LM53X

Bluetooth Smart Development and Evaluation Product Family

User Guide

© All rights reserved.
All trade names are registered trademarks of respective manufacturers listed.
This manual may not be copied in any media or form without the written consent of original maker.

Contents

1. Introduction

1.1. Overview

This user guide explains how to use the LM930/ LM931 Bluetooth Smart Module on the LM53X development kit. A good starting point to developing your own user application firmware within the provided CSR µEnergy SDK. LM offer project support including developing new application firmware to meet your project requirements.

1.2. Out of the box

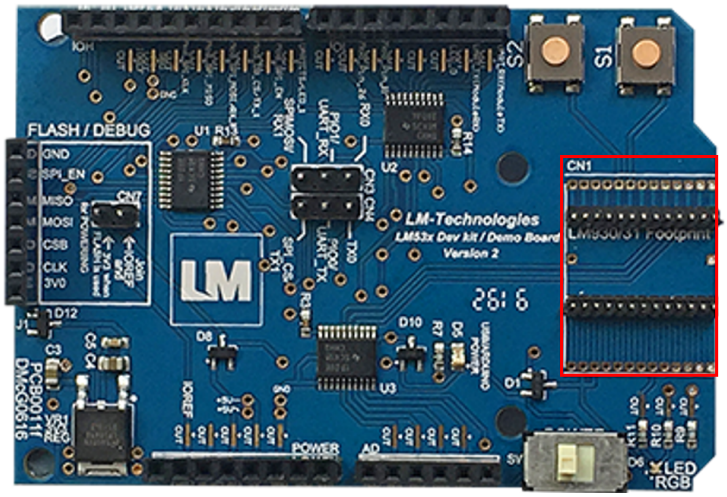
- CSR toolchain (including CSR µEnergy SDK (xIDE))
- LM930/ LM931 Bluetooth Smart module
- LM53X (including the CSR USB-SPI Programmer Board)
- USB to mini USB cable (Not Included)
- Jumpers (x3)
- CR2032 Coin Cell Battery

2. LM53X Hardware

2.1. Mounting the LM930/ LM931 to the LM53X

The LM930/ LM931 module is mounted in one of 2 ways to the LM53X:

1) Soldered to the Pads on CN1 (for a permanent connection).
2) Connected via 1.27 mm pitch pins that is soldered to CN1 (for a temporary connection).*

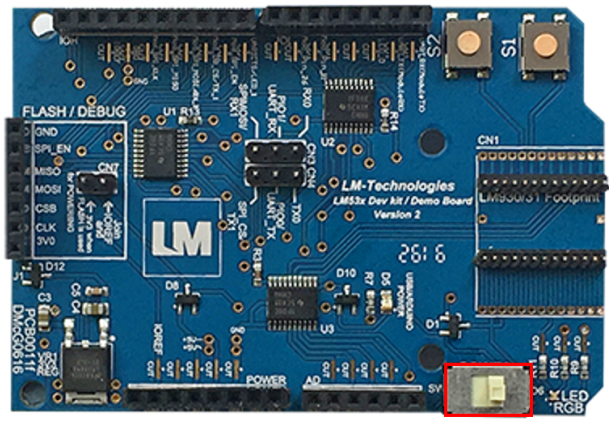


*Note: for connection (option 2) the pins need to be angled slightly inwards and ideally some soft foam can be placed under the Module (to ensure some upward force is applied to the module and a good connection is made between the module and the pins).

2.2. Powering the LM53X

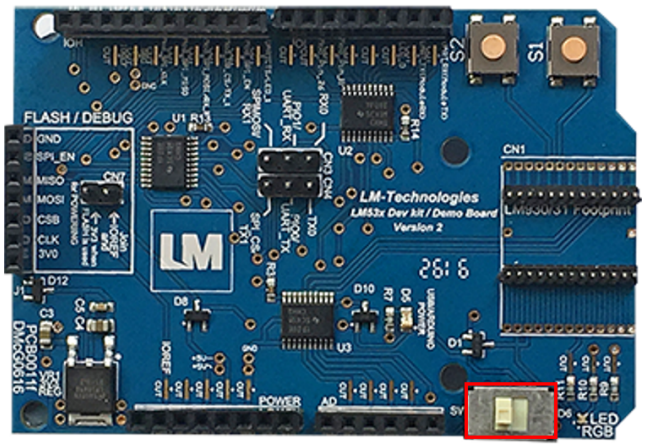
The LM53X is powered in one of 3 ways:

1) USB powered from the CSR USB-SPI Programmer Board (power switch positioned to the right in RED)



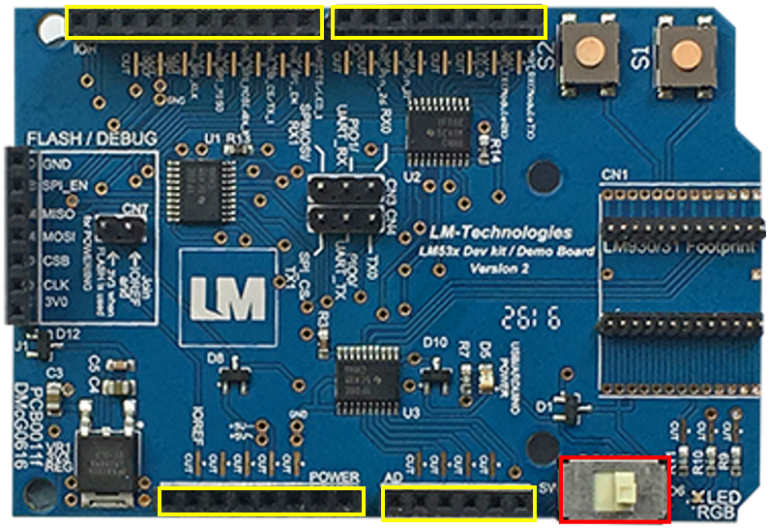
This is a convenient powering option. Requiring a USB to mini USB cable to connect the programmer board to the development PC.

2) Battery Power (power switch positioned to the left in RED)



The battery connector is situated underneath the board and accepts a CR2032 Coin Cell Battery.
This option is useful when the developer wants to test their software in a “real world” powered situation.
Note: No jumpers should be connected to CN3, CN4 and CN7.

3) Powered from Arduino board (power switch positioned to the right in RED)



This option allows the LM53X to be mounted directly on and powered by an Arduino Uno board.
The 4 edge connectors (Marked with yellow rectangles) needs to be mounted as if the LM53X is an Arduino “Shield” with the pins protruding from the bottom of the PCB.

2.3. Connecting External Devices

2.3.1. Connecting an Arduino Board

Note: Using an Arduino is optional.

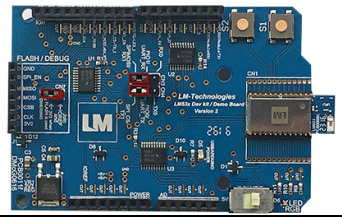
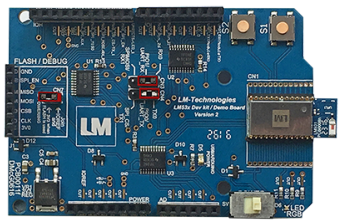
The Arduino microcontroller is connected to the LM930/ LM931 via the LM53X. The LM930/ LM931 effectively becomes a slave device to the Arduino. LM53X is simply connected into the female headers of the Arduino, by soldering male pins under the LM53X. The same way an Arduino shield is connected. When the Arduino and the module are connected, power the LM53X in one of the three ways described in the previous section.
Note: The Arduino needs its own power supply.

The Arduino can program the module using either a SPI or a UART interface.

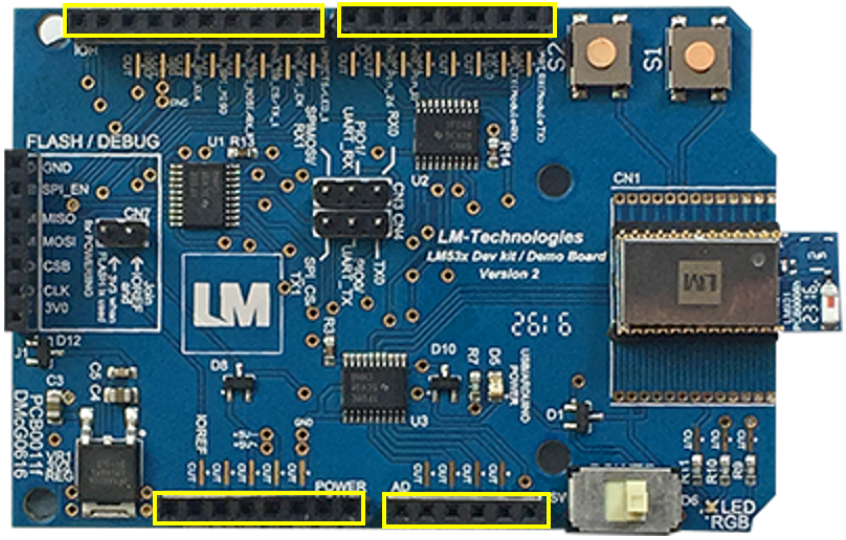
For SPI mode, the LM930/ LM931 SPI pins are always connected to the Arduino General I/O pins as follows:

Arduino Pins	LM930/ LM931 Pins
PB1 (IOH2/IO9 on Shield)	SPI_ENABLE (PIN 25)
PB2 (IOH3/IO10 on Shield)	SPI_CS (PIN 18)
PB3 (IOH4/IO11 on Shield)	SPI_MOSI (PIN 19)
PB4 (IOH5/IO12 on Shield)	SPI_MISO (PIN 20)
PB5 (IOH6/IO13 on Shield)	SPI_CLK (PIN 12)

For UART mode, the LM930/ LM931 UART pins (TX (PIN 8) & RX (PIN 9)) are connected to the Arduino I/O pins in one of two ways, with the CN3/ CN4 jumpers positioned either LEFT or RIGHT:

CN3/ CN4 Jumper Positions	Arduino Pins	LM930/ LM931 Pins
LEFT (shown in RED) 	PB2 (IOH3/IO10 on Shield)	TX (LM930/ LM931 PIN 8)
	PB3 (IOH4/IO11 on Shield)	RX (LM930/ LM931 PIN 9)
RIGHT (shown in RED) 	RX0 (UART_RX (LM930/ LM931_TX)) IOL1/ IO0	TX (LM930/ LM931 PIN 8)
	TX0 (UART_TX (LM930/ LM931_RX)) IOL2/ IO1	RX (LM930/ LM931 PIN 9)

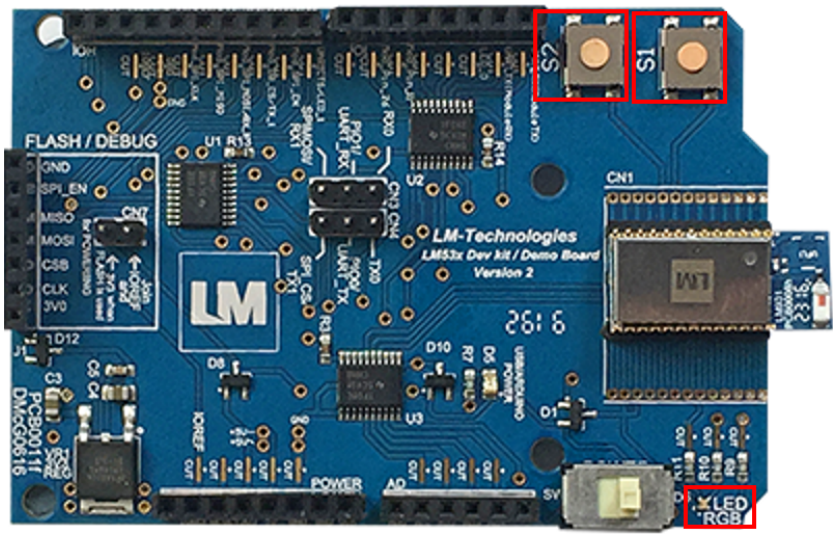
2.3.2. Connecting Peripheral Devices



I²C is used to connect peripheral devices (e.g. sensors) to the LM930/ LM931 module via the LM53X. The peripheral device is connected to the

corresponding female header holes (highlighted in YELLOW). When the module and peripheral devices are connected, power the LM53X in one of the three ways described in the previous section.

2.4. User Definable Switches and RGB LED



The LM53X has two inbuilt switches and an RGB LED highlighted in RED. These are all user programmable, depending on the Bluetooth LE application been developed.

Two general purpose switches S1 and S2 are connected to P10_0/ UART_TX (Pin 8) and P10_1/ UART_RX (Pin 9) respectively.

A RGB LED has Red connected to LED_0 (Pin 22), Green connected to SDA (Pin 23) and Blue connected to SCL (Pin 24).

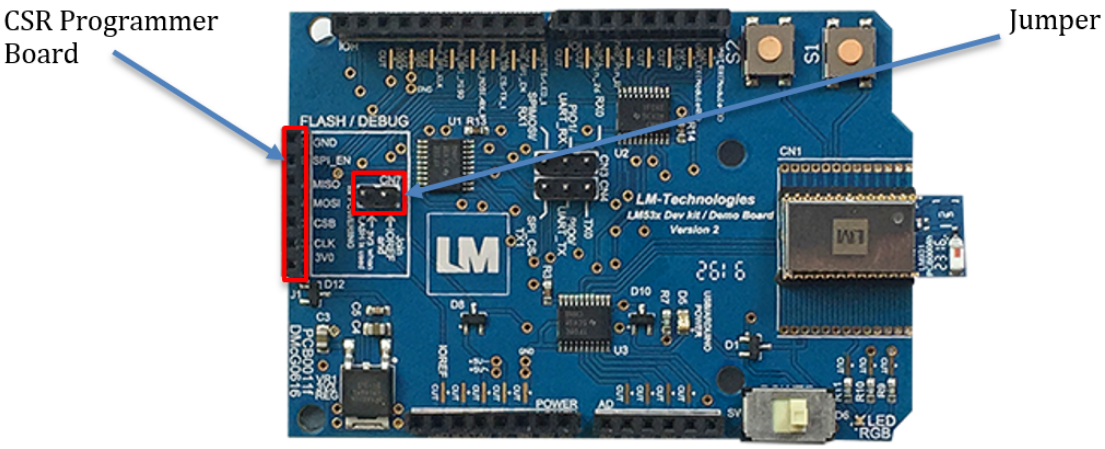
Use the PIO for any additional LEDs, switches and other I/O devices required.

3. Bluetooth LE Firmware Development

3.1. Using the CSR µEnergy SDK

For modifying the provided LM demo application firmware and developing new application firmware, you must follow the steps below:

1. Install CSR µEnergy SDK (xIDE) toolchain on the PC you are developing with. (The installation includes the SDK, support documentation and necessary drivers).
2. Connect the LM930/ LM931 Bluetooth Smart module to CN1 of the LM53X development kit.
3. Plug a jumper to CN7 of the LM53X development kit (this allows the LM53X development kit and LM930/ LM931 module to be powered from the connected PC).
4. Plug the CSR USB-SPI Programmer Board to J1 header of the LM53X and then use the USB to mini USB cable to connect the CSR USB-SPI Programmer Board to the PC you are developing on.



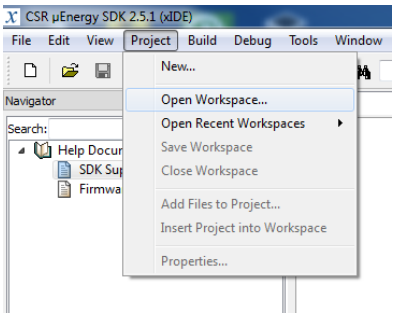
3.2. Flashing a Demo Application Image

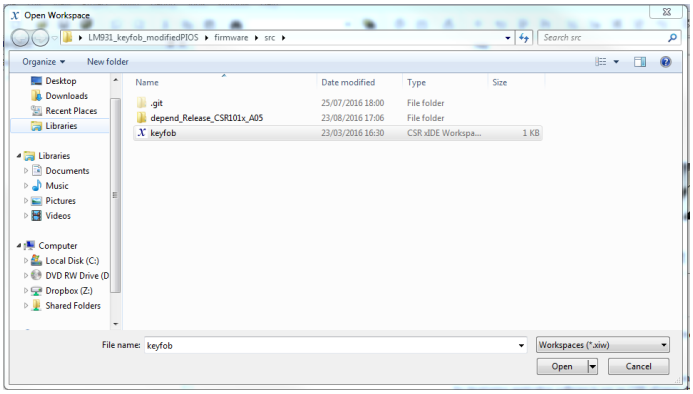
Demo applications are compiled and the resulting image is ready to be flashed into the LM930/ LM931 module EEPROM memory. This is interfaced with the MCU through the I2C bus.

Execute the following steps to flash the demo application image to the LM930/ LM931 memory using the CSR µEnergy SDK:

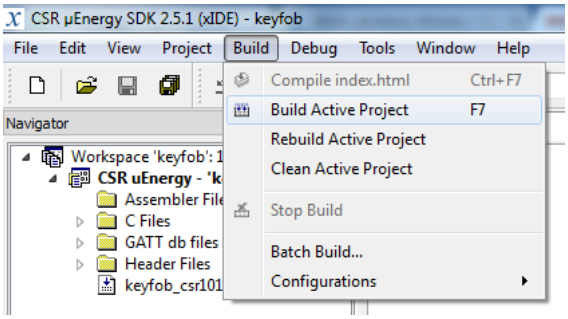
1/ Download the required LM demo application zip provided e.g. ‘Key Fob V1.0’. This includes the source files (.c, .h, etc.).

2/ Open the workspace of the demo application in the CSR µEnergy SDK by clicking the project tab and then ‘Open Workspace’.

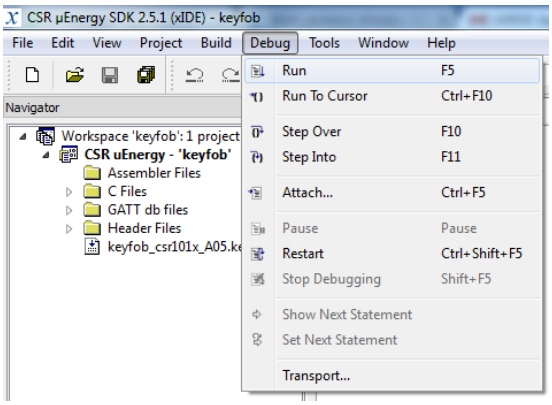




3/ Build the active project by clicking the build tab and then ‘Build Active Project’.



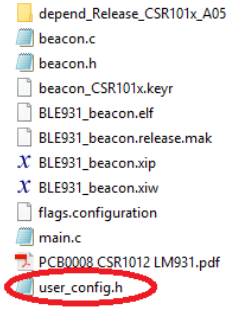
4/ Download the application to the module by clicking the debug tab followed by ‘run’.



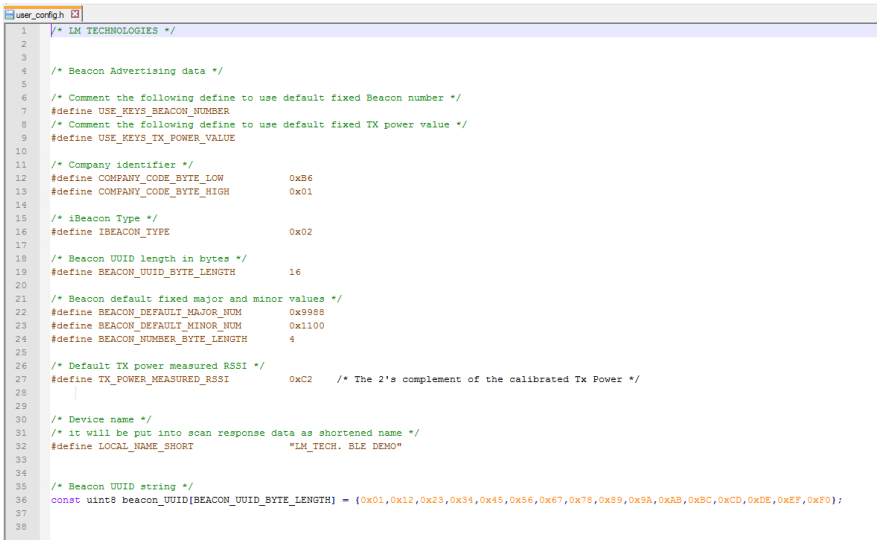
3.3. Customising Demo Application Firmware

Demo applications are configured by changing the parameters and values according to the demo application behaviour. In every src folder there is a header file named user_config.h which contains configurable code definitions. Refer to the specific demo application file for more details as the definitions vary depending on the related application.

iBeacon example:



The file below contains all the user customisable definitions:



After modifying desired definitions you compile the project again using the CSR µEnergy SDK. A new image file is created to be flashed into the memory by using the SDK. For more details about µEnergy SDK and the entire toolchain please refer to the CSR documentation and support.

3.4. Developing Application Firmware

For developing a new application refer to the CSR documentation provided with the SDK. The CSR µEnergy SDK allows you to develop new applications. That can be compiled and debugged seamless on the LM930/ LM931 module. The LM demo applications are a good starting point for new applications, saving the time required to develop an application from scratch.

It is necessary to modify the CS keys included in LM demo applications to fit your application requirements e.g. the BD address.

CS keys is modified in one of two ways:

1. Modify the XXXX.keyr file. Build the project and then download the image file.
2. By using “Cs_Config tool” available in the SDK

8 “users keys” is used in your application to store application specific settings in the persistent storage.

```
 5
 6// (0002) - Crystal frequency trim
 7&CRYSTAL_FTRIM = 0014
 8
 9// (0007) - Transmit power level
10&TX_POWER_LEVEL = 7
```

Note: &CRYSTAL_FTRIM and &TX_POWER_LEVEL should not be modified by the user.