Application Report
SLAA208A–October 2004–Revised August 2010

Interfacing an EEPROM via I²C Using the MSP430

William Goh — *MSP430 Applications*
Christian Hernitscheck — *MSP430 Applications Europe*

ABSTRACT

This report describes the implementation of I²C communication between the MSP430F16x USART or the MSP430F2xx USCI I²C hardware module and an external EEPROM (24xx128). The application report implements various EEPROM protocols such as Byte Write, Page Write, Current Address Read, Random Address Read, Sequential Read and Acknowledge Polling.

Code for this application report can be downloaded from www.ti.com/lit/zip/slaa208.

Contents

List of Figures

List of Tables

1 Example Schematic

The schematic in Figure 1 shows how an EEPROM device can be connected to a MSP430 that has a hardware I²C module. On the F16x devices, it uses a USART module. On the F2xx devices, it uses an USCI module.

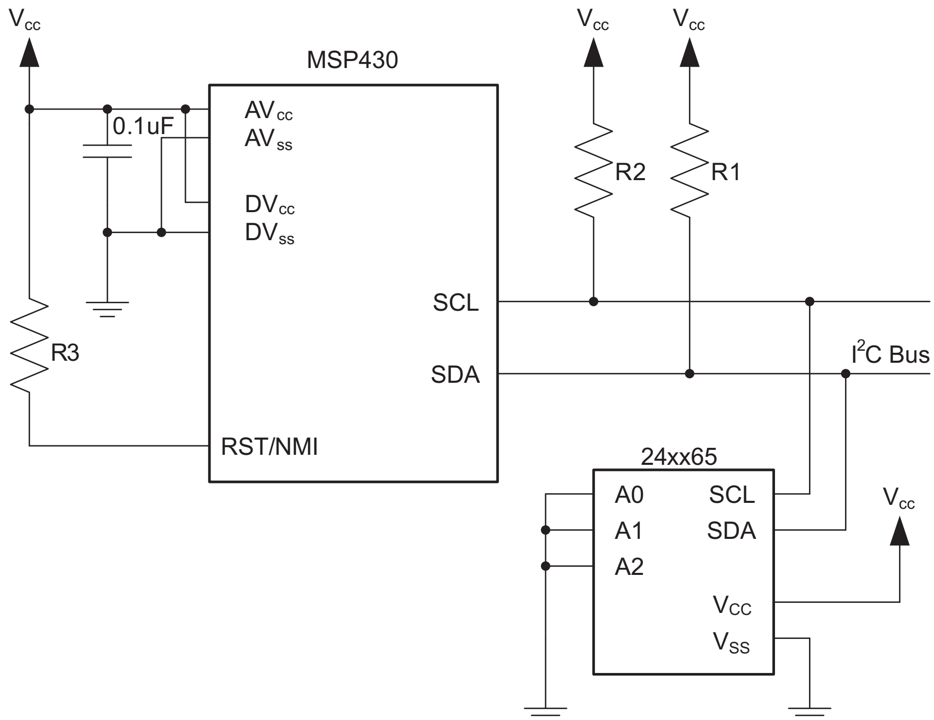

Figure 1. Interfacing the 24xx128 EEPROM to the MSP430 via I²C Bus

On the 24xx128, the user configurable pins A0, A1, and A2 define the I²C device addresses of the connected EEPROMs. The inputs are used to allow multiple devices to operate on the same bus. The logic levels applied to these pins define the address block occupied by the device in the address map. A particular device is selected by transmitting the corresponding bits (A0, A1, and A2) in the control byte.

2 MSP430 Source Code

The software example shows how to use the MSP430 USART and USCI module for communication with EEPROM via I^2C bus. Depending on the memory size of the EEPROM the addressing scheme may look different. There are EEPROM versions that only need one byte for addressing (memory size of 256 bytes or less) and there are EEPROM versions that need two bytes. The example code uses two bytes for addressing. A detailed description of the MSP430 USART and USCI module operation in I^2C mode can be found in references [1] and [2].

There are 2 folders in this project which are files for 1xx and 2xx devices. Both folders contain the same filename and functionality:

Table 1. Project Filenames and Sescriptions

Filename	Description
Main.c	Contains the main function that demonstrates various examples on how to use the available EEPROM functions
I2Croutines.c	EEPROM function library source file
I2Croutines.h	EEPROM function library header definitions file

The I^2C routines have the functions described in the following sections and can be used as a library.

2.1 InitI2C

Declaration void InitI2C(unsigned char eeprom_i2c_address);

Description Initializes the MSP430 for I^2C communication

Arguments eeprom_i2c_address

Target EEPROM address to be initialized

Return none

2.2 EEPROM_ByteWrite

Declaration void EEPROM_ByteWrite(unsigned int Address , unsigned char Data);

Description A byte write command that writes a byte of data into the specified address that is provided.

Arguments Address

Address to write in the EEPROM

Data

Data to be written

Return none

Figure 2 shows the Byte Write protocol.

S	Contol Byte	R/W	ACK	Word Address 1	ACK	Word Address 2	ACK	Data	ACK	P

Figure 2. EEPROM "Byte Write" Command

When using a 24xx128 EEPROM, the upper seven bits of the control byte is always structured in the following way: The upper 4 bits are a fixed number (1010) and the lower 3 bits are defined by the logical level connected to the pins A2, A1, and A0 of the EEPROM.

For storing one byte of data in the EEPROM, four bytes have to be sent via I^2C. The first byte is the control byte, which is followed by the EEPROM address. This is the address a byte to be stored in the EEPROM. Large EEPROM memory uses two bytes for defining the EEPROM address.

The fourth and last byte is the actual data that is stored in the accessed EEPROM. The data is written into the EEPROM address that is transmitted as a part of the command format (see Figure 2).

2.3 EEPROM_PageWrite

Declaration	void EEPROM_PageWrite(unsigned int StartAddress , unsigned char * Data , unsigned char Size);
Description	A Page Write command that writes a specified size of data from the data pointer into the specified address.
Arguments	StartAddress Starting point address to start writing in the EEPROM Data Pointer to data array to be written Size Size of data to be written
Return	none

Figure 3 shows the Page Write protocol implementation. The control byte is first transferred followed by the address. Then, a sequence of data is transmitted. The Page Write command writes 64-bytes at a time as per the EEPROM datasheet [5]. Then the EEPROM is polled for an acknowledge via the acknowledge polling function to ensure that the data write is complete before sending the next 64-byte packet. This 64-byte packetization of data array buffer is handled automatically inside the page write function.

S	Contol Byte	W	ACK	Word Address 1	ACK	Word Address 2	ACK	Data Byte 0	ACK	...	Data Byte 63	ACK	P

Figure 3. EEPROM "Page Write" Command

2.4 EEPROM_AckPolling

Declaration	void EEPROM_AckPolling(void);
Description	The Acknowledge Polling is used to check if the write cycle of the EEPROM is complete. After writing data into the EEPROM, the Acknowledge Polling function should be called.
Arguments	none
Return	none

The EEPROM requires a finite time to complete a write cycle. This is typically defined in the EEPROM datasheet however it is possible that the write cycles may complete faster than the specified time.

The function EEPROM_AckPolling() takes advantage of the fact that the EEPROM will not acknowledge its own address as long as it is busy finishing the write cycle. This function continues to send a control byte until it is acknowledged, meaning that the function will only return after the write cycle is complete.

2.5 EEPROM_RandomRead

Declaration	unsigned char EEPROM_RandomRead(unsigned int Address);
Description	The Random Read command of the EEPROM allows reading the contents of a specified memory location.
Arguments	Address Address to be read from the EEPROM
Return	Data byte from EEPROM

Figure 4 shows the Random Address read protocol. It uses master-transmit and master-receive operation without releasing the bus in between. This is achieved by using a repeated START condition.

S	Contol Byte	W	ACK	Word Address 1	ACK	Word Address 2	ACK	S	Control Byte	R	ACK	Data	NACK

Figure 4. EEPROM "Random Access Read" Command

First the address counter of the EEPROM has to be set. This is done using a master-transmit operation. After sending the address, the MSP430 is configured for master-receive operation and initiates data read by sending a repeated START condition.

2.6 EEPROM_CurrentAddressRead

Declaration	unsigned char EEPROM_CurrentAddressRead(void);
Description	The EEPROM internal address pointer is used. After execution of a write or read operation the internal address pointer is automatically incremented.
Arguments	none
Return	Data byte from EEPROM

Figure 5 shows the current address read operation. The MSP430 is configured as a master -receive before executing this command. Before the communication is started by the MSP430 the I^2C module is configured to master-receive mode.

S	Contol Byte	R	ACK	Data	NACK	P

Figure 5. EEPROM "Current Address Read" Command

After the STOP condition has occurred the received data is returned to the caller of the function.

2.7 EEPROM_SequentialRead

Declaration: void EEPROM_SequentialRead(unsigned int Address , unsigned char * Data , unsigned int Size);

Description: The sequential read is used to read a sequence of data specified by the size and starting from the known address. The data pointer points to where the data is to be stored.

Arguments:
- StartAddress
  - Starting point address to start reading from the EEPROM
- Data
  - Pointer to data array to be stored
- Size
  - Size of data to be read

Return: none

Figure 6 shows the sequential read protocol implementation. The MSP430 is configured as a master transmitter and sends out the control byte followed by the address. After that, a re-start is issued with the MSP430 configured as a master receiver. When the last character is read, a stop command is issued.

S	Control Byte	W	ACK	Word Address 1	ACK	Word Address 2	ACK	S	Control Byte	R	ACK	Data Byte 0	ACK	...	Data Byte 63	ACK	P

Figure 6. EEPROM "Sequential Read" Command

3 Example

The following example shows how to use the functions from the files "I2Croutines.c":

```
#include "msp430.h"
#include "I2Croutines.h"

unsigned char read_val[150];
unsigned char write_val[150];
unsigned int address;

int main(void)
{
  unsigned int i;

  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer

  InitI2C();                                // Initialize I2C module

  EEPROM_ByteWrite(0x0000,0x12);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion
  EEPROM_ByteWrite(0x0001,0x34);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion
  EEPROM_ByteWrite(0x0002,0x56);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion
  EEPROM_ByteWrite(0x0003,0x78);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion
  EEPROM_ByteWrite(0x0004,0x9A);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion
  EEPROM_ByteWrite(0x0005,0xBC);
  EEPROM_AckPolling();                      // Wait for EEPROM write cycle
                                            // completion

  read_val[0] = EEPROM_RandomRead(0x0000);  // Read from address 0x0000
  read_val[1] = EEPROM_CurrentAddressRead();// Read from address 0x0001
  read_val[2] = EEPROM_CurrentAddressRead();// Read from address 0x0002
  read_val[3] = EEPROM_CurrentAddressRead();// Read from address 0x0003
  read_val[4] = EEPROM_CurrentAddressRead();// Read from address 0x0004
  read_val[5] = EEPROM_CurrentAddressRead();// Read from address 0x0005

  // Fill write_val array with counter values
  for(i = 0 ; i <= sizeof(write_val) ; i++)
  {
    write_val[i] = i;
  }

  address = 0x0000;                         // Set starting address at 0
  // Write a sequence of data array
  EEPROM_PageWrite(address , write_val , sizeof(write_val));
  // Read out a sequence of data from EEPROM
  EEPROM_SequentialRead(address, read_val , sizeof(read_val));

  __bis_SR_register(LPM4);
  __no_operation();
```

```
}
```

4 References

1. MSP430x2xx Family User's Guide (SLAU144)
2. MSP430x1xx Family User's Guide (SLAU049)
3. MSP430x261x Data Sheet (SLAS541)
4. MSP430x16x Data Sheet (SLAS368)
5. 24LC128 Data Sheet (24AA128/24LC128/24FC128)

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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