SPI™ Master Library Module (Polled)

1. Introduction

The SPIMPol is a general-purpose library module. It configures the MSSP/SSP/BSSP module in Master mode and helps in communicating with the SPI Slave and with the Microwire® Slave.

The module code is linkable and relocatable, which provides the user, the facility to use it without modifications.

By using this Module one can write his application to interact with any of the SPI or the Microwire Slaves like EEPROM, ADC, Digital Potentiometer, LCD etc.

The module allows user to concentrate more on his application's development by providing these library functions.

2. Module Features

It supports following features:-

- It provides simple and primitive functions to communicate with the SPI Slave.
- It generates Error flags on the occurrence of an error. All error conditions are passed through the 'W' Register.

3. List of Component Modules

`SPIMPol.P16.ex.txt`	This is an example file developed to demonstrate the use of the library functions for PIC16 family.
`SPIMPol.P18.ex.txt`	This is an example file developed to demonstrate the use of the library functions for PIC18 family.
`SPIMPol.asm`	This is the SPI Master code implementation file. One needs to include this file in their project.
`16SPIMP.asm`	This is the SPI Master code implementation file for PIC16 family. The `SPIMPol.asm` file will include this file if the PIC16 family processor is used.
`18SPIMP.asm`	This is the SPI Master code implementation file for PIC18 family. The `SPIMPol.asm` file will include this file if the PIC18 family processor is used.
`SPIMPol.inc`	This file contains the definitions of all the shared parameters and the macros. One needs to include this in the Assembly file where the library functions and macros are called. This file is taking care of definition of all Extern Global parameter so one can directly call library routines in their program.
`P16xxx.inc`	General purpose processor definition file for PIC16 family
`P18xxx.inc`	General purpose processor definition file for PIC18 family

4. Using the Library Module in a Project

Please follow the steps below to use this library module in your project.

1. Use the Application Maestro to configure the module as required.
2. At the 'Generate Files' step, save the output to the directory where your project code resides.
3. Launch MPLAB, and open the project's workspace.
4. Verify that the Microchip language tool suite is selected (*Project>Select Language Toolsuite*).
5. In the Workspace view, right-click on the "Source Files" node. Select the "Add Files" option. Select `SPIMPol.asm` and click **OK**.
6. Now right-click on the "Linker Scripts" node and select "Add Files". Add the appropriate linker file (`.lkr`) for the project's target microcontroller.
7. Add any other files that the project may require. Save and close the project.
8. In your main source (assembler) file, add include directive at the head of the code listing to include `SPIMPol.inc`. By doing so, all files required to make the generated code work in your project will be included by reference when you build the project.
9. To use the module in your application, invoke the functions or macros as needed.

5. List of Shared Parameters

Shared Functions

`SPIMPolInit`	It is used for Synchronous Serial Port Initialization It initializes Port according to the options opted through Application Maestro.
`SPIMPolPut`	It is used for transmitting a byte on SPI Bus.
`SPIMPolGet`	It is used for reading the received byte.
`SPIMPolIsTransmitOver`	It is used for checking/waiting for completion of transmission.
`SPIMPolIsDataReady`	It is used for checking/waiting for reception of data on SPI Bus.

Shared Macros

`mSPIMPolSetClockIdleState`	This sets the Idle state of the Clock line, ‘Hi’ (High) or ‘Lo’ (Low).
`mSPIMPolSetTransmitOnClockEdge`	This sets the Clock edge at which the data is to be Transmitted, ‘IdleToActive’ or ‘ActiveToIdle’.
`mSPIMPolSetSampleAtDataOut`	This sets at which phase of the Data Out, the Data In should be sampled, ‘Mids’ (Middle) or ‘Ends’ (End).
`mSPIMPolDisable`	Disables Synchronous Serial Port.

6. Functions

Function	`SPIMPolInit`
Preconditions	TRIS bits of SCK and SDO should be made output. TRIS bit of SDI should be made input. TRIS bit of Slave Chip Select pin (if any used) should be made output.
Overview	This function is used for initializing the MSSP/SSP/BSSP module. It initializes the module according to Application Maestro options.
Input	Application Maestro options
Output	None
Side Effects	Bank selection bits and 'W' register are changed
Stack Requirement	1 level deep

Function	`SPIMPolPut`
Preconditions	'`SPIMPolInit`' should have been called.
Overview	This function sends the byte in 'W' Reg. over SPI bus and checks for Write Collision.
Input	'W' Register.
Output	'W' Register. It will have: '0' - On proper initialization of transmission. '`SPIMErrWriteCollision`' - On occurrence of the Write Collision error.
Side Effects	Bank selection bits and 'W' register are changed
Stack Requirement	1 level deep

Function	`SPIMPolGet`
Preconditions	'`SPIMPolIsDataReady`' should return a '0'.
Overview	This function reads the byte received.
Input	None
Output	'W' Register.
Side Effects	Bank selection bits and 'W' register are changed
Stack Requirement	1 level deep

Function	`SPIMPolIsTransmitOver`
Preconditions	'`SPIMPolPut`' should have been called.
Overview	In Non Blocking Option – This function checks whether the transmission of the byte is completed. In Blocking Option – This function waits till the transmission of the byte is completed.
Input	None
Output	In Non Blocking Option – 'W' Register. It will have: '0' - If the transmission is over. '`SPIMTransmitNotOver`' - If the transmission is not yet over. In Blocking Option – None
Side Effects	Bank selection bits and 'W' register are changed
Stack Requirement	1 level deep

Function	`SPIMPolIsDataReady`
Preconditions	'`SPIMPolPut`' should have been called. '`SPIMPolPut`' initiates the reception also.
Overview	In Non Blocking Option – This checks weather the Data is received. It also checks for the Over flow error. In Blocking Option – It checks for the Over flow error. If there is no error waits till Data is ready.
Input	None
Output	'W' Register. It will have: In Non Blocking Option – '0' - If the Data is ready '`SPIMDataNotReady`' - If Data is not ready In Blocking Option – None
Side Effects	Bank selection bits and 'W' register are changed
Stack Requirement	1 level deep

7. Macros

Macro	`mSPIMPolSetClockIdleState`
Overview	This Macro is used to specify the Idle State of the Clock pin (SCK).
Input	The Clock pin Idle state: '**`Hi`**' (for High) '**`Lo`**' (for Low) Example- To set Clock pin Idle State High, `mSPIMPolSetClockIdleState Hi`
Output	None
Side Effects	Bank selection bits are changed.
Stack Requirement	None

Macro	`mSPIMPolSetTransmitOnClockEdge`
Overview	This Macro is used to specify on what edge of the Clock the transmission should take place.
Input	Transmission at the clock edge: '**`IdleToActive`**' '**`ActiveToIdle`**' Example- To transmit on Clock edge Idle to Active `mSPIMPolSetTransmitOnClockEdge IdleToActive`
Output	None
Side Effects	Bank selection bits are changed.
Stack Requirement	None

Macro	`mSPIMPolSetSampleAtDataOut`
Overview	This Macro is used to specify the sampling phase of the Data In, with respect to Data Out.
Input	Sampling phase with respect to Data Out: '**`Mids`**' (Middle) '**`Ends`**' (End). Example- To sample at mid of Data Out `mSPIMPolSetSampleAtDataOut Mids`
Output	None
Side Effects	Bank selection bits are changed.
Stack Requirement	None

Macro	`mSPIMPolDisable`
Overview	Disables the MSSP/SSP/BSSP module.
Input	None
Output	None
Side Effects	Bank selection bits are changed.
Stack Requirement	None

8. Error and Status Flags

All errors/status are set as a content of ‘W’ Register. Individual errors/status are unique. Please refer below list for the information.

`SPIMErrWriteCollision`	This indicates that, Write collision has occurred while trying to transmit the byte.
`SPIMTransmitNotOver`	This indicates that, the transmission is not yet over. This is to be checked only when Non Blocking option is opted.
`SPIMDataNotReady`	This indicates that, the Data is not yet fully received. This is to be checked only when Non Blocking option is opted.