Application Report
SLAA383–December 2007

Using the USCI I²C Slave

Uli Kretzschmar — *MSP430 Systems*
Christian Hernitscheck — *MSP430 Application Europe*

ABSTRACT

This document is an overview of the use of the I²C slave function set for MSP430 devices with the USCI module. The functions provided in the package can be used for MSP430 slave devices performing I²C communication and can handle both transmit and receive requests from I²C masters.

Note: The USCI I²C slave package includes a demonstration application that can be used on any MSP430 2xx device with the USCI module.

Contents

1 Introduction 2
2 Usage From C 3
2.1 Initialization 3
3 Compiling the USCI I²C Slave Code 4
4 Included Files 4
4.1 Function Description 4
5 Code Size 5
6 References 5

All trademarks are the property of their respective owners.

1 Introduction

When using the MSP430 with peripheral modules, I²C is often used for communication. There are several MSP430 devices that have an USCI module, which is capable of this communication protocol.

The USCI I²C slave function set offers some functions that make I²C communication easy. Instead of having to configure the different registers of the UCSI module, the user can use the included functions with well-defined parameters to start communication. These functions serve only for setting up the USCI module, and the user is free to include low-power mode functionality to allow the CPU to be turned off at the application level or to continue calculations during I²C communication. Transmitted and received bytes are managed using callback functions.

The USCI I²C slave package includes functions that support both transmit and receive operations:

- Slave receive (the slave is addressed by the master and receives data from it)

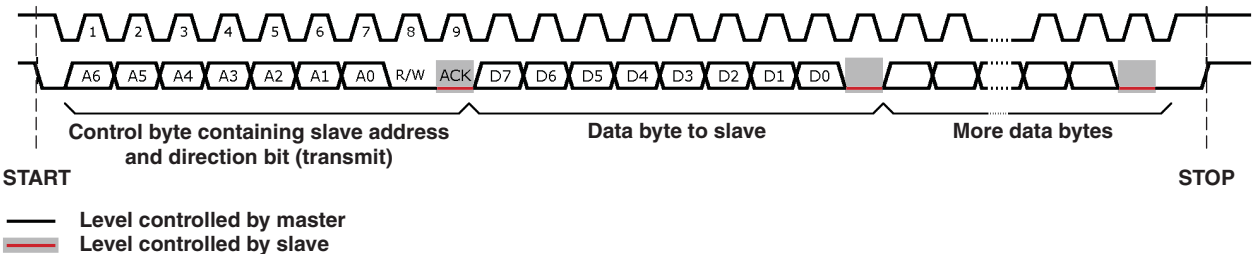

Figure 1. Slave Receive

- Slave transmit (the slave is addressed by the master and transmits data to it)

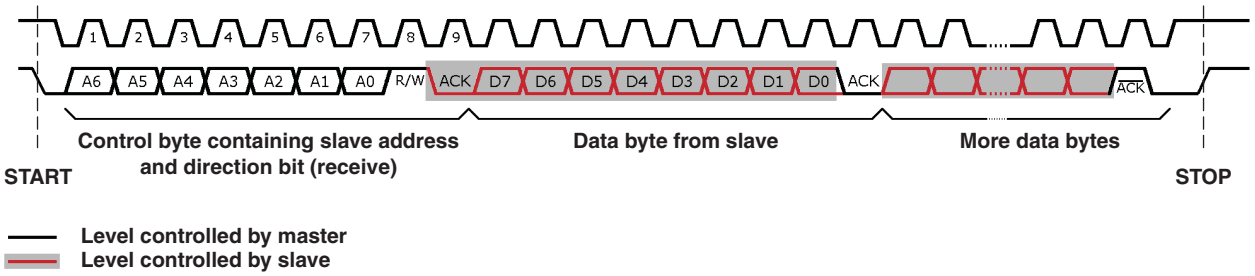

Figure 2. Slave Transmit

Both of these functions support only 7-bit addressing.

2 Usage From C

```
#include "msp430x26x.h"
#include "TI_USCI_I2C_slave.h"

void receive_cb(unsigned char receive);
void transmit_cb(unsigned char volatile *receive);
void start_cb();

unsigned char TXData = 0 ;
unsigned char RXData = 0 ;

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                          // Stop WDT

  TI_USCI_I2C_slaveinit(start_cb, transmit_cb, receive_cb, 0x50);  // init the slave
  _EINT();
  BCSCTL1 = CALBC1_16MHZ;
  DCOCTL = CALDCO_16MHZ;
  LPM0;                                              // Enter LPM0.
}

void start_cb(){
  RXData = 0;
}

void receive_cb(unsigned char receive){
  RXData = receive;
}

void transmit_cb(unsigned char volatile *byte){
  *byte = TXData++;
}
```

The file TI_USCI_I2C_slave.c must be added to the project, and the line `#include "TI_USCI_I2C_slave.h"` must be included in the user source code to access to the slave functions.

The slave program TI_USCI_I2C_slave.c runs on a slave MSP430 that is connected to an MSP430 master running the program TI_USCI_I2C_master.c. [4]

This short program initializes a MSP430F2618 as I^2C slave device with the address 0x50. For each data read request from a master, it returns the last byte the master wrote to it, or it returns a zero if the first transaction is a read.

Note: The slave demonstration applications were developed for use with the 2xx family. However, they can be easily modified for use with any MSP430 device with the USCI module.

2.1 Initialization

As shown in the previous example, configuring the device in slave mode requires calling TI_USCI_I2C_slaveinit once.

The first parameter is a function pointer, which points to a callback function that is executed each time a Start condition occurs and the slave is addressed. This callback function's return type must be void, and its parameter must be empty.

The second parameter is a callback function for values that are to be transmitted to the master. This callback function's return type must be void, and its parameter list must contain one char pointer as its only parameter. This callback function is called for every byte the master requests from the slave.

The third parameter is a callback function that handles the receiving of data from I²C master devices. For each byte received, the callback is called, and the received byte is available trough the parameter *receive*.

The last parameter is the slave address of the MSP430 that acts as a slave.

Note: Receive and transmit callbacks are not called corresponding to the number of bytes actually transmitted and received, but to the number of times the corresponding interrupts occur. In the case of a slave transmit, the transmit callback is called once more than the actual number of transmitted bytes. In the case of heavy CPU loads by different ISRs, the last redundant call of the transmit callback might not occur. Therefore, the data for the communication should be prepared within the Start callback to keep track of the number of frames requested by the master.

3 Compiling the USCI I²C Slave Code

This USCI I²C slave package is distributed as source code and is intended to be compiled with a project. To accomplish this:

- Add TI_USCI_I2C_slave.c to the project.
- Include the necessary header definitions by adding `#include "TI_USCI_I2C_slave.h"` to the user source code.
- Change the MSP430 device-specific include file (MSP430 standard header file) in TI_USCI_I2C_slave.c.
- Adjust the definitions of SDA_PIN and SCL_PIN in TI_USCI_I2C_slave.h.

4 Included Files

TI_USCI_I2C_slave.c	This file contains all functions necessary to perform I²C slave communication using the USCI module of the MSP430.
TI_USCI_I2C_slave.h	This file includes the definitions of the functions and variables that are used in TI_USCI_I2C_slave.c. It also contains the precompiler variables SDA_PIN and SCL_PIN that define which pins of the MSP430 are used for I²C communication. This file must be included in any C program that calls the slave function set.

4.1 Function Description

- **void TI_USCI_I2C_slaveinit(void (*SCallback)(), void (*TCallback)(unsigned char volatile *value), void (*RCallback)(unsigned char value), unsigned char slave_address)**

  This function initializes the USCI module for I²C slave operation. It has the following parameters:
  - **void (*SCallback)()**
    This is the function that is called when a Start condition is detected and the slave is addressed.
  - **void (*TCallback)(unsigned char volatile *value)**
    This is the function that is called for every byte requested by a master. The byte that is to be sent to the master has to be written to value.
  - **void (*RCallback)(unsigned char value)**
    This is the function that is called for every byte that is received from a master. The actual received value is available through the variable *value*.
  - **unsigned char slave_address**
    This parameter is used to set the slave address of the USCI module.

5 Code Size

Table 1. Code Size (IAR)

Function	Size (Bytes)
Slave initialize	120

6 References

1. *MSP430x2xx Family User's Guide* (SLAU144)
2. MSP430x261x data sheet (SLAS541)
3. *I²C-Bus Specification and User Manual*, NXP Semiconductors, 2007 (http://www.nxp.com/acrobat/usermanuals/UM10204_3.pdf)
4. *Using the USCI I²C Master* (SLAA382)

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
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2007, Texas Instruments Incorporated