

Software I²C on MSP430 MCUs

Timothy Logan

MSP Applications

ABSTRACT

Hardware and layout limitations often require smaller package sizes that could potentially create a tradeoff for the number of serial peripherals available to the user. With the I²C protocol, special software can be created to use a pair of simple GPIO ports to emulate an I²C master device. This allows programmers to be flexible with pin assignments and enables smaller package devices to overcome the hardware limitation of scarce number of hardware I²C peripherals. Having a software-emulated I²C master enables programmers to communicate effectively and fully control attached slave devices without excessive overhead. In addition to the basic I²C functionality, several advanced features such as clock stretching and NAK detection are also supported by the software included in this application report.

Source code and other collateral discussed in this application report can be downloaded from <http://www.ti.com/lit/zip/sl原因703>.

Contents

1	Introduction	2
2	Use Cases	2
3	Configuration	3
4	Simple Transactions	3
5	Advanced Features	5

List of Figures

1	Sample Hardware Configuration	2
2	Pin Configuration	3
3	Formula to Derive I ² C Clock Frequency	3
4	I ² C Transaction Configuration Structure	3
5	Simple I ² C Transaction	4
6	Repeated Start I ² C Transaction	4
7	Simple I ² C Write Transaction	4

1 Introduction

Hardware and layout limitations often require smaller package sizes that could potentially create a tradeoff for the number of serial peripherals available to the user. With the I²C protocol, special software can be created to use a pair of simple GPIO ports to emulate an I²C master device. This allows programmers to be flexible with pin assignments and enables smaller package devices to overcome the hardware limitation of scarce number of hardware I²C peripherals. Having a software-emulated I²C master enables programmers to communicate effectively and fully control attached slave devices without excessive overhead. In addition to the basic I²C functionality, several advanced features such as clock stretching and NAK detection are also supported by the software included in this application report.

The implementation of the software in this application report was made to be highly generic and easily scaled to other MSP430™ and MSP432™ MCUs. The core code is written in bare-metal C and requires only a hardware timer for operation. In the provided source example, the highly accessible and abundantly available Timer_A peripheral is used to precisely handle all of the timing requirements of the I²C master. If a timer other than Timer_A is required, the necessary changes that are needed to port to a different architecture are abstracted out to allow high portability and quick integration.

2 Use Cases

The main use case for having a software I²C master is to overcome the limited number of hardware I²C peripherals that arises from having a smaller package device. For example, a simple example of an MSP430FR5738 connected to an LDC1614 inductive sensor is an ideal use case. The MSP430FR5738 device in a DSBGA package has only one hardware I²C peripheral available on the device. In applications that involve consumer electronics, the need for two I²C peripherals (one to be a slave to a host application processor and one to be a master to a sensor) is required. These types of applications also require a small package size. [Figure 1](#) shows the hardware configuration for this scenario.

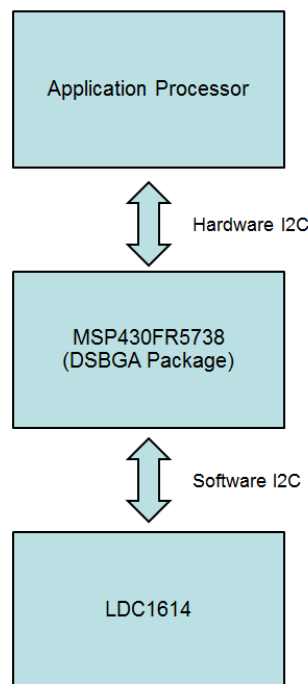


Figure 1. Sample Hardware Configuration

In this example, the MSP430 MCU acts as a buffer between the LDC1614 and the application processor. This sort of scenario is very common in applications such as smart phones and tablets where a low-power centric microprocessor such as the MSP430 MCU is required to relay sensor information to the application processor. The hardware I²C peripheral of the MSP430 MCU is used as an I²C slave to the application processor and the software-emulated I²C is used as a master to the LDC1614.

3 Configuration

The entirety of the configuration parameters is stored within the *msp430_swi2c_master.h* header file. There are two different configuration settings that the user should be concerned about: the pin configurations and the timer period setting. The pin configurations control which pins on the MSP430 MCU are used for the software I²C master. [Figure 2](#) shows a sample configuration.

```
/* Pin Definitions. These should be changed depending on the device that
 * you are using.
 */
#define SWI2C_SCL 0x00000007
#define SWI2C_SDA 0x00000006
#define SWI2C_PxDIR 0x00000007
#define SWI2C_PxOUT 0x00000007
#define SWI2C_PxIN 0x00000007
#define SWI2C_SDA_LOW 0x00000006
#define SWI2C_SCL_LOW 0x00000007
#define SWI2C_SCL_HIGH 0x00000007
#define SWI2C_SDA_HIGH 0x00000006
```

Figure 2. Pin Configuration

As can be seen, this example sets P1.6 and P1.7 as I²C SDA and I²C SCL, respectively. Any ordinary GPIO can be used, and there are no special requirements such as interrupt capability. It is important to note that the configured pins should be reserved exclusively for I²C operation. If another peripheral or function uses or reconfigures the selected pins for a different purpose, the behavior will be unreliable. The second configuration parameter is the parameter that controls the timer. This timer is used to generate the I²C clock frequency. This parameter relies on the clock frequency of SMCLK. [Figure 3](#) shows the formula to derive the desired I²C clock frequency.

```
I2C Data Rate = SMCLK Frequency
                /
                (2 * TimerPeriod)
```

Figure 3. Formula to Derive I²C Clock Frequency

For example, assume that a 100-kHz clock frequency is required, and that the SMCLK frequency is 24 MHz. Putting this information into the formula in [Figure 3](#), a value of 120 must be set for *SWI2C_TIMER_PERIOD*. The main limitation to this software solution is the maximum data rate that can be obtained on the I²C bus. Using the MSP430FR5738 and LDC1614 combination previously described, a maximum of 200 kHz was achieved. This limitation can vary from device to device, and experimentation should be done on the target MSP MCU to discover the ceiling for the device in use.

When all of the configuration settings are set, call the *SWI2C_initI2C* function to persist the settings and configure the pins. This function not only sets the specified pins to the correct direction and output values, but also configures the hardware timer that is used for frequency generation. Note that whenever this function is called, all settings are reset and the software I²C master is returned to the base default settings.

4 Simple Transactions

Transactions using the emulated software I²C master are modeled largely after the same programming model that users might find on host systems such as Linux or UEFI. A simple transaction structure is used to control the slave address, how many bytes are sent, how many bytes are read, transaction buffers, and whether or not to perform a repeated start. [Figure 4](#) shows the structure definition.

```
/* Configuration structure for performing an I2C transaction */
typedef struct _SWI2C_I2CTransaction
{
    uint8_t slaveAddress;
    uint_fast16_t numWriteBytes;
    uint8_t* writeBuffer;
    uint_fast16_t numReadBytes;
    uint8_t* readBuffer;
    bool repeatedStart;
} SWI2C_I2CTransaction;
```

Figure 4. I²C Transaction Configuration Structure

The address variable is the 7-bit slave address to perform the transaction on. Note that only 7-bit addresses are supported. The *numWriteBytes*, *writeBuffer*, *numReadBytes*, and *readBuffer* variables all contain the number of bytes to read or write and pointers to the source and destination buffers. After this structure is configured, a pointer to this structure is passed into the *SWI2C_performI2CTransaction* function, and the software manages the I²C transaction. Note that the pointers being passed for reading data must be pre-allocated, and the software does not do any dynamic memory allocation or boundary checks. If a buffer is passed in for *readBuffer* that is too small to hold the number of read bytes, a buffer overflow will occur.

A variety of different I²C transactions are supported with this software. When the *repeatedStart* variable is set to *false*, simple START/STOP I²C transactions occur. Figure 5 shows this sort of transaction.

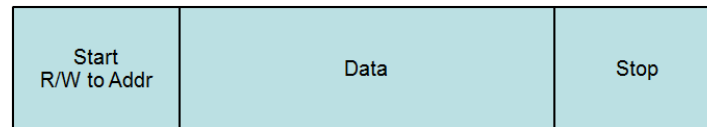


Figure 5. Simple I²C Transaction

Note that if both write and read data is given to the transaction and the *repeatedStart* variable is false, two separate transactions occur on the bus, each having their own I²C START and STOP. If *repeatedStart* is set as true, a repeated start transaction is performed (see Figure 6).

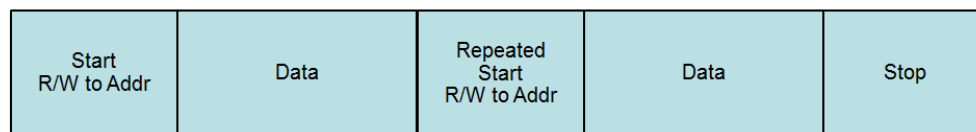


Figure 6. Repeated Start I²C Transaction

Figure 7 shows a simple sample source snippet that configures the device, sets up the transaction, and executes a simple write command to an attached slave with address 0x48.

```
SWI2C_I2CTransaction myTransaction;
uint8_t myBuffer[5] = {1,2,3,4,5};

/* Initializing the master */
SWI2C_initI2C();

/* Setting up the transaction */
myTransaction.address = 0x48;
myTransaction.writeBuffer = myBuffer;
myTransaction.numWriteBytes = 5;

if(!SWI2C_performI2CTransaction(&myTransaction))
{
    /* Handle Error Code Here */
}

while(1);
```

Figure 7. Simple I²C Write Transaction

The code included with this application report is meant to be generic and easily ported between multiple platforms and IDEs. This code compiles without issue on different IDEs such as Code Composer Studio™ IDE, IAR Embedded Workbench® IDE, and GCC.

5 Advanced Features

In addition to being able to perform simple I²C transactions, the software I²C master is also able to perform several advanced I²C features. One of these features is the ability to detect when the slave device is holding the SCL line low in a "clock stretching" scenario. This is a technique that is done often by a slave device when it wants to stall the master and hold off from any additional transactions while it processes data. The software I²C master can detect this condition and wait for the slave device to release the line before continuing with the transaction. Note that there is no time-out feature associated with the clock stretching, and the slave could conceivably cause the master to deadlock if care is not taken by the slave logic.

An additional extended feature of the software I²C library is the ability for the master to detect when the slave device has sent a NAK condition over the I²C line. A NAK condition can occur in several different situations. If the master tries to conduct a transaction on a slave address that does not exist on the I²C bus, a NAK is returned. Additionally, if the master tries to perform a transaction and the slave is not yet ready for that transaction, the slave can NAK the attempt from the master. In the case of a NAK condition, the return value from the *SWI2C_performI2CTransaction* function is a Boolean false.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com