Application Report

SLAA908–September 2019

UART-to-I²C Bridge Using Low-Memory MSP430™ MCUs

Johnson He

MSP430 Applications

ABSTRACT

The universal asynchronous receiver transmitter (UART) interface and the inter-integrated circuit (I²C) interface are two common serial communication interfaces. They both enable communication between the MSP430™ microcontroller (MCU) and another device, such as a personal computer (PC), another MCU, or a processor. Many devices support only one or the other interface, therefore some designs require communication between devices with these different serial protocols. This application report describe a program that can convert between UART and I²C protocols. In this program, both hardware UART and I²C modes are used, which can support baud rates up to 921600 and I²C clock frequencies up to 400 kHz.

Contents

List of Figures

List of Tables

Trademarks

MSP430, LaunchPad are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Introduction

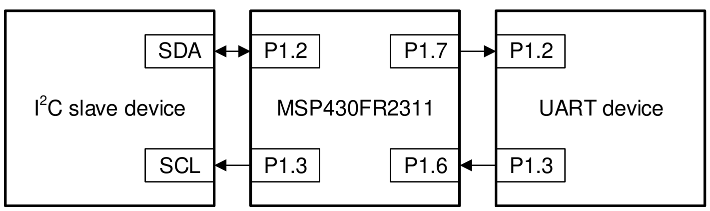
The MSP430FR2311 MCU has two eUSCI modules that can be used as a low-cost UART-to-I^{2}C bridge (configured as an I^{2}C master) using the UART and I^{2}C modes. This bridge make it convenient for hardware devices that support only UART to access I^{2}C protocol devices for data transmission and exchange. To get started, download the project files and a code example that demonstrate this functionality.

2 Implementation

Figure 1 shows the block diagram for the UART-to-I^{2}C bridge. The MSP-TS430PW20 target development board was used to connect the peripherals to the MSP430FR2311 MCU. Set the jumpers as listed here:

- Populate jumpers JP14, JP15, JP17, and JP18
- Do not populate jumper JP13
- Set jumper JP16 to UART
- Set jumpers JP3 to JP8 all on 1-2
- Set jumper JP11 on 1-3 and 2-4

These jumper settings allow the back-channel UART interface on the MSP-FET programmer and debugger to simulate the UART device that will communicate with the bridge. At the same time, the 10k resistor is used as a pullup on the I^{2}C pins. To communication with an I^{2}C device, connect the SCL pin of the I^{2}C device to J4.19 (P1.3) and connect the SDA pin to J4.20 (P1.2). A simple I^{2}C slave project was implemented on an MSP430FR2311 LaunchPad™ development kit to demonstrate the functionality of the UART-to-I^{2}C bridge. The UART-to- I^{2}C bridge functions in I^{2}C master mode.



NOTE: Pullup resistors are required on the I^{2}C bus.

Figure 1. UART-to-I^{2}C Bridge Block Diagram

Using a PC, open a new serial connection with a terminal program, and connect to the back-channel UART interface on the MSP-FET by selecting the COM port called *MSP Application UART1*. In the terminal window, change the baud rate to 115200. To demonstrate the functionality of the UART-to-I^{2}C bridge, enter a string of bytes into the terminal window follow command formats and send it. It will be sent to the I^{2}C slave device, and whatever value was in the TX buffer of the I^{2}C slave device will be displayed in the serial terminal.

Figure 2 shows the flowchart for the I^{2}C and UART code.

When an I^{2}C packet is received, the UCRXIFG interrupt flag is set, and the data is read from the I^{2}C RX buffer and stored in the internal buffer (default size is 100 bytes). Next when the data is receive completed, write this data to UART TX buffer and send it through the UART.

When a UART packet is received, it is stored in the internal buffer (the default size is 100 bytes), and then the application analyzes the UART packet command format. Different steps are performed for these two command formats: read or write data through the I^{2}C interface, or read or write internal register data.

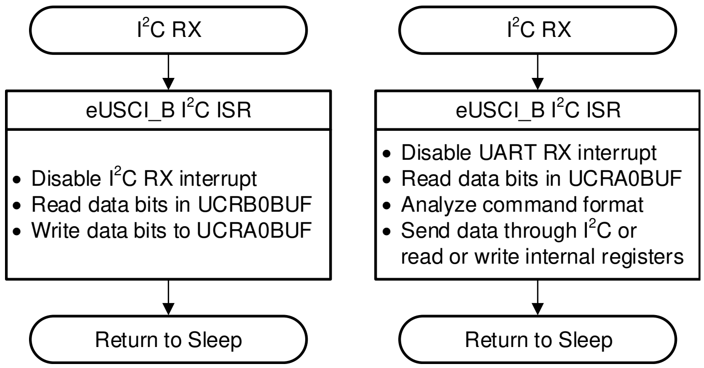


Figure 2. I^2C and UART Code Flow

3 UART Message Format

The host initiates an I^2C bus data transfer, or reads from and writes to internal registers through a series of ASCII commands. Table 1 lists the supported ASCII commands and their hexadecimal value representation. Unrecognized commands are ignored by the device.

Table 1. ASCII Commands

ASCII	Hex Value	Function
S	0x53	I^2C start
P	0x50	I^2C stop
R	0x52	Read from internal register
W	0x57	Write to internal register

3.1 *Write N Bytes to Slave Device*

The host issues the write command by sending an S character followed by an I^2C bus slave device address, the total number of bytes to be sent, and I^2C bus data which begins with the first byte (DATA 0) and ends with the last byte (DATA N). The frame is then terminated with a P character. Once the host issues this command, the MSP430 MCU will access the I^2C bus slave device and start sending the I^2C bus data bytes.

Note that the second byte sent is the I^2C bus device slave address. The least significant bit (W) of this byte must be set to 0 to indicate this is an I^2C bus write command.

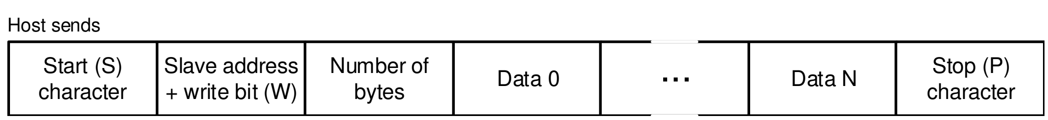


Figure 3. Write N Bytes to Slave Device

3.2 *Read N Bytes From Slave Device*

The host issues the read command by sending an S character followed by an I^2C bus slave device address, and the total number of bytes to be read from the addressed I^2C bus slave. The frame is then terminated with a P character. Once the host issues this command, the MSP430 MCU will access the I^2C bus slave device, get the correct number of bytes from the addressed I^2C bus slave, and then return the data to the host.

The second byte sent is the I^2C bus device slave address. The least significant bit (R) of this byte must be set to 1 to indicate this is an I^2C bus write command.

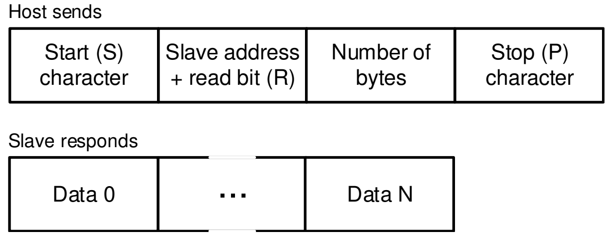


Figure 4. Read N Bytes From Slave Device

3.3 *Repeated Start (Read After Write)*

The bridge also supports ‘read after write’ command as specified in the I^2C bus specification. This allows a read command to be sent after a write command without having to issue a STOP condition between the two commands.

The host issues a write command as normal, then immediately issues a read command without sending a STOP (P) character after the write command.

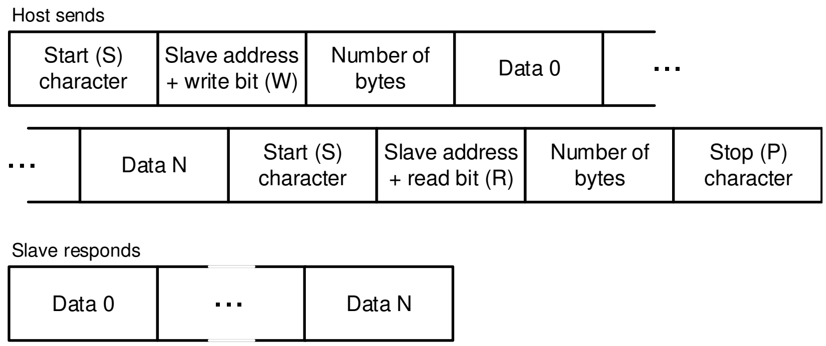


Figure 5. Repeated Start : Read After Write

3.4 *Repeated Start (Write After Write)*

The bridge also supports ‘write after write’ command as specified in the I^2C bus specification. This allows a write command to be sent after a write command without having to issue a STOP condition between the two commands.

The host issues a write command as normal, then immediately issues a second write command without sending a STOP (P) character after the first write command.

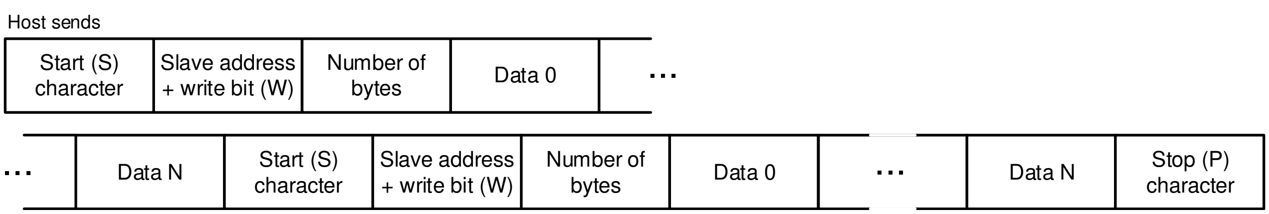


Figure 6. Repeated Start : Write After Write

3.5 *Write to Internal Register*

The host issues the internal register write command by sending a W character followed by the register and data pair. Each register to be written must be followed by the data byte.

The frame is then terminated with a P character.

Host sends

Write register (W) character	Register 0	Data 0	...	Register N	Data N	Stop (P) character

Figure 7. Write to Internal Register

NOTE: The values of the written registers are stored in the internal FRAM of the MSP430 MCU and have a memory function.

3.6 Read From Internal Register

The host issues the internal register read command by sending an R character followed by the registers to be read.

The frame is then terminated with a P character.

Host sends

Read register (R) character	Register 0	...	Register N	Stop (P) character

MCU responds

Data 0	...	Data N

Figure 8. Read From Internal Register

4 Internal Registers Available

4.1 Register Summary

Table 2 lists the internal registers of the MSP430FR2311 that control this application.

Table 2. Internal Registers Summary

Address	Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	$\overline{R}$/W
0x00	BRG0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	R/W
0x01	BRG1	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	R/W
0x02	I2CClkL	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	R/W
0x03	I2CClkH	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	R/W

4.2 Baud Rate Generator (BRG)

This register sets the UART baud rate (the default is 115200). Table 3 lists the supported rates.

Table 3. UART Baud Rate

BRG		UART Baud Rate
(hex)	(dec)	
0x0023	35	460800
0x0046	70	230400
0x008A	138	115200
0x0116	278	57600
0x0682	1666	9600

The rate is programmed through the BRG register and the baud rate can be calculated as Equation 1.

$$Baud\ rate = \frac{16 \times 10^6}{(BRG1, BRG0)} \tag{1}$$

NOTE: For the new baud rate to take effect, both BRG0 and BRG1 must be written with new values simultaneously. The new baud rate takes effect immediately after BRG0 or BRG1 are written.

4.3 I²C Bus Clock Rates (I2CClk)

This register sets the serial clock frequency (the default is 100 kHz). Table 4 lists the supported serial rates.

Table 4. I²C Bus Clock Frequency

I2CClk		I²C Bus Clock Frequency (kHz)
(hex)	(dec)	
0x0028	40	400
0x0050	80	200
0x00A0	160	100
0x0140	320	50
0x0280	640	25

The frequency can be determined using Equation 2.

$$I^2C\ Bit\ Frequency = \frac{16 \times 10^6}{(I2CClkH, I2ClkL)} \tag{2}$$

NOTE: For the new bit frequency to take effect, both I2CClkH and I2CClkL must be written with new values simultaneously. The new baud rate takes effect immediately after I2CClkH or I2CClkL are written.

5 Performance

The firmware supports full duplex UART communication. It also supports UART packets with eight data bits, least significant bit (LSB) first, no parity bit, and one stop bit. Because two serial interfaces are required, this firmware can only be used on MSP430 MCUs that have at least two USCI modules.

The firmware creates two 100-bytes buffer to hold the data in the conversion. After the data is stored in the buffer, the next conversion is performed. Therefore, the maximum number of bytes sent in a single transmission cannot exceed 100 bytes (including the frame header). If you require packets larger than 100 bytes of data in single transmission, modify the two parameters I2C_TO_UART_BUFF_SIZE and UART_TO_I2C_BUFF_SIZE in the firmware to meet the requirements.

Table 5. Maximum Rates

Interface	Maximum Rate
I²C	400-kHz bit clock
UART	921600-bps baud rate

To change the baud rate of the UART interface or the clock frequency of the I²C interface, change the UCA0BRW or UCB0BRW register in the code. In this application, SMCLK operates at 16 MHz. To change this, see the *MSP430FR4xx and MSP430FR2xx Family User's Guide* for the proper configuration and the *MSP430FR231x Mixed-Signal Microcontrollers datasheet* for the maximum I²C clock value. After making these changes, close the serial terminal, rebuild the code, reprogram and reset the MCU, and then reopen the terminal with the new baud rate.

The I²C clock speed and the UART baud rate can also be changed using UART commands (see Section 3 and Section 4). When the baud rate of the UART or the clock frequency of the I²C is changed by the UART, the change takes effect immediately and is saved after power off and restart.

To reduce power consumption while the MCU is not receiving or transmitting data, LPM0 is used. Other low-power modes can achieve lower power consumption, but they might require an external crystal oscillator and can limit the maximum baud rate due to increased wake-up times.

6 Application Examples

6.1 Test With I²C Slave Device

In this example, another MSP430FR2311 is used as the I²C slave device, and the I²C slave program is loaded internally You can download the slave program for reference, and you need to add RX interrupts and buffers for correct operation. The slave address is 0x48. When receiving a write command sent by the host device, the data is stored in the buffer area. When receiving a read command sent by the host device, the slave device sends incremental data starting at 0x00.

Figure 9 shows the serial port command sent by the I²C master to write data and read the data.

In the write example, the UART data command writes 8 bytes to the I²C slave. The command is composed of these bytes:

- The first byte is 'S' (0x53) to indicate I²C start
- The second byte indicates the slave address and write command (0x48 << 1 | 0 = 0x90)
- The third byte indicates the length of the write data (0x08)
- The next 8 bytes are the data to write to the slave (0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88)
- The last byte is 'P' (0x50) to indicate I²C stop

In the read example, the UART data command reads 8 bytes from I²C slave.

- The first byte is 'S' (0x53) to indicate I²C start
- The second byte indicates the slave address and the read command (0x48 <<1 | 1 = 0x91)
- The third byte indicates the length of the read data (0x08)
- The last byte is 'P' (0x50) to indicate I²C stop

```
[14:40:27.327]OUT→◇53 90 08 11 22 33 44 55 66 77 88 50 □ Write to slave device 8 bytes (slave address: 0x48)
[14:40:34.263]OUT→◇53 91 08 50 □ Read 8 bytes from slave device (slave address: 0x48)
[14:40:34.345]IN←◆00 01 02 03 04 05 06 07 8 bytes data from slave device
```

Figure 9. I²C Write and Read

Figure 10 shows the waveform of the I²C write data. From the figure, we can see the start bit, slave address, write command, write data and the end bit in SDA line.

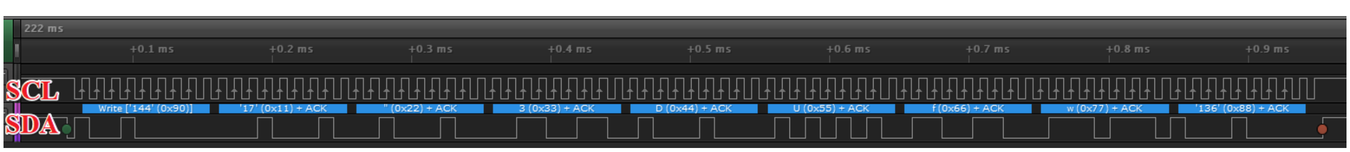


Figure 10. I²C Write Data Waveform

Figure 11 shows the waveform of the I²C read data. From the figure, we can see the start bit, slave address, read command, read data and the end bit in SDA line.

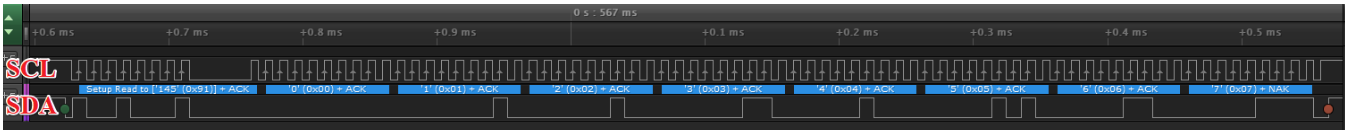


Figure 11. I²C Read Data Waveform

6.2 *Read and Write EEPROM*

This firmware was tested using EEPROM as the I²C slave device to verify the correctness of the firmware by reading and writing the EEPROM. The EEPROM address is 0x50. Figure 12 to Figure 15 show the read and write format.

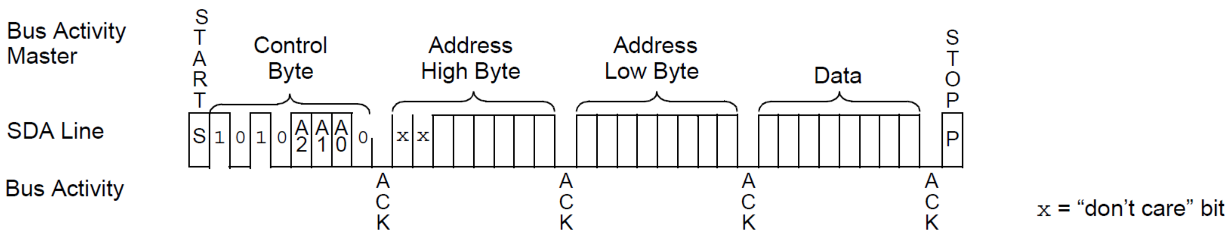


Figure 12. EEPROM Byte Write

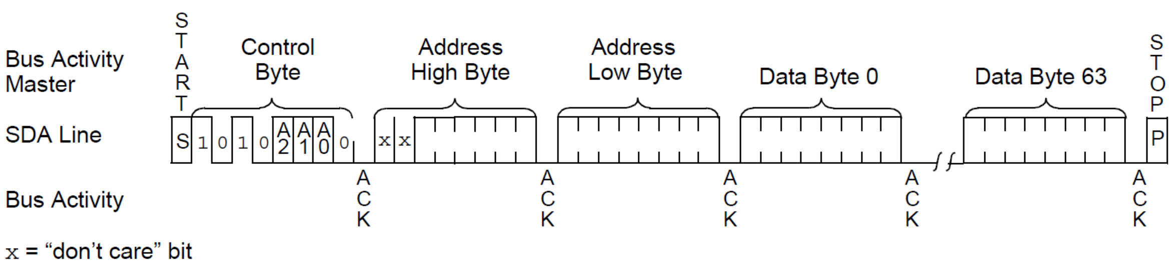


Figure 13. EEPROM Page Write

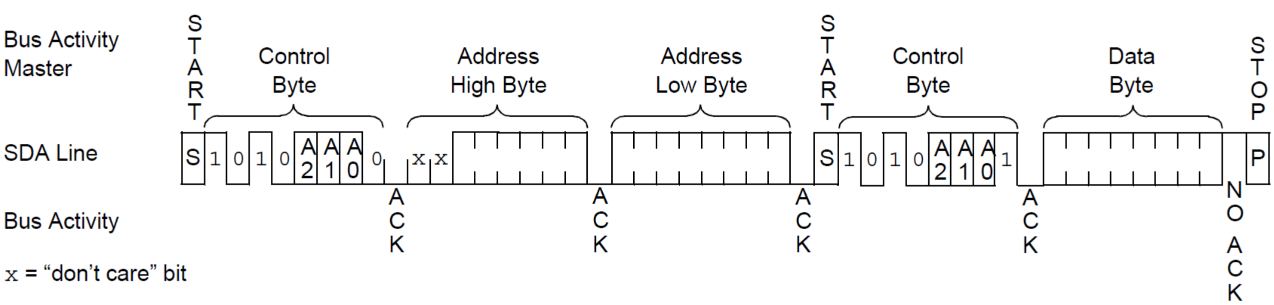


Figure 14. EEPROM Random Read

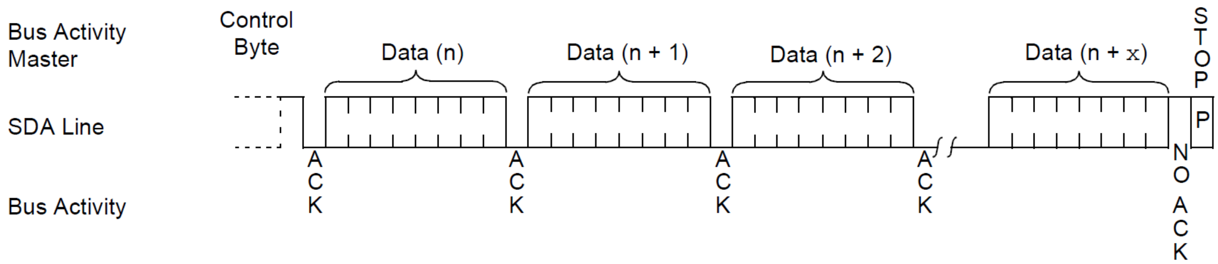


Figure 15. EEPROM Sequential Read

The firmware first sets the I^2C clock frequency to 100 kHz and sets the UART baud rate to 115200 by writing the internal registers (Figure 16 shows the UART commands to read and write the internal register values). For the write test, the firmware writes 64 bytes of data to the EEPROM starting at address 0x0000. For the read test, the firmware reads 64 bytes of data starting at address 0x0000.

Figure 16 shows the UART data commands to write to and read from the internal registers.

The write command is composed of these bytes:

- The first byte is 'W' (0x57) to indicate a write to internal registers
- The second byte is the internal register start address to write
- The next four bytes are the data to write
- The last byte is 'P' (0x53) to indicate stop

The read command is composed of these bytes:

- The first byte is 'R' (0x52) to indicate a read from internal registers
- The next four bytes are the addresses to read (0x00, 0x01, 0x02, 0x03)
- The last byte is 'P' (0x53) to indicate stop

The final four bytes in the last row of Figure 16 are the values of the registers specified in the read command.

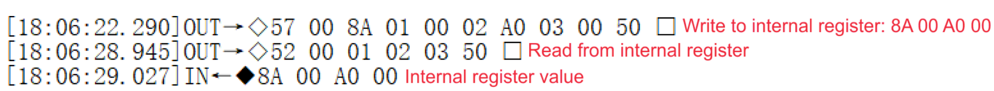


Figure 16. Write and Read Internal Register (Set I^2C Clock Frequency and UART Baud Rate)

Figure 17 shows the UART commands to read and write the EEPROM.

The write command is composed of these bytes:

- The first byte is 'S' (0x53) to indicate an I^2C start
- The second byte is the EEPROM I^2C address and write command (0x50 << 1 | 0 = 0xA0)
- The next three bytes is the length of the write data (0x42), which includes the 2-byte address plus the number of bytes to write (64 in this case)
- The next two bytes are the start address in the EEPROM to write
- The next 64 bytes are the data
- The last byte is 'P' (0x50) to indicate stop

The read command is composed of these bytes:

- The first byte is 'S' (0x53) to indicate an I^2C start
- The second byte is the EEPROM I^2C address and write command (0x50 << 1 | 0 = 0xA0)
- The third byte is the length of the data (0x02), which is for the 2-byte address
- The next two bytes are the start address in the EEPROM to read
- The next byte is 'S' (0x53) to indicate an I^2C restart

- The next byte is the EEPROM I²C address and the read command (0x50<<1 | 1 = 0xA1)
- The next byte is the length of the read data (64 bytes) (0x40)
- The last byte is 'P' (0x50) to indicate stop

Figure 17 shows that the read data is the same as the write data.

```
[18:08:05.194]OUT→◇53 A0 42 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D
1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 50 □ Write 00 to 3F to EEPROM from address 0x0000
[18:08:11.994]OUT→◇53 A0 02 00 00 53 A1 40 50 □ Read 40 bytes from EEPROM address 0x0000
[18:08:12.077]IN←◆00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22
23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F Value from EEPROM
```

Figure 17. Write and Read EEPROM Value

7 Reference

1. *MSP430FR4xx and MSP430FR2xx Family User's Guide*
2. *MSP430FR231x Mixed-Signal Microcontrollers data sheet*
3. *UART-to-SPI Bridge Using Low-Memory MSP430™ MCUs*
4. *UART-to-UART Bridge Using Low-Memory MSP430™ MCUs*
5. *Microchip 128Kb I²C compatible 2-wire Serial EEPROM (24LC128)*
6. *NXP Master I²C bus Controller with UART Interface (SC18IM700IPW)*

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