MSP430 SMBus Library for MSP430FR5xx_6xx Devices

User's Guide

DOCNUM-1.00.00.00

Copyright

Texas Instruments
13532 N. Central Expressway MS3810
Dallas, TX 75243
www.ti.com/

Revision Information

This is version 1.00.00.00 of this document, last updated on Thu Jan 8 2015 17:27:35.

Table of Contents

1 Introduction

1.1 Introduction

Ths SMBus (Sytem Management Bus) API (application programming interface) stack for MSP430 is a turnkey API. It includes support for applications where the MSP430 is acting as the master or a slave.

The API is designed to minimize the SMBus knowledge required to write an application:

- All SMBus protocol is handled automatically by the API
- The data interface presented to the application is simple to use, abstracting the application from SMBus protocol

The user should not need to modify the API source. However, for experienced developers, the source is open and available for editing. Accessing the API's source can also be useful for system debug.

Application examples are included in the MSP430 SMBus Library Package.

1.2 SMBus

The System Management Bus (SMBus) is a lightweight two-wire interface based on the principles of I2C, commonly used as a control bus and for power management tasks in computing, mobile computing and battery operated applications. A device performing data transfers on the bus can be considered a master, which is the device which initiates a transaction and drives the clock, or a slave, which is the target of a SMBus transaction driven by the master. Both the master and the slave can act as transmitters or as receivers.

SMBus 2.0 shares a lot of similarities with I2C, but some of the most relevant differences include:

- Time-out detection when a device stretches the clock for too long
- Packet Error Checking (PEC) can be optionally appended at the end of each transaction, allowing the bus to automatically validate packets
- I2C only defines a PHY and Data-Link layers, but SMBus defines a network layer with different SMBus protocols which can be used to exchange data between devices
- Optional use of additional lines such as SMBAlert# and SMBSUS#.

For more information about SMBus, please refer to the SMBus 2.0 specification: `http://smbus.org/specs/`. For more information about I2C, please refer to `http://www.nxp.com/documents/user_manual/UM10204.pdf`

1.2.1 Supported Features

The following table details the SMBus features supported in the MSP430 SMBus library.

SMBus Feature	MSP430FR5xx_6xx		MSP430G2xx3		Applicable SMBus 2.0
	Master	Slave	Master	Slave	specification section
Clock Timeout detection[1]	Yes	Yes	Yes[1]	Yes[1]	3.1.1.2
Manual NACK on invalid address/data	N/A	Yes[2]	N/A	No	4.2
Arbitration	Yes	N/A	Yes	N/A	4.3.2
Clock stretching	Yes	Yes	Yes	Yes	4.3.3
Multiple Slave address	N/A	No	N/A	No	5.2
General Call address	N/A	No	N/A	No	5.2
PEC	Yes	Yes	Yes	Yes	5.4
Quick Command	Partial[3]	Partial[3]	Parial[3]	Partial[3]	5.5.1
Send Byte	Yes	Yes	Yes	Yes	5.5.2
Receive Byte	Yes	Yes	Yes	Yes	5.5.3
Write Byte/Word	Yes	Yes	Yes	Yes	5.5.4
Read Byte/Word	Yes	Yes	Yes	Yes	5.5.5
Process Call	Yes	Yes	Yes	Yes	5.5.6
Block Write/Read	Yes	Yes	Yes	Yes	5.5.7
Block write-block read process call	Yes	Yes	Yes	Yes	5.5.8
Host notify protocol	No	No	No	No	5.5.9
ARP	No	No	No	No	5.6
SMBAlert#	No	No	No	No	Appendix A
SMBSUS#	No	No	No	No	Appendix A

Notes:

1. MSP430G2xx3 USCI doesn't have integrated timeout detection and requires a timer
2. MSP430FR5x/6xx can optionally- use DMA to stretch SCL while deciding to ACK/NACK the current byte
3. Only QuickCommand Write is supported, not QuickCommand Read

1.3 The MSP430 SMBus Library Package

This User's Guide documents the SMBus API and examples. The contents of the package is described below:

- smbuslib: Top level directory. Contains release notes and the manifest file related to licensing.
  - docs: Contains the API and User's Guides for the supported MSP430 devices
  - driverlib: Contains the standard MSP430 driverlib which is used in the library implementation for device families that support driverlib (for example, MSP430FR5xx_6xx)
  - examples: Contains the example projects for each of the supported MSP430 devices
  - src: Contains the source code for the SMBus Library stack

2 Introduction to the MSP430 SMBus API

2.1 Overview

The MSP430 SMBus API stack allows easy creation of MSP430 applications that communicate with other system components over a SMBus interface.

This API supports using the MSP430 as the SMBus master or as a SMBus slave.

2.2 Supported Device Families

The SMBus API stack is supported on the following MSP430 device families:

- MSP430FR5xx_6xx
- MSP430G2xx3

2.3 Supported Development Environments

The SMBus API stack and examples build and run on both the IAR and CCS environments for MSP430. See the Release Notes HTML file in the SMBus Library Package for specific IAR/CCS version information.

IAR and CCS are both available in free, code-size-limited versions (8K and 16K, respectively, of object code). Applications that fit under 8K of memory can be run on both free versions. Applications that are greater than 8K cannot be built using the free IAR Kickstart tool. Instead, the free version of CCS can be used; or a licensed version of either environment.

See the Release Notes within the SMBus Library Package zip file for additional information specific to a given release.

2.4 Stack Organization

The software stack is organized into three layers:

- The public API layer defines the API's that should be called by application programs. These functions are defined in smbus.h.
- The network layer manages the SMBus protocol state machine and interfaces with the physical layer. These functions are defined in smbus_nwk.h and should not be called directly from application programs.
- The physical layer contain all the device specific code to interact with the MSP430 hardware. These functions are defined in smbus_phy.h and should not be called directly from application programs.

2.5 Usage of MCU Peripheral Resources

Within the SMBus API, the resources shown below are considered owned by the API. If the application accesses them, it should be aware of how the API uses them.

Resource	Owned by API When	How it's Used
eUSCIB0	Always	I2C communication
DMA0	When "MANUAL_ACK_ENABLE" is defined	This is an opitonal feature. See smbus.h for documentation.

2.6 Release Notes and Migration from Previous Versions

A Release Notes HTML file accompanies each release of the SMBus Library Package. Reference this file for any information specific to this release, including:

- All changes from the previous versions
- Instructions for migration from previous versions
- Updated IDE configuration information
- Known issues

3 SMBus API Usage

3.1 Introduction

This chapter contains the detailed documentation for the application API functions and descriptions on using the API to create a SMBus master or slave application.

3.2 Usage

This section illustrates the basic application template for master and slave applications. See the examples for complete applications, and the `HTML API documentation` for details on the individual APIs.

3.2.1 Master Usage Outline

```
// Declare master SMBus structure
SMBus SMB;

// Initialize GPIOs and clocks
...
// Initialize GPIO I2C pins
...

// Initialize SMBus Master using eUSCI_B0 always at 100kbps per SMBus spec
SMBus_masterInit(&SMB, EUSCI_B0_BASE, (MCLK_MHZ*1000000));

// Initialize I2C and enable SMBus Interrupts
SMBus_masterEnableInt(&SMB);

// Send SMBus Sendbyte command (0x33)
uint8_t ret = SMBus_masterSendByte(&SMB,  // SMB struct
                                   0x40,  // Slave Addr
                                   0x33); // SMB Command
...
```

3.2.2 Slave Usage Outline

```
main()
{
    // Declare slave SMBus structure
    SMBus SMB;

    // SMBus receive and transmit buffers
    uint8_t au8RxBuff[SMB_MAX_PACKET_SIZE];
    uint8_t au8TxBuff[SMB_MAX_PACKET_SIZE];

    // Initialize GPIOs and clocks
    ...
    // Initialize GPIO I2C pins
    ...

    // Initialize SMBus Slave using eUSCI_B0
    SMBus_slaveInit(&SMB, EUSCI_B0_BASE);

    // Set the slave's address
    SMBus_slaveSetAddress(&SMB, 0x40);

    // Set the RX and TX buffers for SMBus
    SMBus_slaveSetRxBuffer(&SMB, au8RxBuff, sizeof(au8RxBuff));
```

```
    SMBus_slaveSetTxBuffer(&SMB, au8TxBuff, sizeof(au8TxBuff));

    // Initialize I2C and enable SMBus Interrupts
    SMBus_slaveEnableInt(&SMB);

    ...
    while (1)
    {
        __disable_interrupt();
        {
            _BIS_SR(LPM3_bits+GIE); // Go to sleep
        }
        __enable_interrupt();
    } // While (1)

}
#pragma vector=USCI_B0_VECTOR
__interrupt void eUSCI_ISR (void)
{
    // Check the state of SMBus
    switch (SMBus_slaveProcessInt(&SMB))
    {
        case SMBus_State_Slave_QCMD:
            // If a Quick command was detected, execute function (if any)
        break;
        case SMBus_State_Slave_CmdComplete:
            // Get command using SMBus_slaveGetCommand(&SMB) and process command

            // if command is not valid/supported
            //    SMBus_slaveReportError(&SMB, SMBUS_ErrorCode_Cmd);

            LPM3_EXIT;      // Exit to main loop if required
        break;
        default:
        break;
    }
    // Clear flags to be ready for next packet
    SMBus_processDone(&SMB);
}
```

3.3 Examples

Several examples are provided with the release package that illustrate using the library to implement both SMBus master and slave application.

For each example, matching master and slave implementations are provided.

- ReadByte_Echo - sends and echoes back a ReadByte command
- WriteWord_Dimmer - sends and echoes back a WriteWord command
- AllProtocols - sequences through all the SMBus protocols

Examples are configured for the MSP-EXP430G2 and MSP-EXP430FR5969 launch pad boards and CCS and IAR projects are provided.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265