dsPIC30F/dsPIC33F/dsPIC33E/PIC24H/PIC24E
I^2C™ Peripheral Module Library Help

Table Of Contents

1 Library Features

This peripheral library module provides:

- Module initialization functions for both master and slave modes
- Simple functions to read/write one or multiple bytes
- Functions to read registers and configure interrupts
- Simple interface macros to enable/disable interrupts

2 Using the Library Functions in Your Code

Library routine parameters can be constructed using an AND-based mask. For more information on this mask, see 16-bit Peripheral Libraries. An example of use is shown below.

dsPIC30F Example of Use (AND mask)

```
#include<i2c.h>

void main(void )
{
  unsigned int config2, config1;
  unsigned char *wrptr;
  unsigned char tx_data[] = {'M','I','C','R','O','C','H','I','P','\0'};
  wrptr = tx_data;
  /* Baud rate is set for 100 kHz */
  config2 = 0x11;
  /* Configure I2C for 7 bit address mode */
  config1 = (I2C_ON & I2C_IDLE_CON & I2C_CLK_HLD &
             I2C_IPMI_DIS & I2C_7BIT_ADD &
             I2C_SLW_DIS & I2C_SM_DIS &
             I2C_GCALL_DIS & I2C_STR_DIS &
             I2C_NACK & I2C_ACK_DIS & I2C_RCV_DIS &
             I2C_STOP_DIS & I2C_RESTART_DIS &
             I2C_START_DIS);
  OpenI2C(config1,config2);
  IdleI2C();
  StartI2C();
  /* Wait till Start sequence is completed */
  while(I2CCONbits.SEN);
  /* Clear interrupt flag */
  IFS0bits.MI2CIF = 0;
  /* Write Slave address and set master for transmission */
  MasterWriteI2C(0xE);
  /* Wait till address is transmitted */
  while(I2CSTATbits.TBF);  // 8 clock cycles
  while(!IFS0bits.MI2CIF); // Wait for 9th clock cycle
  IFS0bits.MI2CIF = 0;     // Clear interrupt flag
  while(I2CSTATbits.ACKSTAT);
  /* Transmit string of data */
  MasterputsI2C(wrptr);
  StopI2C();
```

```
  /* Wait till stop sequence is completed */
  while(I2CCONbits.PEN);
  CloseI2C();
}
```

3 Functions

3.1 AckI2C

Function Prototype	`void AckI2C(void);`
Include	`i2c.h`
Description	Generates I2C bus Acknowledge condition.
Arguments	None
Return Value	None
Remarks	This function generates an I2C bus Acknowledge condition.
Source File	`AckI2C.c`
Code Example	`AckI2C();`

3.2 CloseI2C

Function Prototype	`void CloseI2C(void);`
Include	`i2c.h`
Description	This function turns off the I2C module.
Arguments	None
Return Value	None
Remarks	This function disables the I2C module and clears the Master and Slave Interrupt Enable and Flag bits.
Source File	`CloseI2C.c`
Code Example	`CloseI2C();`

3.3 ConfigIntI2C

Function Prototype	`void ConfigIntI2C(unsigned int` *`config`*`);`
Include	`i2c.h`
Description	This function configures the I2C interrupt.

Arguments	*`config`* - I2C interrupt priority and enable/disable information as defined below:
	I2C master Interrupt enable/disable
	`MI2C_INT_ON`
	`MI2C_INT_OFF`
	I2C slave Interrupt enable/disable
	`SI2C_INT_ON`
	`SI2C_INT_OFF`
	I2C master Interrupt priority
	`MI2C_INT_PRI_7`
	`MI2C_INT_PRI_6`
	`MI2C_INT_PRI_5`
	`MI2C_INT_PRI_4`
	`MI2C_INT_PRI_3`
	`MI2C_INT_PRI_2`

```
   MI2C_INT_PRI_1
   MI2C_INT_PRI_0
```

I2C slave Interrupt priority

```
   SI2C_INT_PRI_7
   SI2C_INT_PRI_6
   SI2C_INT_PRI_5
   SI2C_INT_PRI_4
   SI2C_INT_PRI_3
   SI2C_INT_PRI_2
   SI2C_INT_PRI_1
   SI2C_INT_PRI_0
```

Return Value	None.
Remarks	This function clears the Interrupt Flag bits, sets the interrupt priorities of master and slave and enables/disables the interrupt.
Source File	`ConfigIntI2C.c`
Code Example	`ConfigIntI2C(MI2C_INT_ON & MI2C_INT_PRI_3 & SI2C_INT_ON & SI2C_INT_PRI_5);`

3.4 DataRdyI2C

Function Prototype	`unsigned char DataRdyI2C(void);`
Include	`i2c.h`
Description	This function provides status back to the user if the I2CRCV register contains data.
Arguments	None
Return Value	This function returns '`1`' if there is data in I2CRCV register; else return '`0`' which indicates no data in I2CRCV register.
Remarks	This function determines if there are any bytes to read from the I2CRCV register.
Source File	`DataRdyI2C.c`
Code Example	`if(DataRdyI2C());`

3.5 IdleI2C

Function Prototype	`void IdleI2C(void);`
Include	`i2c.h`
Description	This function generates Wait condition until I2C bus is Idle.
Arguments	None
Return Value	None
Remarks	This function will be in a wait state until Start Condition Enable bit, Stop Condition Enable bit, Receive Enable bit, Acknowledge Sequence Enable bit of I2C Control register and Transmit Status bit I2C Status register are clear. The `IdleI2C` function is required since the hardware I2Cperipheral does not allow for spooling of bus sequence. The I2C peripheral must be in Idle state before an I2C operation can be initiated or write collision will be generated.
Source File	`IdleI2C.c`
Code Example	`IdleI2C();`

3.6 MastergetsI2C

Function	`unsigned int MastergetsI2C(unsigned int` *`length`*`,`

Prototype	unsigned char *rdptr, unsigned int i2c_data_wait);
Include	i2c.h
Description	This function reads predetermined data string length from the I2C bus.
Arguments	*length* - Number of bytes to read from I2C device. *rdptr* -Character type pointer to RAM for storage of data read from I2C device *i2c_data_wait* - This is the time-out count for which the module has to wait before return. If the time-out count is 'N', the actual time out would be about (20 * N – 1) instruction cycles.
Return Value	This function returns '0' if all bytes have been sent or number of bytes read from I2C bus if its not able to read the data within the specified *i2c_data_wait* time out value
Remarks	This routine reads a predefined data string from the I2C bus.
Source File	MastergetsI2C.c

Code Example	unsigned char string[10];
	unsigned char *rdptr;
	unsigned int length, i2c_data_wait;
	length = 9;
	rdptr = string;
	i2c_data_wait = 152;
	MastergetsI2C(length, rdptr, i2c_data_wait);

3.7 MasterputsI2C

Function Prototype	unsigned int MasterputsI2C(unsigned char *wrptr);
Include	i2c.h
Description	This function is used to write out a data string to the I2C bus. .
Arguments	*wrptr* - Character type pointer to data objects in RAM. The data objects are written to the I2C device.
Return Value	This function returns '-3' if a write collision occurred. This function returns '0' if the null character was reached in data string.
Remarks	This function writes a string to the I2C bus until a null character is reached. Each byte is written via a call to the MasterputcI2C function. The actual called function body is termed MasterWriteI2C. MasterWriteI2C and MasterputcI2C refer to the same function via a #define statement in the i2c.h
Source File	MasterputsI2C.c
Code Example	unsigned char string[] = " MICROCHIP "; unsigned char *wrptr; wrptr = string; MasterputsI2C(wrptr);

3.8 MasterReadI2C

Function Prototype	unsigned char MasterReadI2C(void);
Include	i2c.h
Description	This function is used to read a single byte from I2C bus.
Arguments	None
Return Value	The return value is the data byte read from the I2C bus.
Remarks	This function reads in a single byte from the I2C bus. This function performs the same function as MastergetcI2C.
Source File	MasterReadI2C.c

Code Example	`unsigned char value;` `value = MasterReadI2C();`

3.9 MasterWriteI2C

Function Prototype	`unsigned char MasterWriteI2C(unsigned char data_out);`
Include	`i2c.h`
Description	This function is used to write out a single data byte to the I^{2}C device.
Arguments	*`data_out`* A single data byte to be written to the I^{2}C bus device.
Return Value	This function returns '`-1`' if there was a write collision else it returns a '`0`'.
Remarks	This function writes out a single data byte to the I^{2}C bus device. This function performs the same function as `MasterputcI2C`.
Source File	`MasterWriteI2C.c`
Code Example	`MasterWriteI2C('a');`

3.10 NotAckI2C

Function Prototype	`void NotAckI2C(void);`
Include	`i2c.h`
Description	Generates I^{2}C bus Not Acknowledge condition.
Arguments	None
Return Value	None
Remarks	This function generates an I^{2}C bus *Not Acknowledge* condition.
Source File	`NotAckI2C.c`
Code Example	`NotAckI2C();`

3.11 OpenI2C

Function Prototype	`void OpenI2C(unsigned int config1,` `unsigned int config2);`
Include	`i2c.h`
Description	Configures the I^{2}C module.

Arguments	*`config1`* - This contains the parameter to configure the I2CCON register.
	I^{2}C Enable bit
	`I2C_ON`
	`I2C_OFF`
	I^{2}C Stop in Idle Mode bit
	`I2C_IDLE_STOP`
	`I2C_IDLE_CON`
	SCL Release Control bit
	`I2C_CLK_REL`
	`I2C_CLK_HLD`
	Intelligent Peripheral Management Interface Enable bit
	`I2C_IPMI_EN`
	`I2C_IPMI_DIS`
	10-bit Slave Address bit
	`I2C_10BIT_ADD`
	`I2C_7BIT_ADD`
	Disable Slew Rate Control bit

```
I2C_SLW_DIS
I2C_SLW_EN
```
SMBus Input Level bits
```
I2C_SM_EN
I2C_SM_DIS
```
General Call Enable bit
```
I2C_GCALL_EN
I2C_GCALL_DIS
```
SCL Clock Stretch Enable bit
```
I2C_STR_EN
I2C_STR_DIS
```
Acknowledge Data bit
```
I2C_ACK
I2C_NACK
```
Acknowledge Sequence Enable bit
```
I2C_ACK_EN
I2C_ACK_DIS
```
Receive Enable bit
```
I2C_RCV_EN
I2C_RCV_DIS
```
Stop Condition Enable bit
```
I2C_STOP_EN
I2C_STOP_DIS
```
Repeated Start Condition Enable bit
```
I2C_RESTART_EN
I2C_RESTART_DIS
```
Start Condition Enable bit
```
I2C_START_EN
I2C_START_DIS
```

config2 - computed value for the baud rate generator

Return Value	None
Remarks	This function configures the I^2C Control register and I^2C Baud Rate Generator register.
Source File	`OpenI2C.c`
Code Example	`OpenI2C(I2C_ON & I2C_IDLE_IDLE_CON);`

3.12 RestartI2C

Function Prototype	`void RestartI2C(void);`
Include	`i2c.h`
Description	Generates I^2C bus Not Acknowledge condition.
Arguments	None
Return Value	None
Remarks	This function generates an I^2C Bus Restart condition.
Source File	`RestartI2C.c`
Code Example	`RestartI2C();`

3.13 SlavegetsI2C

Function Prototype	`unsigned int SlavegetsI2C(unsigned char *rdptr, unsigned int i2c_data_wait);`

Include	`i2c.h`
Description	This function reads pre-determined data string length from the I²C bus.
Arguments	*`rdptr`* - Character type pointer to RAM for storage of data read from the I²C device. *`i2c_data_wait`* - This is the time-out count for which the module has to wait before return. If the time-out count is 'N', the actual time out would be about (20*N - 1) instruction cycles.
Return Value	Returns the number of bytes received from the I²C bus.
Remarks	This routine reads a predefined data string from the I²C bus.
Source File	`SlavegetsI2C.c`

Code Example	`unsigned char string[12];`
	`unsigned char *rdptr;`
	`rdptr = string;`
	`i2c_data_out = 0x11;`
	`SlavegetsI2C(rdptr, i2c_data_wait);`

3.14 SlaveputsI2C

Function Prototype	`unsigned int SlaveputsI2C(unsigned char *wrptr);`
Include	`i2c.h`
Description	This function is used to write out a data string to the I²C bus.
Arguments	*`wrptr`* - Character type pointer to data objects in RAM. The data objects are written to the I²C device.
Return Value	This function returns '0' if the null character was reached in the data string.
Remarks	This routine reads a predefined data string from the I²C bus.
Source File	`SlavegetsI2C.c`

Code Example	`unsigned char string[] =”MICROCHIP”;`
	`unsigned char *rdptr;`
	`rdptr = string;`
	`SlaveputsI2C(rdptr);`

3.15 SlaveReadI2C

Function Prototype	`unsigned char SlaveReadI2C(void);`
Include	`i2c.h`
Description	This function is used to read a single byte from the I²C bus.
Arguments	None
Return Value	The return value is the data byte read from the I²C bus.
Remarks	This function reads in a single byte from the I²C bus. This function performs the same function as `SlavegetcI2C`.
Source File	`SlaveReadI2C.c`
Code Example	`unsigned char value;` `value = SlaveReadI2C();`

3.16 SlaveWriteI2C

Function Prototype	`void SlaveWriteI2C(unsigned char data_out);`
Include	`i2c.h`
Description	This function is used to write out a single byte to the I²C bus.

Arguments	*data_out* - A single data byte to be written to the I2C bus device.
Return Value	None
Remarks	This function writes out a single data byte to the I2C bus device. This function performs the same function as SlaveputcI2C.
Source File	SlaveWriteI2C.c
Code Example	SlaveWriteI2C('a');

3.17 StartI2C

Function Prototype	void StartI2C(void);
Include	i2c.h
Description	Generates I2C Bus Start condition.
Arguments	None
Return Value	None
Remarks	This function generates a I2C Bus Start condition.
Source File	StartI2C.c
Code Example	StartI2C();

3.18 StopI2C

Function Prototype	void StopI2C(void);
Include	i2c.h
Description	Generates I2C Bus Stop condition.
Arguments	None
Return Value	None
Remarks	This function generates a I2C Bus Stop condition.
Source File	StopI2C.c
Code Example	StopI2C();

4 Macros

4.1 EnableIntMI2C

Macro	EnableIntMI2C
Overview	This macro sets Master I2C Enable bit of Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	EnableIntMI2C;

4.2 DiableIntMI2C

Macro	DiableIntMI2C
Overview	This macro clears Master I2C Interrupt Enable bit of Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	DisableIntMI2C;

4.3 SetPriorityIntMI2C

Macro	`SetPriorityIntMI2C`
Overview	This macro sets Master I^{2}C Interrupt Priority bits of Interrupt Priority Control register.
Input	Interrupt Priority Level.
Output	None
Remarks	None
Code Example	`SetPriorityIntMI2C(7);`

4.4 EnableIntSI2C

Macro	`EnableIntSI2C`
Overview	This macro sets Slave I^{2}C Enable bit of Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	`EnableIntSI2C;`

4.5 DisableIntSI2C

Macro	`DisableIntSI2C`
Overview	This macro clears Slave I^{2}C Enable bit of Interrupt Enable Control register.
Input	None
Output	None
Remarks	None
Code Example	`DisableIntSI2C;`

4.6 SetPriorityIntSI2C

Macro	`SetPriorityIntSI2C`
Overview	This macro sets Slave I^{2}C Interrupt Priority bits of Interrupt Priority Control register.
Input	Interrupt Priority Level.
Output	None
Remarks	None
Code Example	`SetPriorityIntSI2C(7);`