Application Report
SNAA138C–March 2012–Revised May 2013

AN-2230 MSP430 Interface to LMP91000 Code Library

ABSTRACT

The MSP430™ is an ideal microcontroller solution for low-cost, low-power precision sensor applications because it consumes very little power. The LMP91000 is a programmable analog front end (AFE) for use in micro-power electrochemical sensing applications. It provides a complete signal path solution between a sensor and a microcontroller that generates an output voltage proportional to the cell current. This library provides functions to facilitate the interfacing of any MSP430 device to a LMP91000. Any device within the MSP430 family can be used with this library, made possible by hardware abstraction. Similarly, any I2C-capable interface module within the MSP430 family is supported by the library. This allows the designer maximum flexibility in choosing the best MSP430 device for the application. This document provides descriptive information and instructions for using the library either for demonstration purposes or implementation into a project. This is the recommended starting point for developing software for the LMP91000 and MSP430 combination. The software examples have been developed for the MSP430F5528 breakout board, but can easily be ported to another hardware platform.

NOTE: This application report and library is also applicable for the MSP430 interfacing to LMP91002 devices. The LMP91002 specific differences, if any, are highlighted throughout this document as well as in the library source code.

Contents

List of Figures

List of Tables

1 Introduction

This application report describes different ways to interface and use the TI LMP91000 devices with an MSP430. The accompanying software contains a function library allowing quick prototyping of LMP91000 AFE setup and control. The software provided in this library is a starting point for developers wanting to get the most out of the MSP430 and the LMP91000 sensor AFE devices.

The LMP91000 is a programmable AFE for use in micro-power chemical sensing applications and an optimal match for the MSP430 ultra low power microcontrollers. The LMP91000 is designed for 3-lead single gas sensors and for 2-lead galvanic cell sensors. It is equipped with a slave I2C port, through which it can communicate with an MSP430. The LMP91000 generates an output voltage proportional to the cell current and is ideally suited for detecting changes in gas concentration based on a delta current at the working electrode. The MSP430 is a great fit for applications where power conservation is a priority. The many power-saving mechanisms designed into the MSP430 make it ideal for such applications.

2 Purpose and Scope

To aid in interfacing these devices, TI has produced a code library that significantly reduces the need to write low-level interface functions. It provides a boost in the development of an MSP430/LMP91000-based product, saving time and allowing quick progression to the application-specific aspects of the project. This library is designed to be used with any MSP430 device. Since an I2C master can be implemented using one of many peripherals within the MSP430 family, and since the peripherals available may differ by device and application, library calls are provided for each of these interfaces. The chosen interface is selected by assigning a value to a system variable, which causes the compiler to conditionally include the appropriate function calls. As such, application code utilizing the library remains portable between various MSP430 devices, with minimal modification required.

Several complete example application projects are provided with the library. The purpose of these projects is to demonstrate use of the library. It is not intended as a comprehensive guide to using the LMP91000, and it does not make use of all the features of these devices. It does, however, use all the register access functions provided by the library.

3 File Organization

The library has been implemented with modular hardware abstraction. There is a header file specific to each of the hardware components (LMP91000, MSP430, and the board). The hardware definition header files are shown in Table 1. Table 2 shows the library code files and its header, and Table 3 shows the demonstration applications that accompanies the library.

Table 1. Hardware Definition Files

Filename	Description
TI_LMP91000.h	Definitions specific to the LMP91000 device, including register locations and commonly-used masks for use with these registers.
TI_MSP430.h	Definitions specific to the MSP430 device; primarily, the pins used in the I2C interface. Definitions for USART0 and USCI_B0/1/2/3 are included. Also, labels are defined for use with the system variable *TI_LMP91000_SER_INTF*. This selects the modules to be used for accessing the LMP91000 I2C interface.
TI_MSP430_hardware_board.h	Definitions specific to the board being used; that is, the connections between the MSP430 and LMP91000, such as the MENB and GPIO pins. I2C connections are not defined here because they are defined inherently within *TI_MSP430.h*. The system variable *TI_LMP91000_SER_INTF* is defined in this file.

Table 2. Library Code

Filename	Description
TI_MSP430_i2c_USCIB0_5xx.c	Function for accessing LMP91000 registers via I2C USCIB0 module from MSP430 5xx family
TI_MSP430_other_i2c_modules	Functions for accessing LMP91000 registers via other MSP430 I2C modules like I2C USART0, USCIB1_5xx, USCIB2_5xx, and so forth.
TI_MSP430_i2c_ USCIB1_5xx.c	
TI_MSP430_i2c_ USCIB2_5xx.c	
TI_MSP430_i2c_ USCIB3_5xx.c	
TI_MSP430_i2c_ USCIB0.c	
TI_MSP430_i2c_ USCIB1.c	
TI_MSP430_i2c_ USART0_1xx.c	
TI_MSP430_i2c_ eUSCIB0_6xx.c	
TI_MSP430_i2c.h	Function declarations for *TI_MSP430_i2c_*.c*

Table 3. Demo Applications Included With the Library

Demo Application	Filename	Description
Application1: Read/Write LMP91000 Registers	*demo-app01\main.c*	Application code with functions to demonstrate read/write of LMP91000 registers
	demo-app01\ TI_LMP91000_register_settings.h	Application specific initialization values for the LMP91000 registers
Application2: Setup and Read LMP91000 Temperature Sensor	*demo-app02\main.c*	Application code with functions to demonstrate setup and read LMP91000 temperature sensor. Onchip ADC12 module in MSP430F5528 is used to convert the temperature sensor output. **Note:** This demo application is not applicable to the LMP91002 devices as they don't support an on-chip temperature sensor.
	demo-app02\ TI_LMP91000_register_settings.h	Application specific initialization values for the LMP91000 registers
Application3: Setup and measure LMP91000 VREF	*demo-app03\main.c*	Application code with functions to setup and measure LMP91000 VREF. Onchip ADC12 module in MSP430F5528 is used to convert the LMP91000 VOUT.
	demo-app03\ TI_LMP91000_register_settings.h	Application specific initialization values for the LMP91000 registers
Application4: Setup and measure LMP91000 VREF using onboard ADC161S626 (applicable only when using the new evaluation module LMP91000EVM instead of LMP91000SDEVAL)	*demo-app04\main.c*	Application code with functions to setup and measure LMP91000 VREF. Onboard ADC161S626 in LMP91000 EVM is used to convert and measure the LMP91000 VOUT.
	demo-app04\ TI_LMP91000_register_settings.h	Application specific initialization values for the LMP91000 registers

NOTE: The register settings values for *TI_LMP91000_register_settings.h* can easily be obtained from the “Register configuration file” saved from Sensor AFE Software [2].

Figure 1 shows a stack diagram of the library. Note that one of the files displayed in the stack is the standard definition file for the specific MSP430 device being used. This file is included with the development environment being used to create the MSP430 software.

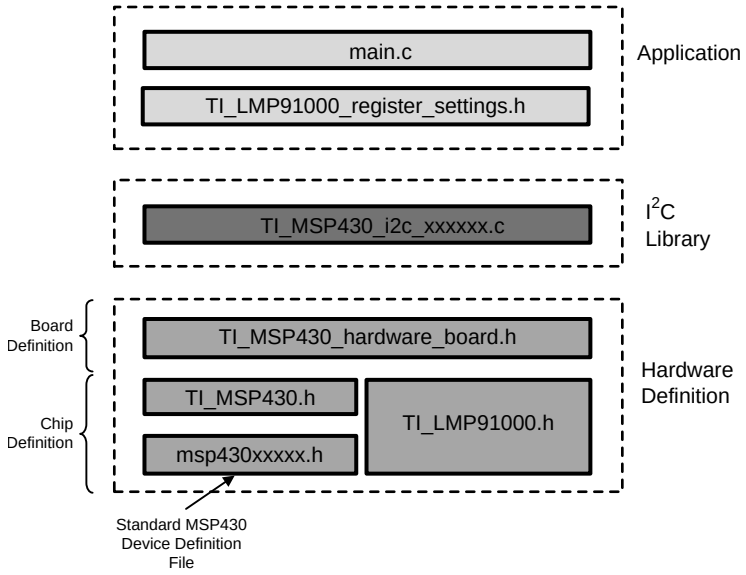

Figure 1. Code Library Stack

4 Functions

Table 4 shows the I2C register-access functions provided in the library, with a brief description.

Table 4. Register Access and Control Functions Provided by the Library

Function Name	Description
void TI_LMP91000_I2CSetup (uint8_t device_I2c_address)	Configures the I2C port assigned by the *TI_LMP91000_SER_INTF* system variable to control slave with device address *"device_i2c_address"*. Must be called before calling any of the other functions.
void TI_LMP91000_I2CWriteReg (uint8_t address, uint8_t data)	Writes "*data*" to a single configuration register at address "*address*". Table 5 shows the I2C write register protocol.
uint8_t TI_LMP91000_I2CReadReg(uint8_t address)	Reads a single register at address "*address*" and returns the value read. Table 6 shows the I2C read register protocol.

4.1 TI_LMP91000_I2CWriteReg

The *TI_LMP91000_I2CWriteReg* function writes a byte of data into the specified register address. The I2C write register protocol is shown in Table 5. For writing one byte of data in the LMP91000, three bytes have to be sent via I2C. The first byte is the control byte that is the I2C device address of the LMP91000. This is followed by the LMP91000 register address where a byte is to be stored. The third and last byte is the actual data that is written to the LMP91000 register.

Table 5. LMP91000 Write Register Protocol

S	Control Byte	W	ACK	Address	ACK	Data	ACK	P

4.2 TI_LMP91000_I2CReadReg

The *TI_LMP91000_I2CReadReg* function allows reading the contents of a specified register address. The I2C read register protocol is shown in Table 6. It uses master-transmit and master-receive operation without releasing the bus in between. This is achieved by using a repeated START condition.

Table 6. LMP91000 Read Register Protocol

S	Control Byte	W	ACK	Address	ACK	S	Control Byte	R	Data	ACK	P

First, the LMP91000 register address has to be sent. 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.

4.3 MSP430 Peripherals Supported

A version of these library functions is provided for all MSP430 peripherals that are capable of communicating using the I2C protocol. These peripherals are:

- USCI_B0 for 5xx and 6xx families
- USCI_B1 for 5xx and 6xx families
- USCI_B2 for 5xx and 6xx families
- USCI_B3 for 5xx and 6xx families
- USCI_B0 for 2xx and 4xx families
- USCI_B1 for 2xx and 4xx families
- USART0 for 1xx family
- eUSCI_B0 for 673x family

5 Using the Software

5.1 Prerequisites

To successfully compile, download and run the software described in this document, the following material is needed:

- MSP430 Target Board MSP-TS430RGC64USB Board
- LMP91000 Evaluation Board LMP91000SDEVAL or the new LMP91000 Evaluation Module LMP91000EVM
- MSP430 USB Debugging Interface MSP430-FET430UIF
- IAR Embedded Workbench or TI Code Composer Studio™ for MSP430

The software can be adapted to run on other MSP430 hardware boards as well. For instructions, see Section 5.3.

A free, code size limited, but fully functional edition of IAR Embedded Workbench (IAR Kickstart) is available from the IAR Systems website (www.iar.com) or from the TI MSP430 software tools page

http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/msp430_software_landing.page

As an alternative to IAR Embedded Workbench, it would be possible to use Code Composer Studio. A trial edition of the Code Composer Studio is available from the TI MSP430 software tools page.

5.2 Getting Started

Follow these simple steps to get your application up and running:

1. Install IAR Workbench
2. Download the source code for this application report and unzip the files to your working directory.
3. Open IAR Embedded Workbench and create a new project:
   (a) Select Project → Create new project...
   (b) Select tool chain MSP430.
   (c) Base the project on the empty project template.
   (d) Save (you will also be asked to save the current workspace).

4. Add the following C files from the software that you downloaded and unzipped in step 2:
   (a) All C files from the *code\library* folder
   (b) All C files from the *code\library\TI_MSP430_other_i2c_modules* folder
   (c) All C files from the *code\demo-application-examples\demo-app01* folder
5. Open the "options..." dialog for the new project by right clicking the project name in the workspace window. A window should appear.
   (a) Under "General options", select the MSP device. For the MSP-TS430RGC64USB target board, use MSP430F5528.
   (b) Under "C/C++ compiler", click on the preprocessor pane.
      (i) The "Ignore standard include directories" tick box should not be ticked.
      (ii) In the "Additional include directories", add include paths telling the compiler where to find the header files included by the C files. You should add the *$PROJ_DIR$\code\include* folder.
   (c) Under "Debugger", select "FET Debugger" from the "Driver" drop down list.
   (d) Under "FET Debugger", in the "Connection" section, choose the connection type of the FET tool (e.g. Texas Instruments USB-IF). Leave the rest of the settings as is.
6. Click “OK” to close the options window.
7. Select “Project → Rebuild All”. There should be no errors or warnings when IAR rebuilds the executables (if not done already, you will also be asked to save the current workspace).
8. The configuration of the hardware definition files in the library as distributed by TI is for an MSP430F5528 equipped board. The system variable *TI_LMP91000_SER_INTF* defined within *TI_hardware_board.h* identifies *USCIB0_5xx* as the connected I2C port to control LMP91000. Peripheral pinouts can change slightly between individual MSP430 devices and families. For this reason *TI_MSP430.h* identifies the pins that correspond to a peripheral for any given device.
9. Connect the LMP91000 Evaluation Board I2C interface lines to the MSP430 target board I2C port as shown in Figure 2.

NOTE: If using the new LMP91000EVM, the USCIA1 pins of the MSP430 are used to control the ADC161S626 serial peripheral interface (SPI) pins as shown in Figure 3. Also, note the I2C bus of the LMP91000EVM requires two 10 kΩ pull-up resistors (R1, R2) when used with an external microcontroller. For more information, see reference [10].

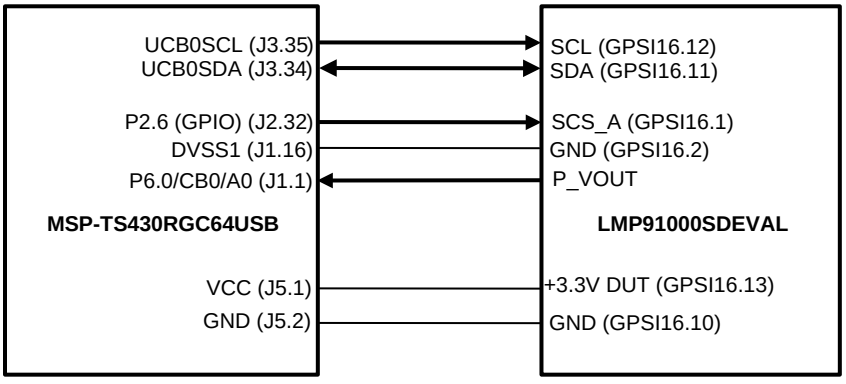

Figure 2. LMP91000SDEVAL to MSP430 Connection Diagram

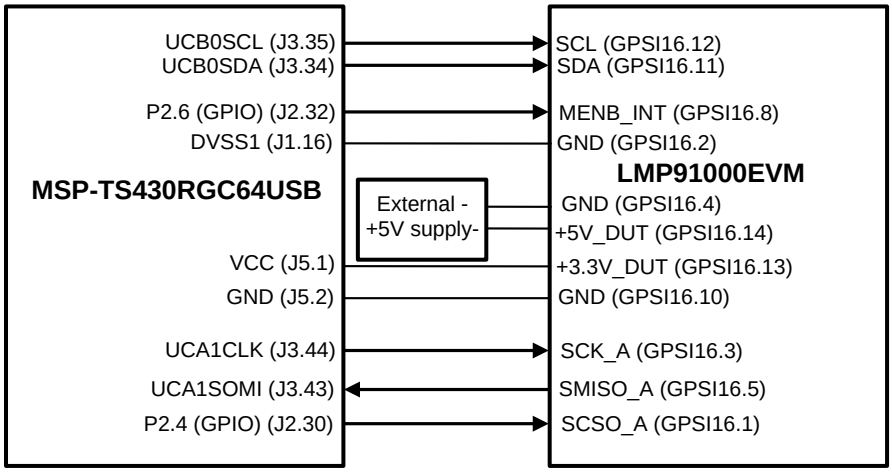

Figure 3. LMP91000EVM to MSP430 Connection Diagram

10. The LMP91000 Evaluation Board jumper connections can be seen in Figure 4 and Table 7. The Gas Sensor connection (Jumpers J_SENSOR_CFG, J_RLOAD_O2, and J_CE_RE_GND) should be made as described in the LMP91000 user's guide. For I2C control, only jumpers J_VDD and J_MENB matter and are described in Table 7.

NOTE: The new LMP91000EVM is shown in Figure 5 with jumper connections.

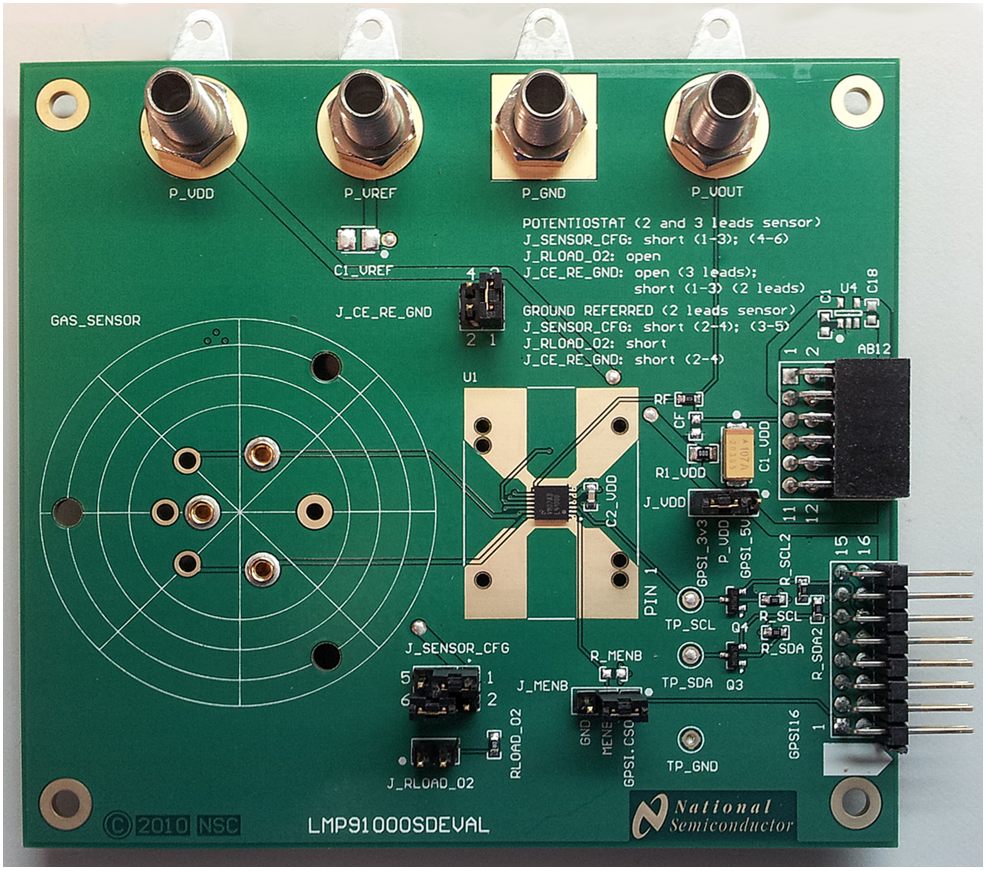

Figure 4. LMP91000SDEVAL Jumper Settings

Table 7. LMP91000SDEVAL Jumper Settings

Jumpers	Pin	Purpose
J_VDD	P2–P3	P_VDD – GPSI_3v3 shorted
J_MENB	P1–P2	MENB – GPSI.CS0 shorted

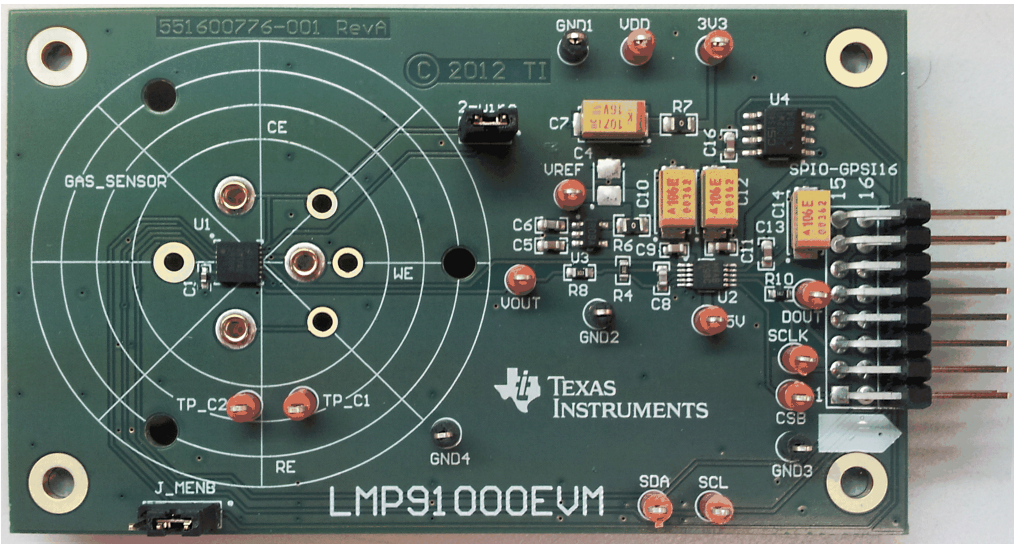

Figure 5. LMP91000EVM

11. Attach the MSP430 FET to your PC. If you are running Windows and using the USB FET tool for the first time, you will asked to install some drivers for the tool. For Windows, they are located in *$IAR_INSTALL_DIR$\430\drivers\TIUSBFET*.
12. Attach the MSP430 FET to the MSP430 target board using the JTAG connector. The V_{CC} power select jumper JP3 should be set to 1-2 (int) where the board is powered from the FET alone. The MSP430 target board jumper connections can be seen in Figure 6.

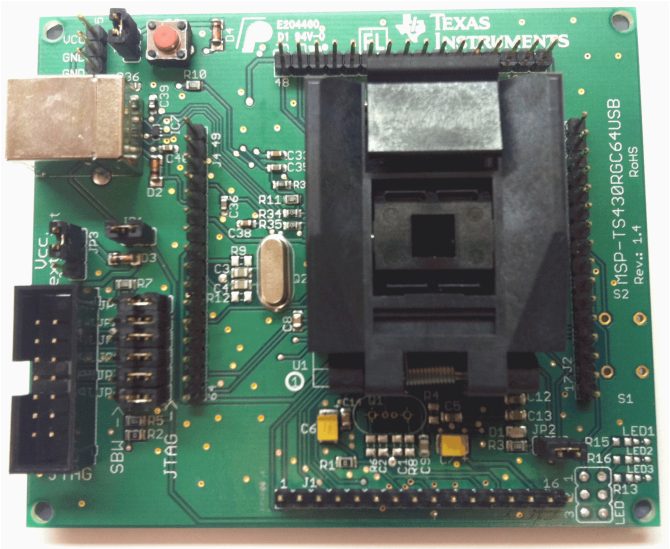

Figure 6. MSP-TS430RGC64USB Jumper Settings

13. Select Project → Debug. IAR will now establish a connection with the target MCU, download the application, and program the MSP430. The debugger will be started, halting the target at *main ()*.

14. Demo_app01 is a simple example that demonstrates the I2C calls to write and successfully read back a LMP91000 register. The onboard LED on the MSP430 target board is setup to blink continuously if the value read back matches the value written.

5.3 Adapting the Demo Project to Other Hardware

The procedure for adapting this code to other hardware is as follows:

1. Edit the pin assignments within *TI_MSP430.h* for the interface modules being used. It is not necessary to modify the pins for the interfaces not selected for use with the I2C bus, as they will not be referenced by the library. The labels being referenced in the *#define* assignments will be drawn from the standard definition file *(msp430.h)* listed at the top of *TI_MSP430.h*.
2. Edit the pin assignments in *TI_MSP430_hardware_board.h*, taking into account all the necessary connections on the board being used. The assigned labels are drawn from the standard definition file *(msp430.h)* listed at the top of *TI_MSP430.h*.
3. Assign the proper values to *TI_LMP91000_SER_INTF* in *TI_MSP430_hardware_board.h*. The labels available for assignment can be found at the bottom of *TI_MSP430.h*.
4. Appropriately set up the function to configure the system clock source and clock rate. This will depend on your hardware and the particular MSP430 MCU in use.
5. Appropriately set up the function to configure the onboard ADC to convert LMP91000 output. This will depend on your hardware and the particular MSP430 MCU in use.
6. Make sure the physical hardware connections between the MSP430 target board and LMP91000SDEVAL are modified according to the pin assignments above.

After making these changes, rebuild the project and download the code image. The application should function as described earlier.

5.4 Using the Library With an Application

The same procedure as described in Section 5.3 should be applied in order to adapt the library to the new hardware.

The function*TI_LMP91000_I2CSetup* should always be called after a POR event within the MSP430. After this the access of registers is straightforward.

6 References

1. *LMP91000 Sensor AFE System: Configurable AFE Potentiostat for Low-Power Chemical Sensing Applications Data Sheet* (SNAS506)
2. LMP91000 Evaluation Development Platform Software
3. LMP91000 Evaluation Board User Guide
4. *MSP430F551x/MSP430F552x Mixed Signal Microcontroller Data Sheet* (SLAS590)
5. *MSP430x5xx/MSP430x6xx Family User's Guide* (SLAU208)
6. MSP430F55xx 64-Pin Target Board MSP-TS430RGC64USB
7. MSP430 USB Debugging Interface MSP-FET430UIF
8. *Interfacing an EEPROM via I2C Using the MSP430* (SLAA208)
9. *LMP91002 Sensor AFE System: Configurable AFE for Potentiostat for Low-Power Chemical Sensing Applications Data Sheet* (SNIS163)
10. *LMP91000EVM User's Guide* (SNAU121)

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		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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