMA is STD\_ON; otherwise undefined |
| MIBSPI5\_DMACTRL0 | Count | Unitless | Not Defined | 0x02108000 when BC\_SPINXT\_USEDMA is STD\_ON; otherwise undefined |

#### Global

This section lists the global constants used by the module. For details on global constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
|  |
|  |

### Module specific Lookup Tables Constants

(This is for lookup tables (arrays) with fixed values, same name as other tables)

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Value | Software Segment |
| None |  |  |  |

# Functions/Macros used by the Sub-Modules

## Library Functions / Macros

The library and functions / Macros that are called by the various sub modules are identified below,

1. CALL\_MIBSPI3\_NOTIFFCN()

## Data Hiding Functions

1. <None>

## Global Functions/Macros Defined by this Module

### Spi Setup External Buffer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | SpiNxt\_SetupEB | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Channel | Spi\_ChannelType | 0 | 255 |  |
|  | SrcDataBufferPtr | Spi\_DataType | NA | NA |  |
|  | DesDataBufferPtr | Spi\_DataType | NA | NA |  |
|  | Length | Spi\_NumberOfDataType | 0 | 65535 |  |
| **Return Value** |  | Std\_ReturnType | E\_OK | E\_NOT\_OK |  |

#### Description

This driver function is a minimalist implementation intended for use with the Turns Counter component. Additionally this function does not provide any error detection at this time. The function body is conditionally compiled such that the SetupEB functionality is provided when D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXT\_USEWITHTC, and the function simply returns E\_NOT\_OK when D\_SPINXTUSEWITH\_CNT\_ENUM has any other value.



### Spi Get Sequence Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | SpiNxt\_GetSequenceResult | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Sequence | Spi\_SequenceType | 0 | 255 |  |
| **Return Value** |  | Spi\_SeqResultType | SPI\_SEQ\_OK | SPI\_SEQ\_CANCELLED |  |

#### Description



### MIBSPI Transfer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiTransfer | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t \* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

Set the Enable Transfer Group bit in the appropriate register to enable the transfer (when triggered) of the mibspi transfer group defined by the arguments:

mibspi->TGCTRL[group] |= 0x80000000UL

### Spi Asynchronous Transmit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | SpiNxt\_AsyncTransmit | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Sequence | Spi\_SequenceType | 0 | 255 |  |
| **Return Value** | result\_T\_Cnt | Std\_ReturnType | E\_OK | E\_NOT\_OK |  |

#### Description

This driver function is a minimalist implementation intended for use with the Turns Counter component. It assumes one-to-one channel-to-sequence correspondence. The function body is conditionally compiled such that the Turns Counter-specific Asynchrounous Transmit functionality is provided when D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXT\_USEWITHTC, and the function simply returns E\_NOT\_OK when D\_SPINXTUSEWITH\_CNT\_ENUM has any other value.



### MIBSPI Set Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiSetData | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | data | uint16[] | FULL | FULL |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

This function copies transmit data from the data[] argument into the appropriate mibspi ram transmit buffer, as selected by the mibspi and group arguments. It is based on the Halcogen function of the same name.



### MIBSPI Set Control and Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiSetCtrlData | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | data | Uint32[] | FULL | FULL |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

This function copies transmit control and data words from the data[] argument into the appropriate mibspi ram transmit buffer, as selected by the mibspi and group arguments. It is based on the Halcogen function mibspiSetData, modified to copy the complete transmit buffer including both control and data

****

### MIBSPI Get Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiGetData | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | data | uint16[] | FULL | FULL |  |
| **Return Value** | [error flags] | uint32 | 0 | 0x5F |  |

#### Description

This function copies receive data from the appropriate mibspi ram receive buffer, as selected by the mibspi and group arguments, into the data[] argument. It is based on the Halcogen function of the same name, modified to optimize for use with the Digital MSB since it will be called from the Motor Control ISR. The function is conditionally compiled such that optimized Digital MSB specific functionality is provided when D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXT\_USEWITHDIGMSB, and the original Halcogen functionality is provided when D\_SPINXTUSEWITH\_CNT\_ENUM has any other value.



## Local Functions/Macros Used by this MDD only

### MIBSPI Set Data 8 Bit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiSetData8 | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | data | uint8[] | FULL | FULL |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

This function copies transmit data from the data[] argument into the appropriate mibspi ram transmit buffer, as selected by the mibspi and group arguments. It is based on the Halcogen function mibspiSetData, modified to copy from a uint8 buffer rather than uint16. 

### MIBSPI Get Data 8 Bit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiSetData8 | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | data | uint8[] | FULL | FULL |  |
| **Return Value** | [error flags] | uint32 | 0 | 0x5F |  |

#### Description

This function copies receive data from the appropriate mibspi ram transmit buffer, as selected by the mibspi and group arguments, into the data[] argument. It is based on the Halcogen function mibspiGetData, modified to copy to a uint8 buffer rather than uint16. 

### MIBSPI Enable Group Notification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiEnableGroupNotification | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t \* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
|  | level | uint32 | 0 | 1 |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

This function enables the transfer group interrupt for the mibspi, transfer group, and interrupt level defined by the arguments.



### MIBSPI Notification

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiNotification | Type | Min | Max | | UTP Tol. |
| **Arguments Passed** | **mibspi** | **mibspiBASE\_t\*** |  | |  |  |
|  | flags | uint32 |  |  | |  |
| **Return Value** | N/A |  |  |  | |  |

#### Description

This is an error callback that is provided by the application and is called on an error interrupt. The parameter passed to the callback is a copy of the error interrupt flag register. It is hardcoded for channel 0/sequence 0 and the Turns Counter application. Because it is hardcoded for the Turns Counter application, it is conditionally compiled such that it provides the Turns Counter error callback functionality when compiled with D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXT\_USEWITHTC, and returns without doing anything when compiled with D\_SPINXTUSEWITH\_CNT\_ENUM set to any other value.



### MIBSPI Group Notification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | mibspiGroupNotification | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | mibspi | mibspiBASE\_t\* | FULL | FULL |  |
|  | group | uint32 | 0 | 7 |  |
| **Return Value** | N/A |  |  |  |  |

#### Description

This is a callback function provided by the application. It is called when a transfer is complete. The parameter is the transfer group that triggered the interrupt. This driver is designed to only receive data on mibspi3 group 0 for the Turns Counter communication function. Because it is hardcoded for the Turns Counter application, it is conditionally compiled such that it provides the Turns Counter error callback functionality when compiled with D\_SPINXTUSEWITH\_CNT\_ENUM == D\_SPINXT\_USEWITHTC, and returns without doing anything when compiled with D\_SPINXTUSEWITH\_CNT\_ENUM set to any other value.



# Software Module Implementation

## Runtime Environment (RTE) Initial Values

This section lists the initial values of data written by this module but controlled by the RTE. After RTE initialization, the data in this table will contain these values.

|  |  |
| --- | --- |
| Data | Value |
| <None> |  |

## Initialization Functions

### Init: SpiNxt\_Init

#### Design Rationale

The initialization function initializes mibspi3 and mibspi5 registers as needed by the Turns Counter and Turns Counter Flash Programming components, or by the Digital MSB component, controlled by a SpiNxt configuration parameter (see Integration Manual) which controls a constant value in the generated SpiNxt\_Cfg.h file. The initialization values are hardcoded in the SpiNxt.h file and are specified by the Turns Counter (FDD20C), Turns Counter Reflash (FDD98) and Digital MSB (ES-50A) FDDs.

Additional information regarding initialization of MIBSPI3 when used by Turns Counter (per FDD20C):

The actual chip select for the PIC Turns Counter SPI transfers on MIBSPI3 uses a Dio channel, configured by the Dio and Port components as SPI\_TCCS. The required chip select hold time before SPI clocking begins is controlled by the use of the MIBSPI3 tick counter triggering the SPI transfer. (See SpiNxt\_AsyncTransmit and the value of MIBSPI3\_TICKCNT.)

The Dio chip select must remain low throughout the 10 byte transfer, with a minimum 9.5 usec delay between bytes. It is desired to implement the 9.5 usec delay in hardware. In order to accomplish this with the TMS570 SPI peripheral, it is necessary to use the C2TDELAY, the T2CDELAY, and WDELAY as the sum of all three is needed to reach the desired delay time at an 80MHz VCLK rate. More details provided in the C2TDELAY, T2CDELAY, and WDELAY sections of the tables below, to explain the values of MIBSPI3\_DELAY and MIBSPI3\_FMT0.

However, use of C2TDELAY and T2CDELAY requires transmit control words configured to use a MIBSPI3 chip select pin and to release the chip select between transfers. Therefore a “dummy” SPI chip select is used in the control words in the transmit buffer (see value of MIBSPI3\_BUFRAMCTRLINIT). This chip select is toggled during the SPI transfer but only to accomplish the delay timing; it is not toggling a pin on the PIC Turns Counter.

|  |  |  |
| --- | --- | --- |
| **MIBSPI 3 SPI Delay Register (SPIDELAY)** | | |
| Position | Desired Value | Description |
| bit 31-24 | 255 | **C2TDELAY:** Chip-select-active to transmit-start delay  t C2TDELAY= (C2TDELAY + 2) **\***VCLK Period   * Per §8.5.3 of [2] parameter 3 specifies a 9.5 uS clock idle time between bytes. * It is desired to implement the delay in hardware. In order to accomplish this with the TMS570 SPI peripheral, the Spi to CS CLK start delay is used. * Per [2] the FDD design desires to maximize the clock idle time between bytes using the hardware delays available. * Table 24-26 of [3] indicates the C2TDELAY has a max of 1FFh, however, the field is 8 bits wide so this is considered a typo in the data sheet and the actual max is FFh (255). With an 80MHz VCLK, using the equation in the table yields a maximum delay of 3.2125 uS. |
| bit 23-16 | 255 | **T2CDELAY:** Transmit-end-to-chip-select-inactive-delay  t T2CDELAY= (T2CDELAY +1) **\***VCLK Period   * Per §8.5.3 of [2] parameter 4 specifies a minimum 1.5 uS delay. * Per [2] the FDD design desires to maximize the clock idle time between bytes using the hardware delays available. * Table 24-26 of [3] indicates the T2CDELAY has a max of 1FFh, however, the field is 8 bits wide so this is considered a typo in the data sheet and the actual max is FFh (255). With an 80MHz VCLK, using the equation in the table yields a maximum delay of 3.2 uS. * The maximum delay of 3.2 uS is achieved by setting this parameter to 255. |
| bit 15-8 | 0 | **T2EDELAY:** Transmit-data-finished to ENA-pin-inactive time-out  ENA pin functionality is disabled, so this parameter has no effect. It is set arbitrarily set to 0. |
| bit 7-0 | 0 | **C2EDELAY:** Chip-select-active to ENA-signal-active time-out  ENA pin functionality is disabled, so this parameter has no effect. It is set arbitrarily set to 0. |

|  |  |  |
| --- | --- | --- |
| **MIBSPI 3 SPI Data Format 0 Register (SPIFMT0)** | | |
| Position | Desired Value | Description |
| bit 31-24 | 255 | **WDELAY:** Delay in between transmissions for data  WDELAY \* PVCLK + 2 \* PVCLK   * The inter-byte delay is accomplished via a combination of the CS activation and deactivation delays and this parameter. The other 2 delays are 3.212uS + 3.200uS = 6.412uS. Per [2] the FDD design desires to maximize the clock idle time between bytes using the hardware delays available. * Table 24-28 of [3] indicates the WDELAY has a max of 63, which is an error in the data sheet, 254 is the actual max reported by Eric Best, TI FAE. With an 80MHz VCLK, using the equation in the table yields a maximum delay of 3.2 uS. |
| bit 23 | 0 | **PARPOL:** Parity polarity: even or odd  Parity not enabled, so this has parameter has no effect. Arbitrarily se to 0. |
| bit 22 | 0 | **PARITYENA:** Parity enable for data  Per §8.5.3 of [2], Data Format specifies Parity to be disabled, so this parameter is set to 0. |
| bit 21 | 0 | **WAITENA:**  Per §8.5.3 of [2], Data Format specifies WAITENA to be disabled, so this parameter is set to 0. |
| bit 20 | 0 | **SHIFTDIR:** Shift direction for data  Per §8.5.3 of [2], Data Format specifies MSB shifted out first, so this parameter is set to 0. |
| bit 19 | 0 | **Reserved:** |
| bit 18 | 0 | **DIS CS TIMERS:** Disable chip-select timers for this format.  Delays specified by C2TDELAY and T2CDELAY are required for creating the inter byte delay specified in §8.5.3 of [2], so this is set to 0 to enable the delays. |
| bit 17 | 0 | **POLARITY:** SPI data clock polarity  §8.5.3 of [2], Clock settings specify the polarity to be Inactive-Low, so this is set to 0. (Note that the Active-High, requirement is not understood and is ignored in the configuration in this module. When the clock is active it is toggling states to provide edges, thus specifying the active state as a constant level has an unknown meaning) |
| bit 16 | 0 | **PHASE:** SPI data clock delay  §8.5.3 of [2], Clock settings specify the SCLK edges to be synchronized with the data stream, so this is set to 0. |
| bit 15-8 | 255 | **PRESCALE:** PRESCALE is use to derive SPICLK from VCLK  BRFormat = VBUSPCLK / (PRESCALEx ÷ 1)   * Turns Counter function in [2]§8.5.3 requires a clock source frequency of 300 kHz, however this is not an actual requirement as discussed with the FDD owner, but rather a documentation of the implementation choice at the time of FDD release. The physical maximum for SPI clock is specified in [1]Table 30-15 by SP71 and SP72 parameters. With the 8MHz PIC clock the time parameters = 1/(8MHz/4) + 20ns = 250ns + 20ns = 270 ns. This results in an overall period of 540ns = 1.85 MHz * Table 24-28 of [3] indicates the PRESCALE is 8 bits wide providing a max of 255. With an 80MHz VCLK, this results in a minimum of baud rate of approx. 314 baud. |
| bit 7-5 | 0 | **Reserved:** |
| bit 4-0 | 8 | **CHARLEN:** SPI data data-word length.  Per §8.5.3 of [2], Data Format specifies a value of 8 for this parameter. |

#### Module Outputs

None

#### Module Internal

#### 









## Periodic Functions

None

## Fault Recovery Functions

None

## Shutdown Functions

None

## Interrupt Functions

### Isr: \_IrqUnit2TxRx

#### Design Rationale

None

#### Program Flow Start

None

#### SpiNxt\_IrqUnit2TxRx



#### Program Flow End

None

### Isr: \_IrqUnit2TxRxERR

None

#### Program Flow Start

None

#### SpiNxt\_IrqUnit2TxRxERR



#### Program Flow End

None

## Serial Communication Functions

None

# Execution Requirements

## Execution Sequence of the Module

(Describe in words relevant details about the execution sequence of the different sub modules.)

## Execution Rates for sub-modules called by the Scheduler

This table serves as reference for the Scheduler design

|  |  |  |
| --- | --- | --- |
| Function Name | Calling Frequency | System State(s) in which the function is called |
| <None> |  |  |

## Execution Requirements for Serial Communication Functions

|  |  |
| --- | --- |
| Function Name | Sub-Module called by (Serial Comm Function Name) |
| <None> |  |

# Memory Map Definition Requirements

## Sub Modules (Functions)

This table identifies the software segments for functions identified in this module.

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| Spi\_SetupEB() | SPINXT\_START\_SEC\_CODE |
| Spi\_GetSequenceResult() | SPINXT\_START\_SEC\_CODE |
| Spi\_AsyncTransmit() | SPINXT\_START\_SEC\_CODE |
| Spi\_Init() | SPINXT\_START\_SEC\_CODE |
| mibspiSetData() | SPINXT\_START\_SEC\_CODE |
| mibspiSetCtrlData() | SPINXT\_START\_SEC\_CODE |
| mibspiGetData() | SPINXT\_START\_SEC\_CODE |
| mibspiTransfer() | SPINXT\_START\_SEC\_CODE |

## Local Functions

This table identifies the software segments for local functions identified in this module.

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| mibspiSetData8() | SPINXT\_START\_SEC\_CODE |
| mibspiGetData8() | SPINXT\_START\_SEC\_CODE |
| mibspiEnableGroupNotification() | SPINXT\_START\_SEC\_CODE |
| mibspiNotification() | SPINXT\_START\_SEC\_CODE |
| mibspiGroupNotification() | SPINXT\_START\_SEC\_CODE |

# Known Issues / Limitations With Design

1. Driver has several functions and all initializations hardcoded for what is needed by the Turns Counter , the Turns Counter Reflash, or the Digital MSB. Future modifications of the SPI interface of those components will probably require changes to this driver. New uses of the SPI driver would also require driver changes. Currently the only configurability is through the configuration parameter SpiNxt/SpiNxtGeneral/SpiNxtUseWith which selects either Turns Counter or Digital MSB functionality.
2. The Autosar API was not implemented for use by the Digital MSB for the following reasons:  
   a) The digital MSB triggers SPI transfers external to the SPI driver, through the NHET as set up in the ePWM component. This means the SpiNxt\_AsyncTransmit () function (if implemented for use by digital MSB) could not follow the Autosar requirement to initiate the transfer and set the driver/handler status accordingly.  
   b) The digital MSB transfers require chip select to be released after each word transferred. This means the four words would have to be defined as four channels/four jobs. Each channel requires its own buffer (internal or external). Because the SPI data reads and writes occur in the motor control ISR, they need to be very efficient and cannot afford the overhead of four calls to SpiNxt\_SetupEB() that would be needed to implement the double buffering being done on the read data. Use of a possible SpiNxt\_ReadIB() would have this same issue even without the double buffering.
3. Several of the initialization constants (delay fields in the delay registers and data format registers, and the baudrate prescale field in the data format registers) have values that depend on the VCLK frequency. These are currently hardcoded for 80 MHz VCLK for Digital MSB, and have values that will work for both 75MHz and 80MHz VCLK for Turns Counter / Turns Counter Reflash (see spreadsheet attached to Turns Counter Reflash FDD). A future improvement would be to calculate these values based on a VCLK configuration parameter.

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | 1 | Initial versiond |  | JJW |
| 2 | 2 | AsyncTransmit design improvement using exclusive areas and message transmit state check to ensure proper management of data transmission.  Added setting DIO CS channels to inactive during init  Added mibspi unit check around Error notification processing | 15-Mar-12 | JJW |
| 3 | 3 | Added Notification function hook for end of sequence notification | 23-Mar-12 | JJW |
| 4 | 4 | Increased WDELAY and T2CDELAY to meet updated FDD 20C requirements | 27-Mar-12 | JJW |
| 5 | 5 | Added metrics hooks to interrupt service routines. | 22-Mar-13 | BWL |
| 6 | 6 | Completed documentation of all functions in the module, along with changes required for the Digital MSB and the fix for anomaly 4713. | 05-Aug-13 | KMC |
| 7 | 7 | Removed conditionally-compiled mibspiSetData() functionality for digital MSB; modified conditionally-compiled mibspiGetData() functionality for digital MSB per ES-50A v005,changed initialization constant values, and added DMA control initialization, all per ES-50A v005 and fix for anomaly 5929. | 01-Apr-14 | KMC |