

Interfacing an EEPROM via I²C Using the MSP430

William Goh
Christian Hernitscheck

MSP430 Applications
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.

Contents

1	Example Schematic	2
2	MSP430 Source Code	3
3	Example	6
4	References	7

List of Figures

1	Interfacing the 24xx128 EEPROM to the MSP430 via I ² C Bus	2
2	EEPROM "Byte Write" Command	3
3	EEPROM "Page Write" Command	4
4	EEPROM "Random Access Read" Command	4
5	EEPROM "Current Address Read" Command	5
6	EEPROM "Sequential Read" Command	5

List of Tables

1	Project Filenames and Descriptions	3
---	--	---

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.

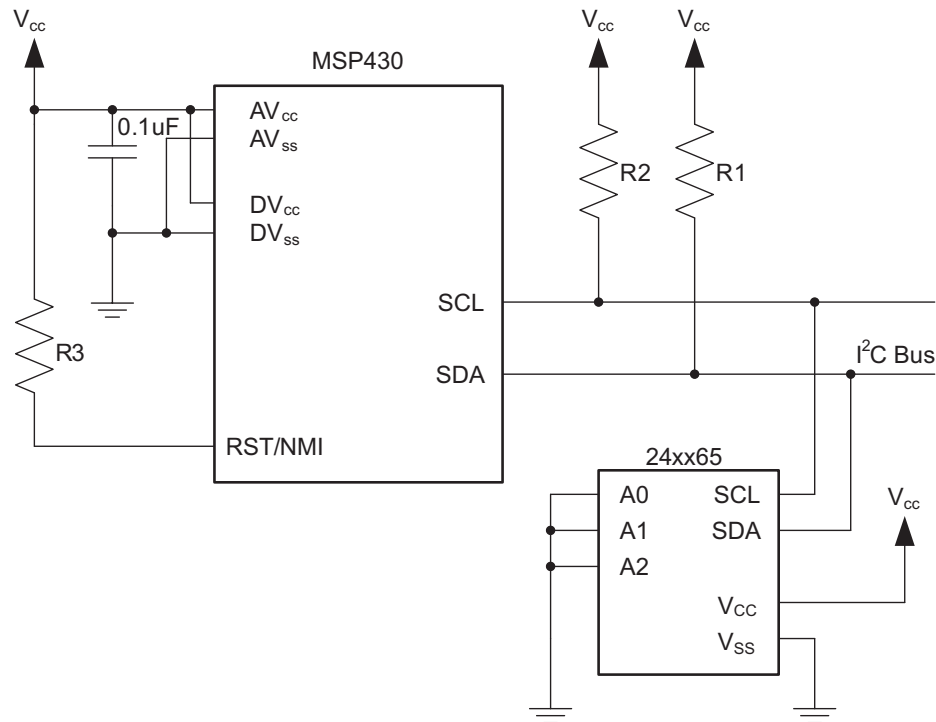


Figure 1. Interfacing the 24xx128 EEPROM to the MSP430 via I²C Bus

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 I²C 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 I²C 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 I²C 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 I²C 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 I²C. 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 I²C module is configured to master-receive mode.

S	Control 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 "I2C routines.c":

```
#include "msp430.h"
#include "I2C routines.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

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	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps