



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: [info@chipsmall.com](mailto:info@chipsmall.com) Web: [www.chipsmall.com](http://www.chipsmall.com)

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



## 1 Important notice

---

NXP provides the enclosed product(s) under the following conditions:

This evaluation kit is intended for use of ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY. It is provided as a sample IC pre-soldered to a printed circuit board to make it easier to access inputs, outputs, and supply terminals. This evaluation board may be used with any development system or other source of I/O signals by simply connecting it to the host MCU or computer board via off-the-shelf cables. This evaluation board is not a Reference Design and is not intended to represent a final design recommendation for any particular application. Final device in an application will be heavily dependent on proper printed circuit board layout and heat sinking design as well as attention to supply filtering, transient suppression, and I/O signal quality.

The goods provided may not be complete in terms of required design, marketing, and or manufacturing related protective considerations, including product safety measures typically found in the end product incorporating the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge. In order to minimize risks associated with the customers applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards. For any safety concerns, contact NXP sales and technical support services.

Should this evaluation kit not meet the specifications indicated in the kit, it may be returned within 30 days from the date of delivery and will be replaced by a new kit.

NXP reserves the right to make changes without further notice to any products herein. NXP makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Typical parameters can and do vary in different applications and actual performance may vary over time. All operating parameters, including Typical, must be validated for each customer application by customer's technical experts.

NXP does not convey any license under its patent rights nor the rights of others. NXP products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the NXP product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use NXP products for any such unintended or unauthorized application, the Buyer shall indemnify and hold NXP and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or





unauthorized use, even if such claim alleges NXP was negligent regarding the design or manufacture of the part.

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © NXP B.V. 2018.

## 2 Introduction

The main intention of this user guide is to enable engineers to set up OL2385 based SIGFOX boards for testing. This user guide explains how to use the OL2385 transceiver for rapid prototyping of microcontroller based SIGFOX applications. It also contains a basic introduction to the SIGFOX ecosystem, an explanation regarding necessary registration of devices at SIGFOX network for demo evaluation and basics of SIGFOX P1 certification testing. This documentation also describes how to install and use the SIGFOX hardware/software setup.

The OM2385/SF001 development kit is an evaluation platform based on the OL2385 chip. See the [OL2385 data sheet](#) for detailed information.

## 3 SIGFOX network architecture

SIGFOX is an operated telecommunication Low-Power Wide-Area (LPWA) public network, dedicated to the Internet of Things. This telecommunication technology has been created with the sole focus on small messages that are transmitted on the radio only very few times per day (12 bytes messages uplink up to 140 times per day and 8 bytes messages downlink up to 5 times per day). It is not appropriate for high-bandwidth applications, such as multimedia and permanent broadcast. The SIGFOX network operates at sub-GHz frequencies on ISM bands, for example, 868 MHz in Europe/ETSI and 902 MHz in the US/FCC. This Ultra-Narrow Band (UNB) modulation based technology has 162 dB budget link, which enables long range communications. These SIGFOX messages, containing a payload of up to 12 bytes, are transmitted by a radio transceiver chip into the SIGFOX network, which is then received by a SIGFOX base station. NXP plays the role of this transceiver chip provider in the SIGFOX ecosystem.

There are four roles in the system:

- **Modem manufacturers** provide transceiver chips running with a SIGFOX software protocol stack and identified by a unique id/private key (used for encrypting the radio messages). These transceivers will be integrated in final modules that are sold to end customers. The modules must have a network registration containing the following information:
  - ID
  - Porting authorization code (PAC), one time use only
  - Certificate number given by the product manufacturer (P\_XXXX\_XXXX\_XX)

The trio of key, ID and PAC is provided by SIGFOX to modem manufacturers. These manufacturers flash this information in nonvolatile memory of their IC during production. The ID and PAC can be retrieved by the user and thus are sharable entities. However, the encrypted key should not be shared, visible, or retrievable by anyone. It must reside inside a protected area of memory where it is only readable by the firmware for encryption purposes only. It should not be writable by anyone after production.

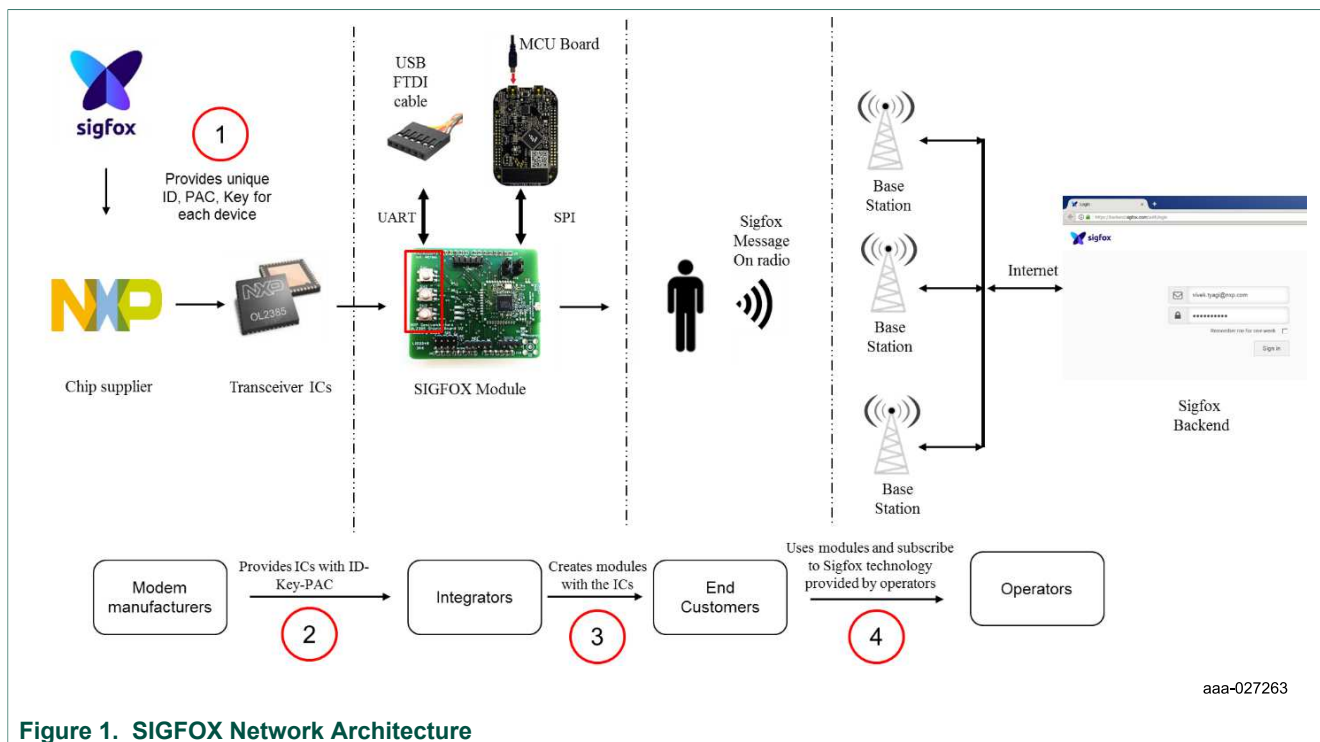


Figure 1. SIGFOX Network Architecture

- **Operators** operate their SIGFOX networks (base stations and backend system).
- **Integrators** build end-user systems or modules that will have manufacturer's transceivers on them with a SIGFOX protocol stack inside.
- **End customers** buy and use these systems or modules that include SIGFOX transceivers. each end customer will also have to buy a subscription to SIGFOX technology from an operator.

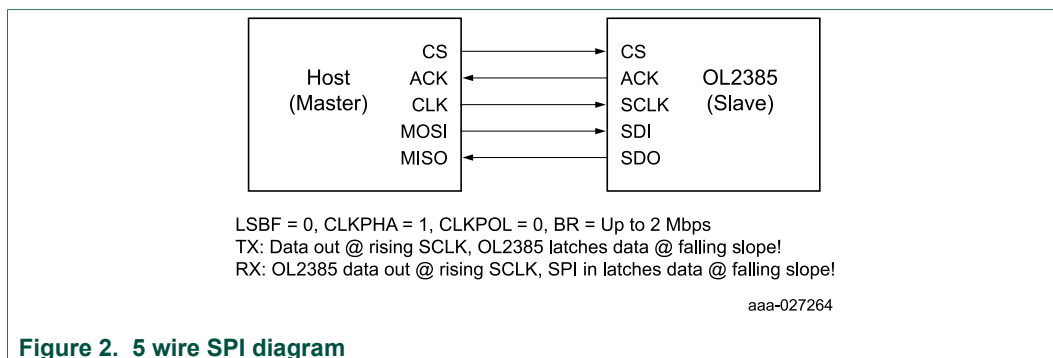
## 4 External/User interfaces to OL2385

### 4.1 Host MCU interface through SPI

A 5-wire SPI connection is defined between OL2385 board and Host microcontroller where host is configured to be the Master and always provides the clock. OL2385 is initiated in Pseudo-Slave mode. The five essential lines required for such interface is shown in [Figure 2](#).

#### Reason for extra pin Ack

The host microcontroller can be sometimes faster than the slave device and can send the data while slave is not yet ready to receive. This might result in communication error due to nonsynchronization. For example, at slave device few initial clocks could be missed from the host, because it was not ready to receive. Therefore, each of the frame transfers between host and OL2385 is completely controlled via a software driven chip select CS and Ack pins. The protocol and timing diagram makes sure that the handshake is properly done and none of the bytes are missed, as shown in the following sections.



**Figure 2. 5 wire SPI diagram**

A messaging protocol has been defined for SPI interface between OL2385 and host microcontroller. The protocol governs the states of each of the five SPI interface pins, as shown in [Figure 3](#). **It is very important to note here that Host is the master and always the initiator of the communication, which means that there cannot be an independent transfer of bytes from OL2385 to host.** The communication will take place always by transfer of bytes from host to OL2385, and if a response is required back only then data transfer will take place from OL2385 to host. The cases in which such a response is desired is explained in further in [Section 6.2.4.4 "SPI command descriptions"](#). The sequence of pin driving should take place strictly in the following manner:

**Note: By default, after a reset, OL2385 goes to low-power mode with CS pin configured as wakeup pin. Before proceeding to the low-power mode, OL2385 configures all its pins to pulled-down configuration, except for CS and Ack pins of SPI. CS and Ack pins are pulled high. For the low-power mode to work properly without any current leakage, it is important to have the same settings of the pins on MCU side. All pins on MCU should be pulled low and only CS and Ack should be high.**

1. Initially, the CS and Ack pins should be driven high on the MCU side. OL2385 as a slave waits for the CS pin to be driven low by the host to start communication.
2. Host drives CS pin low to signal data transfer.
3. OL2385 drives Ack pin low to signal that it is ready for clocking and data transfer.
4. Host provides clock and writes data on MOSI
5. **OL2385 reads the first byte of data. This is the length byte that describes how many valid data bytes are to be followed on SDI, including itself.** After receiving all the bytes, OL2385 drives the Ack pin high to signal data is received successfully.
6. Host drives CS back to high again to signal data transfer from host to OL2385 is complete.

The data is now decoded at OL2385 and processed according to its meaning. If a response is required to be submitted back to host after processing, the following sequence should then be initiated from step number 7:

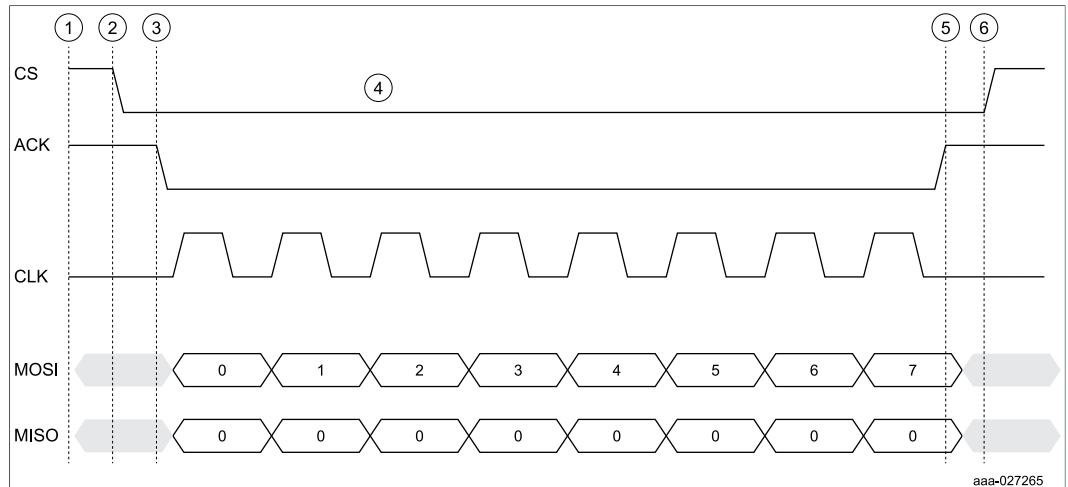


Figure 3. SPI protocol: Host to OL2385 transfer

1. CS and Ack pins are high after data transfer from host to OL2385.
2. OL2385 drives Ack pin low to signal the host that OL2385 has data to transfer as response.
3. Host drives CS pin low to signal host is ready to provide clock for such a transfer.
4. Host provides clock and data is written on SDO by OL2385.
5. **Host reads the first byte of data. This is the length byte that describes how many valid data bytes are to be followed on SDO, including itself.** After receiving all the bytes on MISO, the host waits for the Ack pin to be driven high and OL2385 drives the Ack pin high to signal that data is now completely sent.
6. Host drives CS back to high again to signal data reception is complete from OL2385 to host.

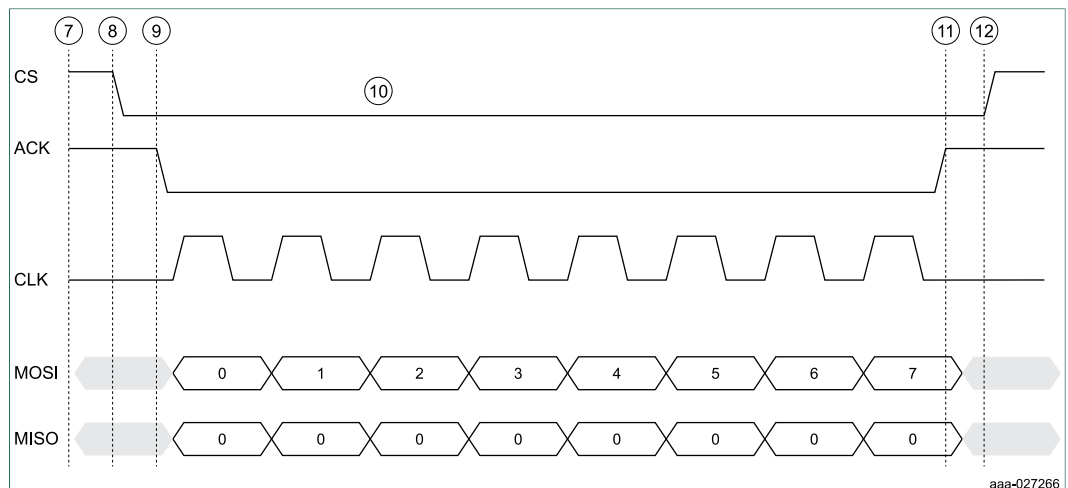


Figure 4. SPI protocol: OL2385 to host transfer

### SPI commands

The bytes received or sent over the SPI interface must also follow a predefined format and order as per the SPI protocol. The bytes sent from host to OL2385 are called as command frames or information frames. The bytes sent from OL2385 to host are called as Ack frames. The format follows a similar pattern for both of these types of frames, as shown in [Figure 5](#). The command frames contain a command from a set of allowed

commands coded in the second byte, which is called *Command code*. An action is taken by OL2385 as per the designed meaning of such code.

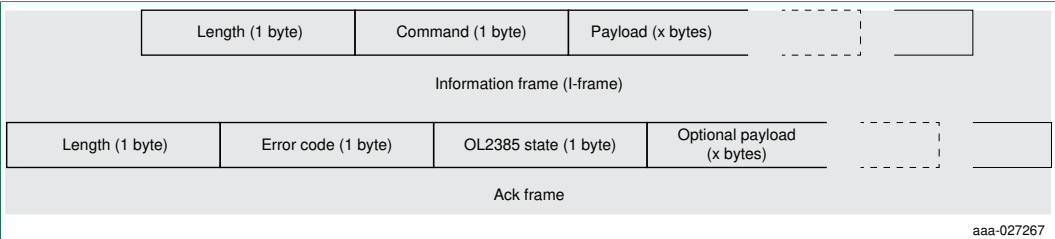


Figure 5. SPI frame types

The first byte of each frame type indicates the length of the frame to be received, including itself. A length of 4 means 3 more bytes are going to follow after the first byte. The payload field in both types of frames is optional. For details of each command, refer to the SIGFOX SPI Command Interface Description.xls file at:

[www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/om2385-sf001-ol2385-wireless-sub-ghz-transceiver-sigfox-development-kit-with-kl43z:OM2385-SF001?tab=Documentation\\_Tab](http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/om2385-sf001-ol2385-wireless-sub-ghz-transceiver-sigfox-development-kit-with-kl43z:OM2385-SF001?tab=Documentation_Tab)

The data bytes between OL2385 and the host do not only just follow the format, they also follow a particular sequence, depending on the state of the device, as shown in the message sequence chart in [Figure 6](#).

Data transfer protocol over SPI

Because the frequency of Sigfox messages is very low per day, the OL2385 device generally remains in power-down SLEEP mode under normal conditions.

1. The device is always awakened by sending the Send wakeup (2 byte, fixed length) frame from the host master.
2. On receiving such frame, an OL2385 device performs a wake-up reset and moves to a ready state in 100 milliseconds.
3. The host should wait for the 100 ms wake-up to complete before sending the further frames. No acknowledgment is sent from OL2385 after wake up.
4. Once the device wakes up, it is ready to receive command frames from the host.
5. An Information frame or I-frame is used to send a command to OL2385.
6. An Ack frame is used to send back a response, if required.

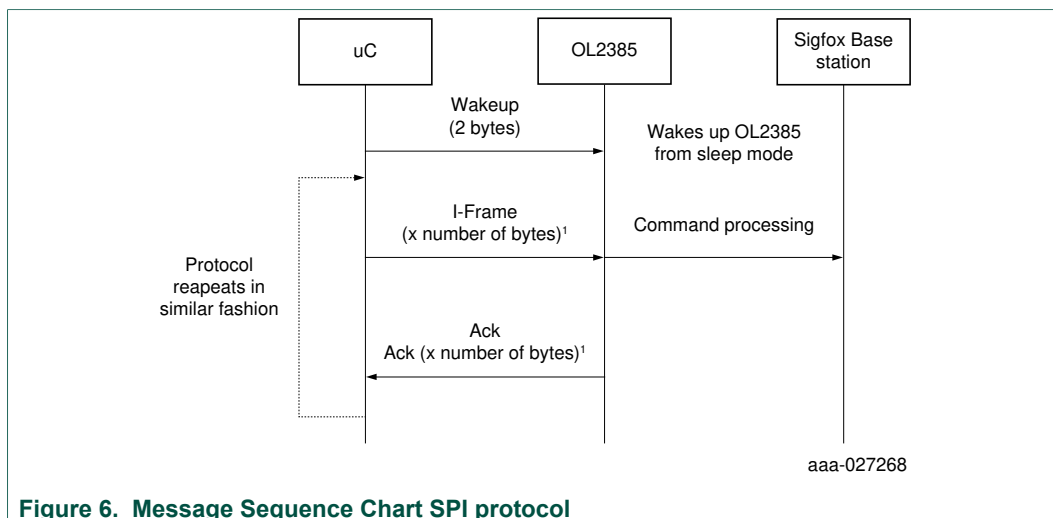


Figure 6. Message Sequence Chart SPI protocol

## 4.2 Button pressed based interface

To be implemented fully in software

## 4.3 UART interface

To be implemented fully in software

# 5 Setting up the hardware

The connection of any OL2385 board with any microcontroller is based on connecting the SPI pins, power supply pins and RESET pin. The positions of SPI pins on OL2385 is fixed and explained in the following sections. The positions of SPI and power pins on any other microcontroller board has to be read out from its corresponding user guide. An example of such connection with KL43Z pins is given below. The NXP reference design is based on the Arduino style of pin layout. Therefore, NXP reference design boards can be plugged directly on top of a KL43Z board as shown in [Figure 11](#). The SPI, RESET and Power pins will match automatically. If your microcontroller is not Arduino style, you will have to connect the pins of the OL2385 with wires to your microcontroller's corresponding pins.

## 5.1 Overview of the OM2385/FS001 development kit

The OM2385/FS001 development kit provides an evaluation platform for designing SIGFOX network applications that use NXP's OL2385 single-chip RF transceiver. The kit consists of three boards: an OL2385 shield board, OL2385 reference design board and FRDM-KL43Z board. The OL2385 reference design board is permanently affixed to the surface of the OL2385 Shield Board. The reference design board contains an embedded OL2385 transceiver and serves as a wireless modem.

When connected to an antenna, included in the kit, the board provides all the functionality required to communicate with the SIGFOX network. The OL2385 shield board contains connectors for external communication. The Shield Board is mounted by means of four Arduino connectors to the FRDM-KL43Z. The FRDM-KL43Z acts as the communication



link between the development kit and a PC. It comes preloaded with microcode that manages the interface between the PC and the OL2385 reference board. Users must initially register their device with SIGFOX using a unique ID and access code provided with the kit. Once the device has been registered, the kit can be used to connect to the SIGFOX network and test the functionality of the OL2385-based application under development.

To interact with the development kit, users must connect the kit to a PC through the OpenSDA port on the FRDM-KL43Z. A terminal emulator, such as HyperTerminal, provides the interface, allowing users to log in to the network and send and receive messages. Designers can also use the Kinetis Design Studio (KDS) or MCUXpresso IDE to develop and download microcode to the KL43Z.

## 5.2 Getting started

NXP's analog product development boards provide an easy-to-use platform for evaluating NXP products. The boards support a range of analog, mixed-signal and power solutions. They incorporate monolithic ICs and system-in-package devices that use proven high-volume technology. NXP products offer longer battery life, a smaller form factor, reduced component counts, lower cost and improved performance in powering state-of-the-art systems.

The tool summary page for the OM2385/SF001 development kit is located at [www.nxp.com/OM2385](http://www.nxp.com/OM2385). The **Overview** tab provides an overview of the device, product features, a description of the kit contents, a list of (and links to) supported devices, list of (and links to) any related products and a Get Started section.

The **Get Started** section provides links to everything needed to start using the device and contains the most relevant, current information applicable to the FRDM-PF1550EVM.

- Go to [www.nxp.com/OM2385](http://www.nxp.com/OM2385)
- On the **Overview** tab, locate the **Jump To** navigation feature on the left side of the window.
- Select the **Get Started** link.
- Review each entry in the **Get Started** section and download an entry by clicking on the title.
- After reviewing the **Overview** tab, visit the other product related tabs for additional information:
  - **Documentation**: download current documentation
  - **Software & Tools**: download current hardware and software tools
  - **Buy/Parametrics**: purchase the product and view the product parametrics

After downloading files, review each file, including the user guide which includes setup instructions. If applicable, the bill of materials (BOM) and supporting schematics are also available for download in the **Get Started** section of the **Overview** tab.

### 5.2.1 Kit contents/packing list

The OM2385/SF001 development kit contents include:

- Assembled and tested OM2385/SF001 FRDM board mounted to a firmware-loaded FRDM-KL43Z board
- Antenna with attached micro-FL connector
- Standard A (male) to mini B (male) USB cable

- Quick start guide

### 5.2.2 System requirements

The kit requires the following to function properly with the software:

- USB enabled computer running Windows XP, Vista, 7, 8, or 10 (32-bit or 64-bit)
- Terminal emulation software (such as HyperTerminal)

## 5.3 Getting to know the hardware

### 5.3.1 SF001 board overview

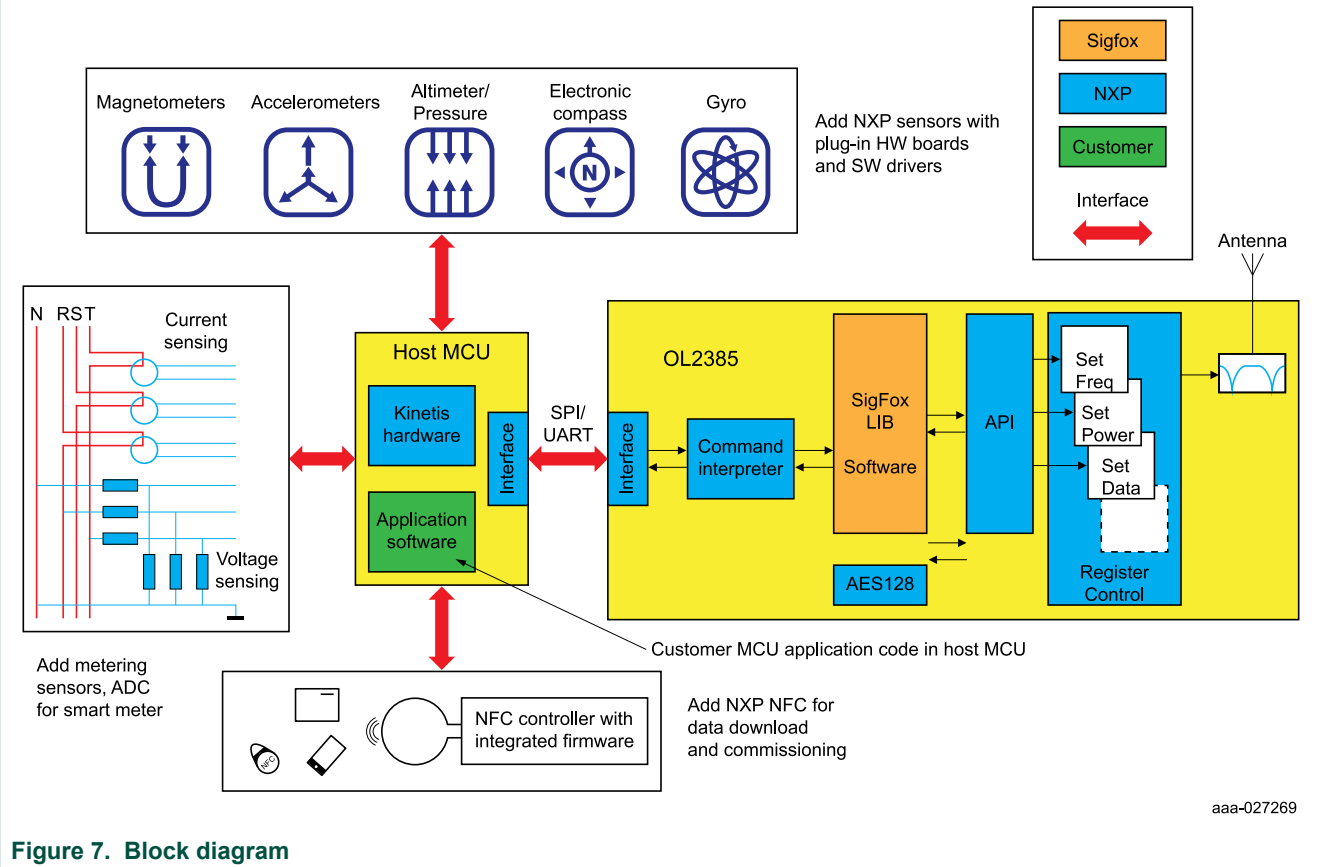
The OM2385/SF001 consists of a base board (the OL2385 shield board) with a permanently attached daughter board (the OL2385 reference design board). The combination, along with the attached FRDM-KL43Z board, serves as a development platform that provides wireless modem access to the SIGFOX network. Once properly registered, the board allows users to send and receive messages across the network.

### 5.3.2 Board features

The board features are as follows:

- Arduino connector compatibility with other Freedom boards
- Support for UART, SPI, MDI and GPIO communication
- SIGFOX Communication Library

5.3.3 OM2385/SF001 Block diagram



5.3.4 OL2385 reference design: Board description

[Figure 8](#) describes the main elements on the OM2385/SF001 board.

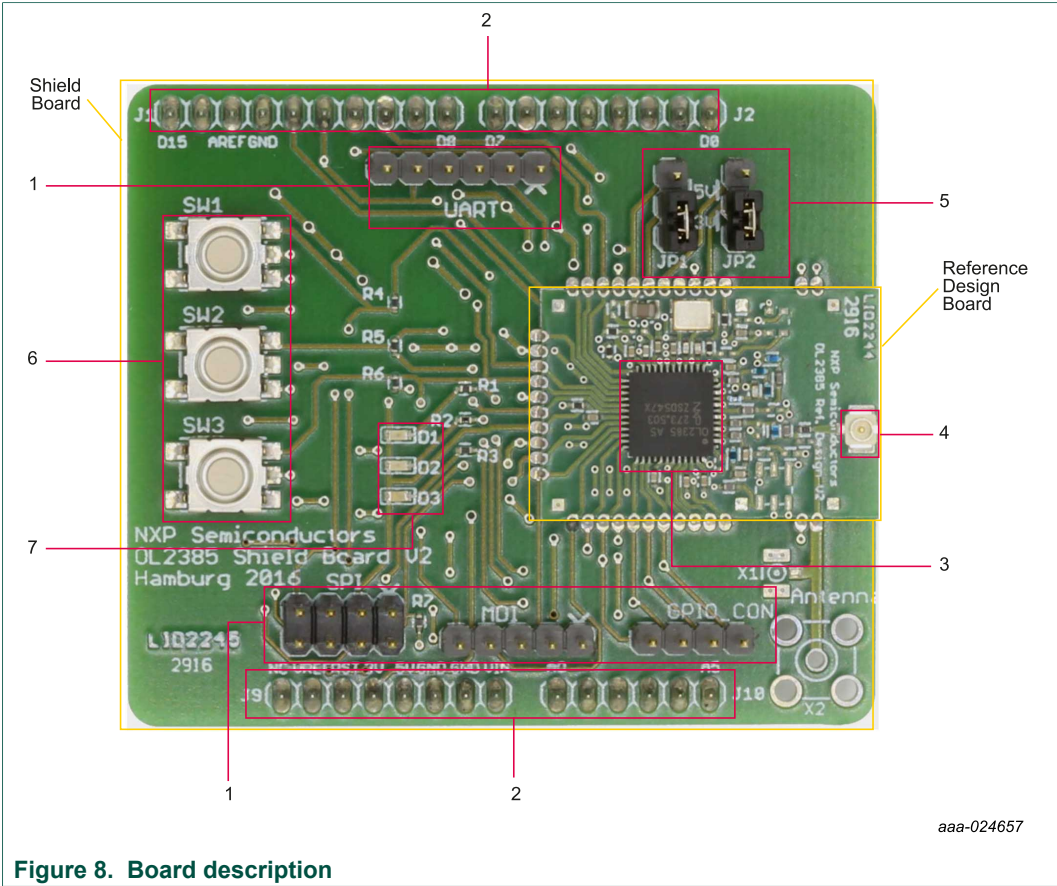


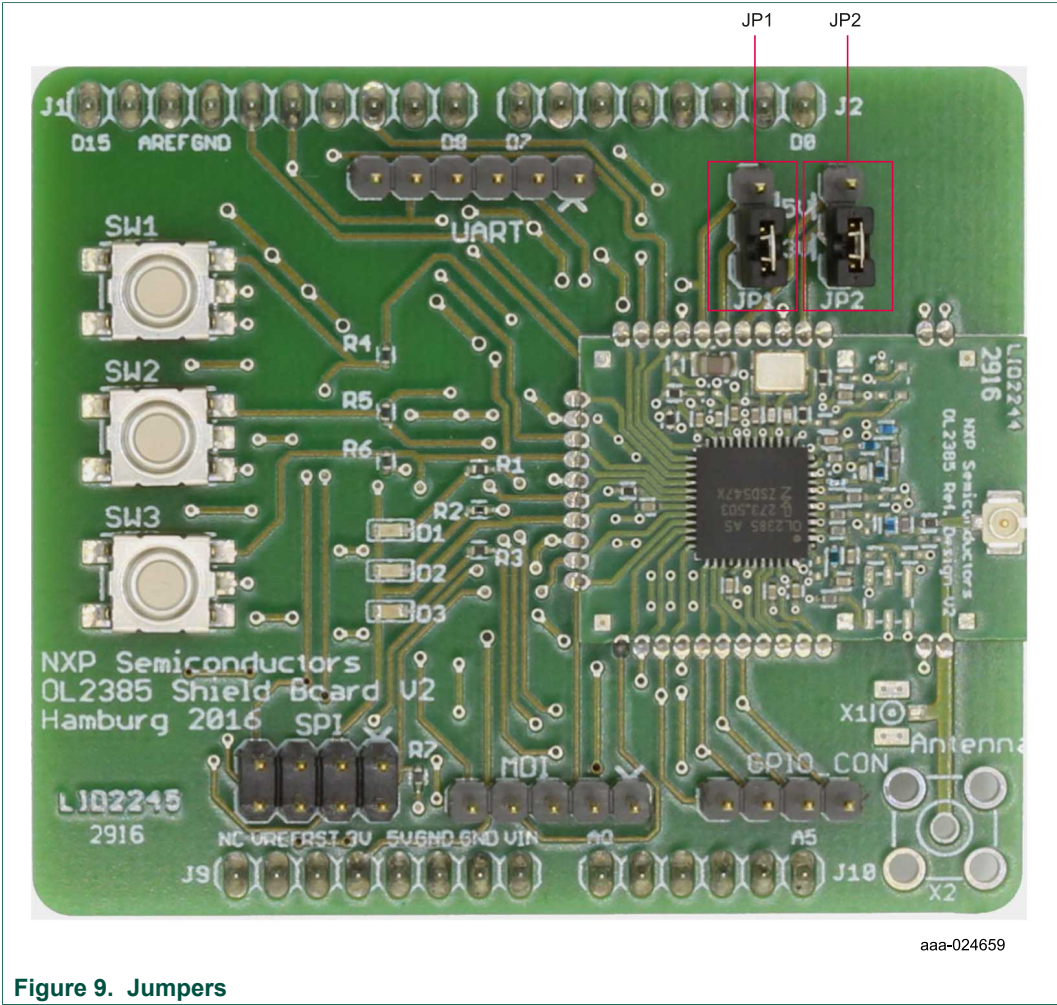
Table 1. Board description

Number	Name	Description
1	Communication connectors	Provide connectivity for SPI, MDI, GPIO and UART support
2	Arduino connectors	Provide connectivity to FRDM-KL43Z and other Freedom boards
3	OL2385	Low-power multichannel UHF RF wireless platform
4	SMA connector	Provides connectivity to UHF antenna
5	Jumpers	Select board voltage levels
6	Button switches	Control digital inputs to Arduino connectors
7	LEDs	Indicate status

5.3.5 OL2385 ref design board jumper definitions

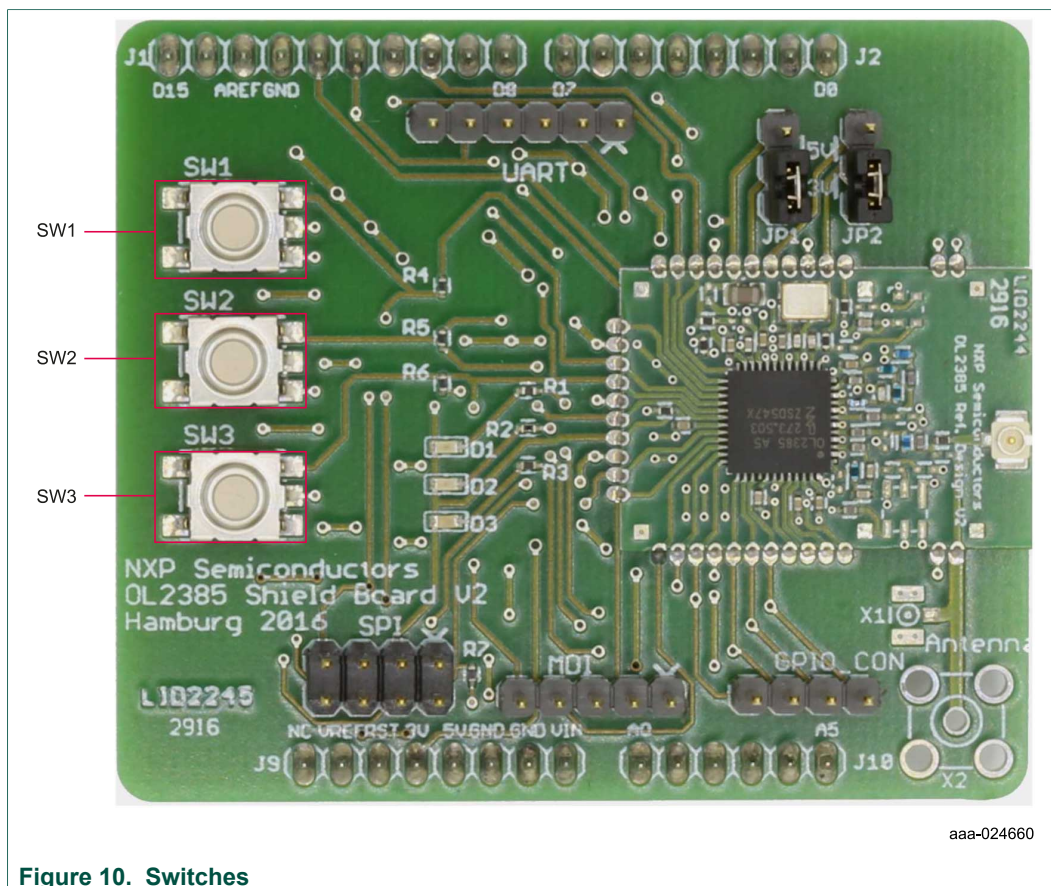
Figure 9 shows the location of jumpers and switches on the OM2385/SF001 evaluation board.





5.3.6 OL2385 ref design board switch definitions

Figure 10 shows the location of switches on the OM2385/SF001 Shield Board.



## 5.4 Connecting OL2385 with MCU

The Arduino based NXP reference design boards (green board) connect to a KL43Z board as shown in [Figure 11](#). Buttons on the OL2385 board are facing toward the USB ports on the KL43Z board. Connect your NXP reference board in a similar manner to the KL43Z. If you have a different type of OL2385 based board, refer to the later sections in this document, specifically the SPI pins of OL2385. Connect the boards accordingly.

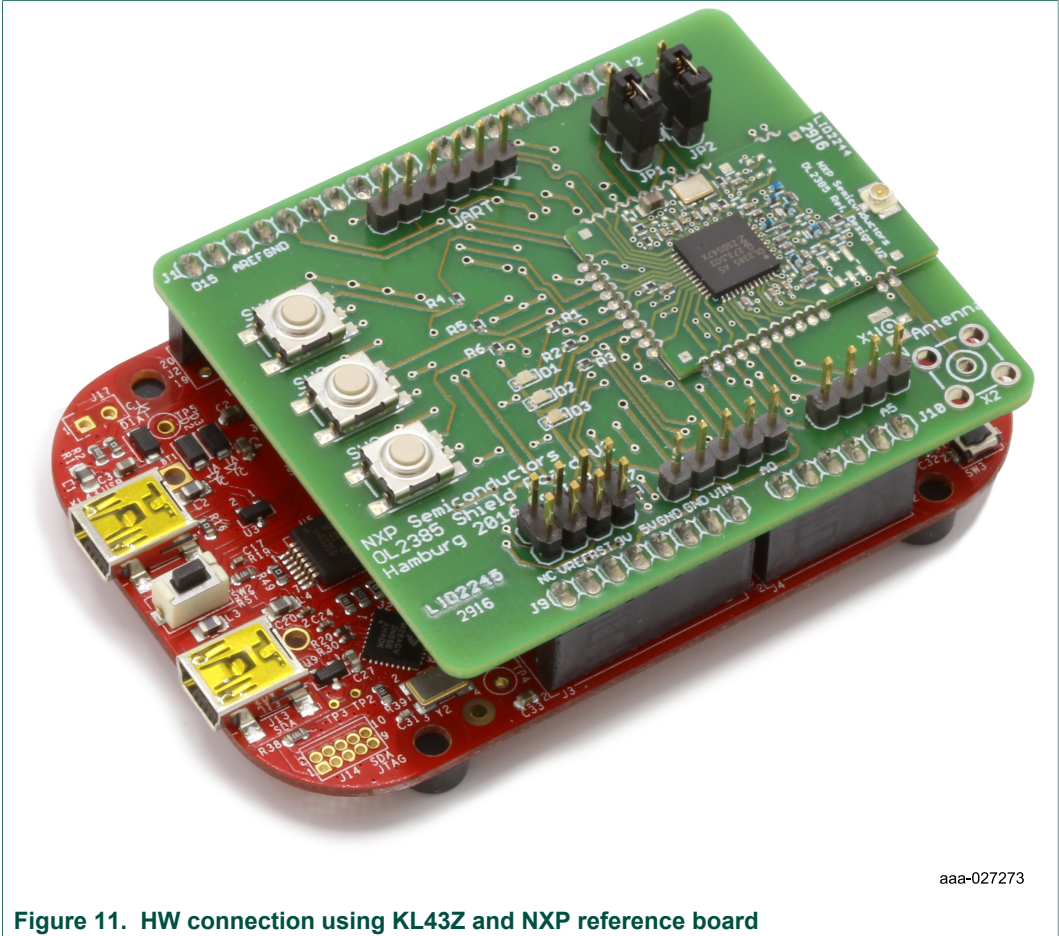
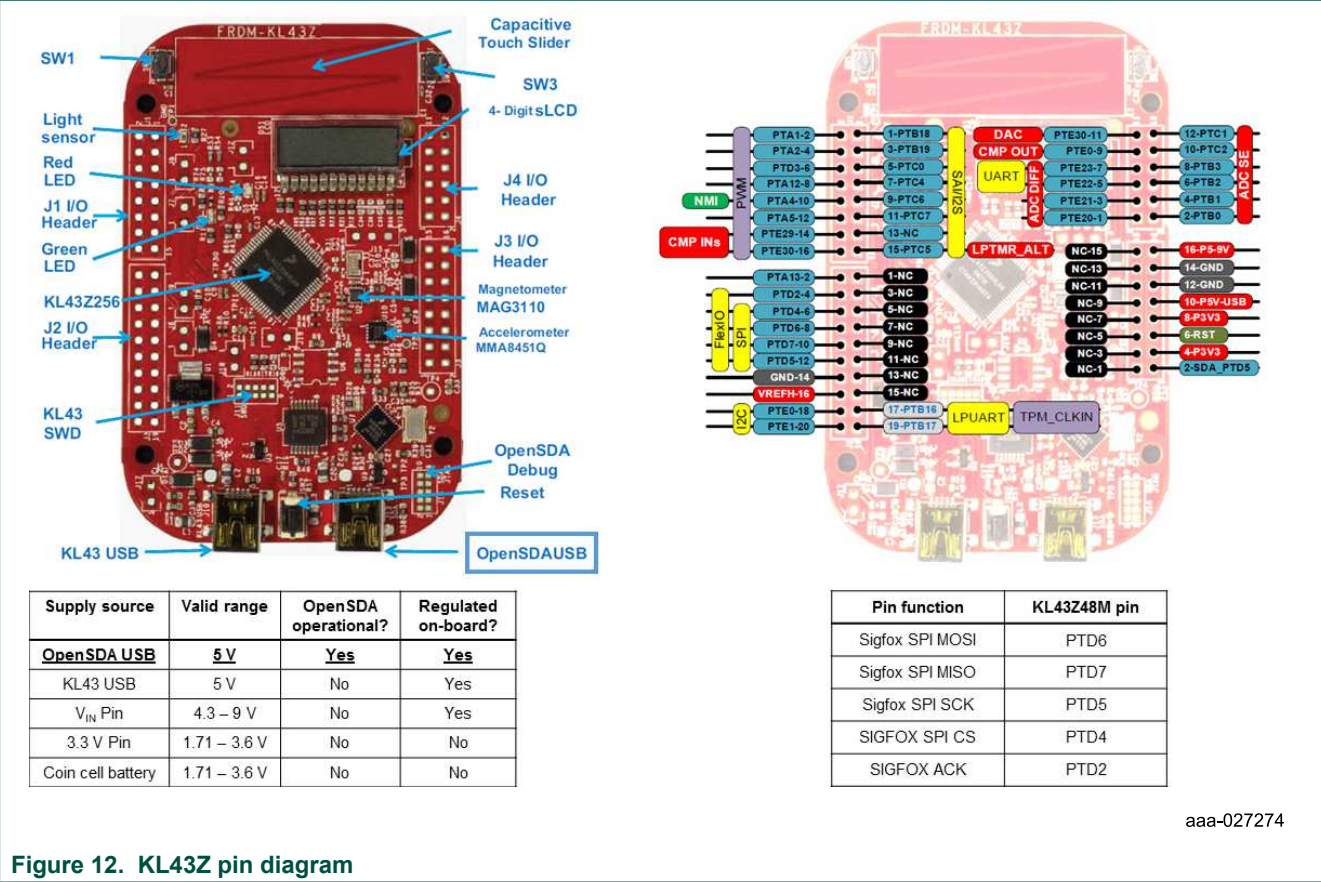


Figure 11. HW connection using KL43Z and NXP reference board

The KL43Z board running with NXP SIGFOX testing firmware `spi_sigfox_demo.srec` executable file has SPI pins mapped on J2 I/O header, which is marked in [Figure 12](#). The exact pin numbers are specified in [Table 2](#). The power supply and RESET pins are mapped on the J9 I/O Header.







#### 5.4.1.1 Downloading and installing the driver for the FRDM-KL43Z

This procedure involves downloading the FRDM-KL43Z driver from the P&E Microcomputer Systems website and installing it on the host PC.

1. Go to the P&E Microcomputer Systems OpenSDA page at [www.pemicro.com/opensda](http://www.pemicro.com/opensda) and, in the Windows USB Drivers box, click to download the PEDrivers\_install.exe file to a location on the host PC.
2. When the download completes, click on the PEDrivers\_install.exe file and follow the instructions to install the driver.
3. Connect a USB cable between the host PC and the FRDM-KL43Z USB port labeled SDA (J13).
4. Open Windows Explorer on the host PC. An icon labeled FRDM-KL43Z appears as a removable drive on the PC.

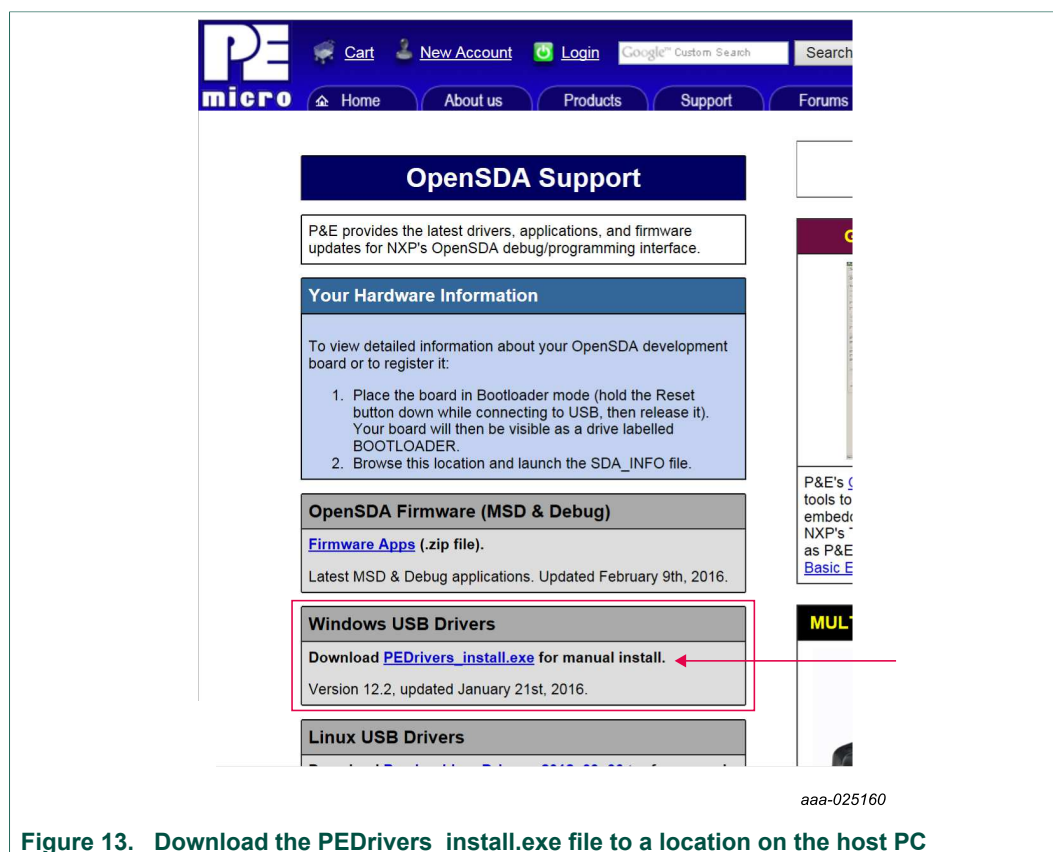


Figure 13. Download the PEDrivers\_install.exe file to a location on the host PC

The FRDM-KL43Z is now ready for use with the OM2385/SF001 development kit.

#### 5.4.1.2 Connecting the hardware for use with the SIGFOX network

To connect the hardware to send messages across the SIGFOX network, do the following:

1. Check to assure that the OM2385 board is firmly attached to the FRDM-KL43Z. When connecting the boards, the three switches on the shield board should be on the same side as the USB ports on the FRDM-KL43Z board.
2. Attach the PCB antenna (included with the kit) by snapping the uFL connector on the antenna to the uFL connector on the shield board.

3. Connect the Standard A end of the supplied USB cable to a Windows host PC. Connect the Mini B to the FRDM-KL43Z USB port labeled SDA (J13).

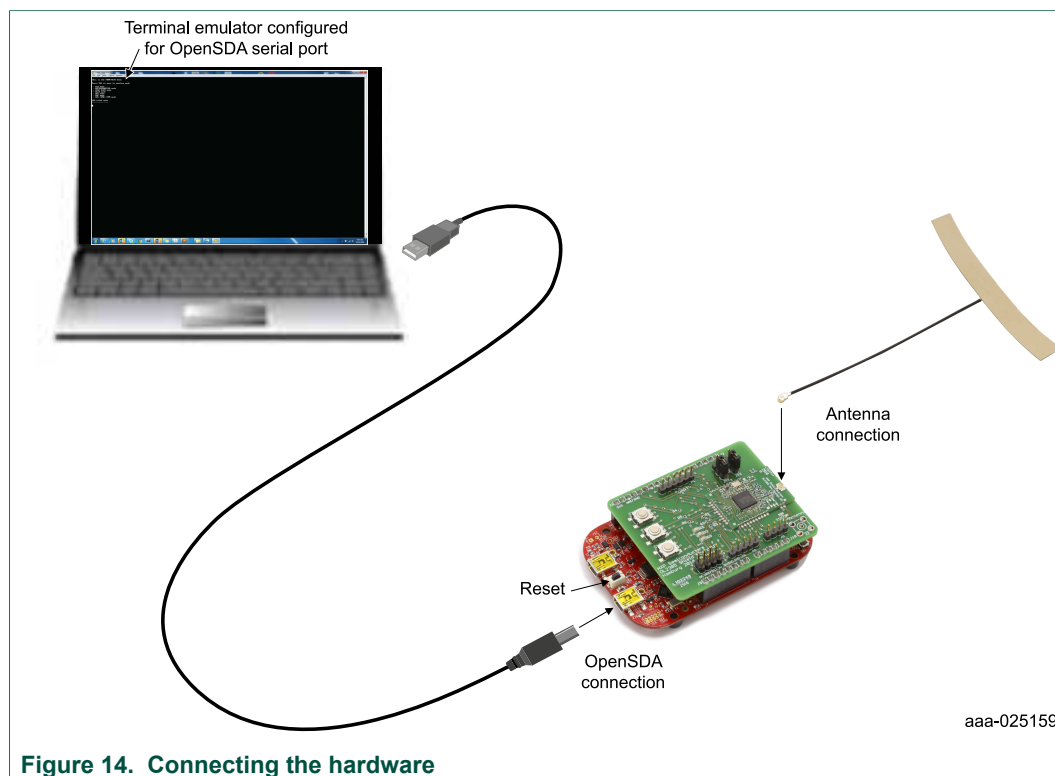


Figure 14. Connecting the hardware

The OM2385/SF001 is now ready to be configured for use with the SIGFOX network.

#### 5.4.1.3 Setting up terminal program

1. Plug one end of a mini USB cable to the SDA USB port on the board. Plug the other end of the cable to a PC. See [Figure 14](#).
2. For the first time, install drivers for this device on a PC. Install PEDrivers\_install.exe downloaded from the page [www.pemicro.com/opensda/](http://www.pemicro.com/opensda/).
3. If the device has an updated bootloader installed in it already, it will be shown detected as a removable device on windows PC labeled as FRDM-KL43Z as shown in [Figure 15](#). If the bootloader is not installed, follow steps given in Quick Start Guide for FRDM-KL43Z ([www.nxp.com/FRDM-KL43Z](http://www.nxp.com/FRDM-KL43Z)) to update bootloader.
4. If the KL43Z is not preflashed with a srec file, download the file from the web else move to next step. Now drag the file into this removable drive as shown above. Do not open the removable drive and copy-paste, but just drag the file on top of it. Press copy and replace, if you are updating the software.
5. Press the reset button located between the two USB ports to let the device accept and run the updated software inside the KL43Z.
6. Install and open a terminal software like Tera Term with the settings as shown in [Figure 15](#). Use the OpenSDA com port number, which you can get from the Windows Device Manager program. Press and hold (or right-click) the **Start** button, then select **Device Manager** from the context menu. In the Device Manager, click on **Ports (COM & LPT)**. Under OpenSDA – CDC Serial Port, note the COM port number. You can set these settings on the **Setup** menu of the terminal program.

**Note:** Understand that the COM connection is established as soon as you select the COM port and click on **OK**. If you remove the USB cable without closing the terminal, the COM connection breaks but is not fully closed. And if you plug the cable back in the USB, the COM connection does not establish again. Therefore, you will see that the terminal does not respond. **Therefore, always close the terminal before removing the USB cable and open the terminal only after you have plugged back in the USB cable.** Connect the RESET pins of both boards and use middle reset button on the KL43Z board to reset. Unplugging and replugging the USB is not the preferred way to reset the boards.

7. After you Press the reset button on the KL43Z, a menu as shown in [Figure 15](#) will appear, indicating that the KL43Z board is properly set up.
8. Type **1** and press Enter to send wakeup command over SPI. By default, after power-on reset, the device goes in deep sleep or POWEROFF2 power-saving mode. **Therefore, the wakeup command always has to be sent first after any kind of board reset.**
9. Type **8** and press Enter to send a Get info command. This command will display the ID and the PAC value of the device. Even if these values are zero, if you see SPI TX buffer and SPI RX buffer bytes, SPI bidirectional communication is working.
10. The next command depends on the region you would like to test, namely RCZ1, RCZ2, RCZ3 or RCZ4. Choose between 0x15, 0x16, 0x17 or 0x18 accordingly. Note that the OL2385 boards for each of these respective regions are different. For example: RCZ2 and RCZ4 uses external PA whereas, RCZ3 uses 27.6 MHz external TCXO. However, the OL2385 firmware supports all four regions. However, the system as a whole will only work if you choose the appropriate region on the terminal for the hardware.
11. The rest of the commands are self-explanatory and depend on your choice of testing tool. For example, a Spectrum analyzer might need a Continuous wave command, SIGFOX base station might need a Send payload command or SIGFOX SFXIQ P1 certification tool might need a Send test mode command. If required, refer to OL2385\_SPI\_Interface\_Commands.xlsx for details about SPI commands located at [www.nxp.com/OM2385](http://www.nxp.com/OM2385)

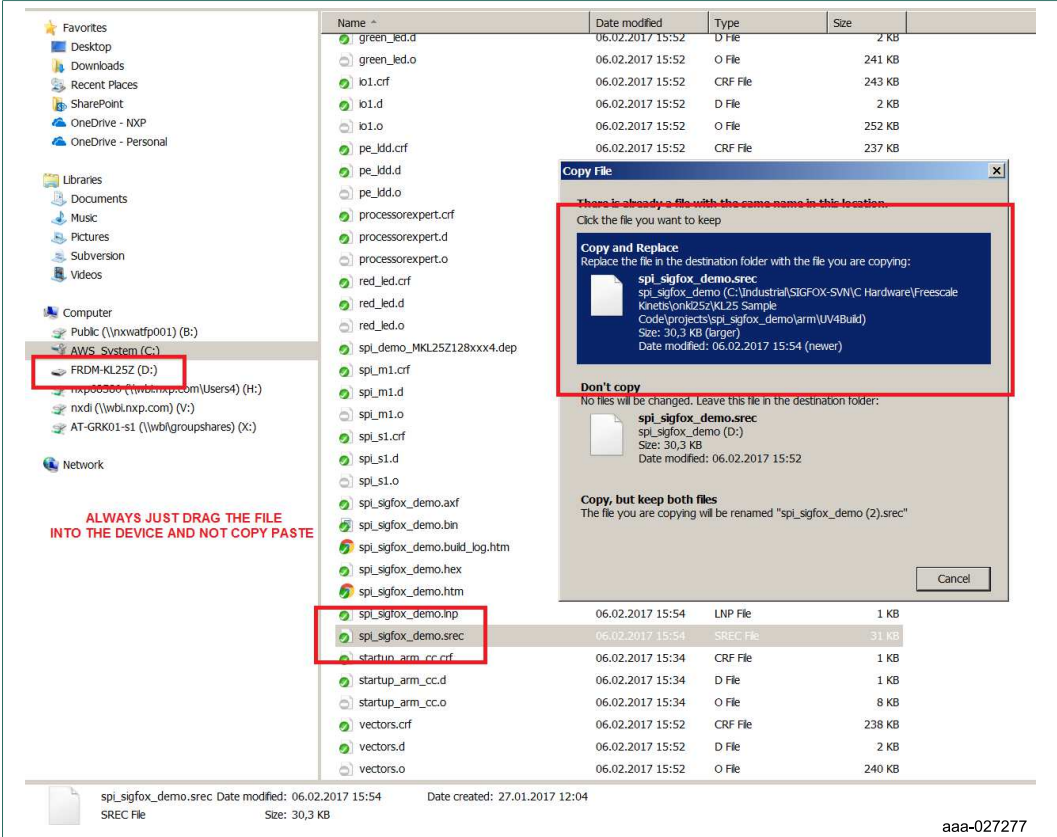


Figure 15. Flashing the firmware on KL43Z

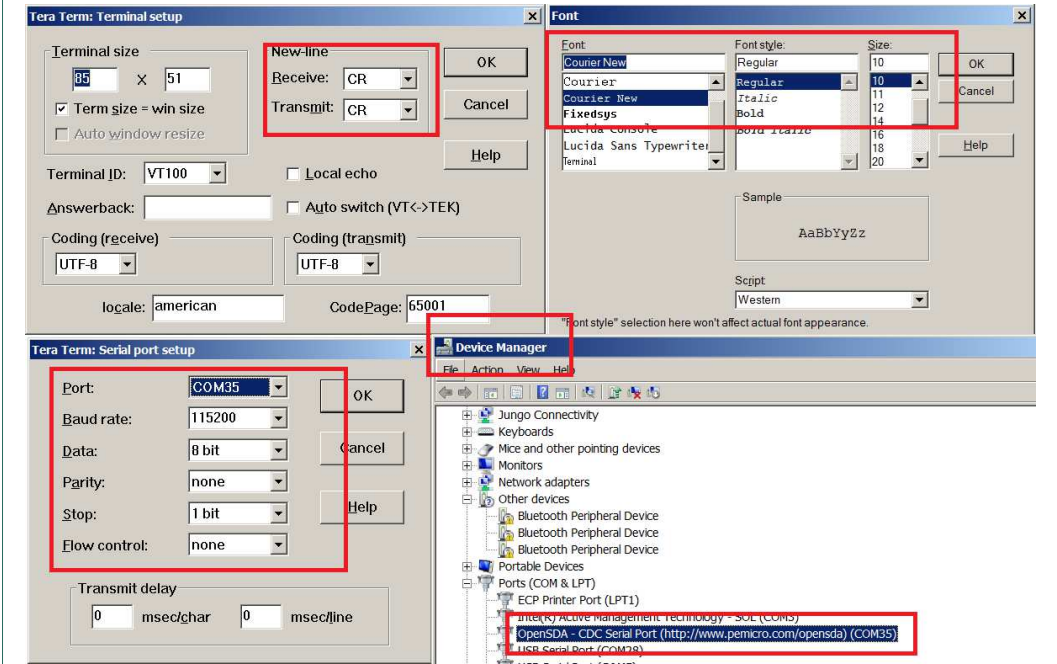


Figure 16. Tera-term example settings



```

COM35:115200baud - Tera Term VT
File Edit Setup Control Window Help

***** KL25Z SIGFOX TEST FIRMWARE VERSION 01.01.00.06.02.20.17 *****
*1. Send Wakeup
*2. Echo Command
*4. Send Payload
*6. Send Out of Band
*8. Get_info
*A. Get_UL_frequency
*D. Get Device Version
*14. Send Test Mode
*16. Change_To_RCZ2
*18. Change_To_RCZ4
*1A. Switch_to_Public_Key
*1C. PSK_ASK_AAAA_Sequence
*1E. Get_DL_frequency
*20. Get_FCC_Macro_Channel_Usage
*22. Static_Frequency_Calibration
*24. Get_Last_Sent_RSSI

*3. Send_To_Sleep
*5. Send_bit
*7. Receive_Frame
*9. Set_UL_frequency
*B. Continuous_Wave
*12. Trigger_Watchdog
*15. Change_To_RCZ1
*17. Change_To_RCZ3
*19. Switch_to_Private_Key
*1B. PSK_ASK_Zero_Sequence
*1D. Set_DL_frequency
*1F. Set_FCC_Macro_Channel
*21. Reset_FCC_Macro_Channel
*23. Temperature_Frequency_Calibration

*****
Choose the hex command code and press enter: 1
SPI TX Buffer
0x02 0x01 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

***** KL25Z SIGFOX TEST FIRMWARE VERSION 01.01.00.06.02.20.17 *****
*1. Send Wakeup
*2. Echo Command
*4. Send Payload
*6. Send Out of Band
*8. Get_info
*A. Get_UL_frequency
*D. Get_Device_Version
*14. Send Test Mode
*16. Change_To_RCZ2
*18. Change_To_RCZ4
*1A. Switch_to_Public_Key
*1C. PSK_ASK_AAAA_Sequence
*1E. Get_DL_frequency
*20. Get_FCC_Macro_Channel_Usage
*22. Static_Frequency_Calibration
*24. Get_Last_Sent_RSSI

*3. Send_To_Sleep
*5. Send_bit
*7. Receive_Frame
*9. Set_UL_frequency
*B. Continuous_Wave
*12. Trigger_Watchdog
*15. Change_To_RCZ1
*17. Change_To_RCZ3
*19. Switch_to_Private_Key
*1B. PSK_ASK_Zero_Sequence
*1D. Set_DL_frequency
*1F. Set_FCC_Macro_Channel
*21. Reset_FCC_Macro_Channel
*23. Temperature_Frequency_Calibration

*****
Choose the hex command code and press enter: 16
SPI TX Buffer
0x02 0x16 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

SPI RX buffer
0x03 0x00 0x02 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

```

aaa-027279

Figure 17. KL43Z MCU using terminal program

## 5.4.2 Flashing the OL2385 binary

### Flashing the OL2385 binary

If the OL2385-based board is not preflashed with firmware or the firmware needs to be updated, the following are required before flashing the board.

1. OL2385 based module board with an MDI interface. Check the schematics or search for an MDI label on the hardware.
2. Hex file to be flashed, if the board is not yet flashed.
3. MRK III 2-link box to flash hex file.

4. MRK III 2-link box driver installation package.
5. Batch file for flashing the hex file.
6. Batch file for powering up the board.

Before flashing the hex file, install the drivers for the 2-link box on your PC. The installation instructions are provided in the 2-Link Installation files folder provided within the Customer\_Support\_Package.zip file. Follow these steps for flashing:

1. Connect the 2-link box as shown in [Figure 18](#). Find the MDI interface on the board and connect the ground wire (brown) of the 2-link box to the marked with a white arrow.
2. Place the hex file, MtGV0-SIGFOX.hex, in the **Customer\_Support\_Package/OL2385** folder. If the hex file name is different, rename the file to MtGV0-SIGFOX.hex. This is the name of the file used inside the batch file script.
3. The MRKIIlink.exe and MRKIIlink.dll should be present in **Customer\_Support\_Package** folder provided by NXP.
4. Run the OL2385\_load\_hexfiles.bat batch file to write the hex file software on OL2385.
5. Run the OL2385\_powerup\_device batch file to power up the device by the 2-link box, if you do not have any other power supply connected like microcontroller board or external battery. Otherwise, you can now disconnect the 2link box.

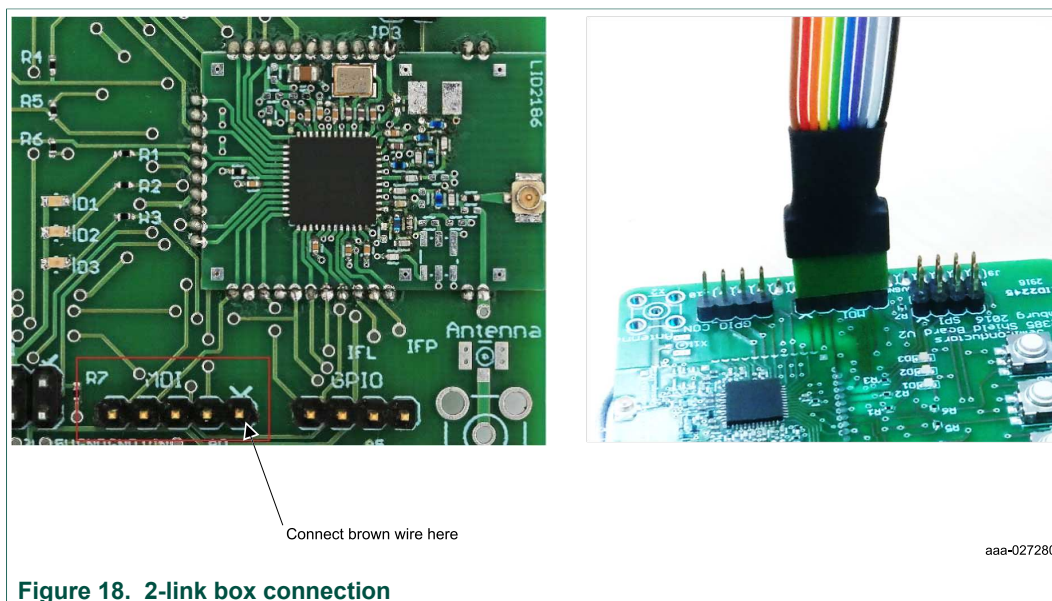


Figure 18. 2-link box connection

## 6 Setting up the software project

### 6.1 OL2385-SIGFOX firmware types

Depending on the interface to the OL2385 transceiver, the firmware running on it can be categorized into the following types:

- **SPI based firmware (fully available):** This version needs an external MCU connected to the OL2385 through a 5-wire SPI interface, as previously explained. This is a standard NXP product and readily available.
- **Button press based firmware (under investigation):** This version needs a small application to be written to the OL2385 to enable use of the three buttons provided on

the board. There is no external MCU required, thereby saving the customer additional cost. Because each application from customer to customer is different, this version will be provided as a skeleton with a combined library (NXP radio plus SIGFOX stack). The customer can modify the skeleton for customizing the behavior of the buttons and LEDs. A basic version of the software exists and can be enabled by a macro (BUTTON\_PRESS\_MODE) in project settings. The creation of a combined library still has to be done, therefore it is not ready to be provided. Additionally, the program memory does not have a lot of space and therefore, the application has to be very small and simple.

- **UART based firmware (under investigation):** This version needs a UART interface to the OL2385. This interface can also be to an external MCU or to a terminal program. From a business point of view, it makes sense to not have an external MCU required, thus saving the customer additional cost. A basic version of the software exists and can be enabled by a macro in project settings (SIGFOX\_UART\_MODE). However, the driver for UART is not robust and needs to be rewritten completely. The AT commands need to be updated and bugs have to be fixed.

**Note:** A SIGFOX based module might go through a P1 certification process at SIGFOX, depending on if there is a hardware change from an NXP reference design or modification of certified NXP software. NXP has only standard product – SPI based firmware certified at SIGFOX for all regions. Even if the radio parts of the software remain unchanged, any small change in software will still be considered a change. And therefore, if a customer decides to use Button press based or UART based firmware, they will have to get their devices recertified. In order to do the certification, SIGFOX will require activation of various test sequences on the OL2385 device. This will be very difficult for Button press based firmware, because OL2385 has only three buttons. Therefore, a SPI or UART based interface to OL2385, where we can control these test sequences, is needed.

Similarly, a UART based firmware without an MCU is good for evaluation using a terminal program. But for end product, customers will use either an MCU to connect to UART interface or use buttons again to write application on the device. In case of using an external MCU, it is highly recommended to use SPI based firmware, because it is certified and readily available. In case of no MCU usage and using buttons as a final interface to the user, UART will just be used as a certification interface. Again, in such a scenario it is easier to use a SPI based interface for certification purposes, because button applications will vary from customer to customer. It will again be our strategy to provide the code to customers with a combined library.

**Proposed solution:** For the applications where buttons are the interface to users and an interface is required only for evaluation and certification, it makes sense to provide the code to customers with a combined library and skeleton source files. A macro can be used by customers to enable/disable the SPI interface, which then will be used for one time certification. This means that customers do not have to use the MCU permanently on all products, but only externally to get certification. Therefore, there is no need for UART based firmware. Also, UART would have required an additional FTDI cable to interface OL2385 for certification. Now, it will just be an MCU based device controlling OL2385 externally for certification.

## 6.2 Kinetis MCU based application

This chapter is relevant for writing the code for an application on Kinetis MCUs. If your KL43Z is preflashed, you can skip to [Section 7.1 "Testing on SIGFOX Base Station"](#).

Otherwise, this chapter describes the MCU peripheral requirements and the resources needed to control a SIGFOX device.

### 6.2.1 Peripheral requirements

Peripherals and resource requirements critical to the MCU's ability to handle a given part are as follows:

- SPI module is required for communication (MISO, MOSI, SCK, CS)
- GPIO is required for an acknowledge pin (ACK)
- Supply and GND pins to power-on the OL2385

### 6.2.2 Supported devices

This section identifies the models supported by the driver and describes the capabilities of each model. The SIGFOX software driver supports the following devices:

#### OL2385 (Available)

- RF transceiver
- Carrier frequency is 160 MHz to 960 MHz
- ETSI, FCC and Japanese compliant
- 2 antenna inputs
- Independent RF switch (TX/RX or RX/RX)
- Up to +14 dBm output power
- 26 Channel Filter BW Options (4 kHz to 360 kHz)
- Smart polling
- Excellent phase noise
- Ultra-low power in receive mode

#### OL2361 (Software to be updated on demand)

- RF transmitter
- Carrier frequency is 310 to 915 MHz
- ETSI, FCC and Japanese compliant
- Multichannel fractional-N PLL
- One reference frequency (XTAL) for all bands
- Programmable FSK/ASK/OOK modulation characteristics
- Improved programmable and stabilized output power
- Low power consumption

### 6.2.3 Supported MCUs

The SIGFOX software driver supports MCUs listed in [Table 3](#). These MCUs are a subset of the MCUs supported by the MCUXpresso Software Development Kit layer.

This software driver is built on the Analog Middleware Layer (AML), which creates an API abstraction layer for the desired Software Development Kit (SDK). The current implementation includes abstractions for MCUXpresso SDK 2.3 and S32 SDK 0.8.4 EAR. This allows support to be added for additional layers, such as the MCUXpresso SDK, without having to change the SIGFOX software driver itself.



Table 3. Supported MCUs in SDK

Supported MCUs	SDK
MKL43Z256 (FRDM-KL43Z)	MCUXpresso SDK 2.3
MKL27Z64 (FRDM-KL27Z)	MCUXpresso SDK 2.3
MKL25Z128 (FRDM-KL25Z)	MCUXpresso SDK 2.3
MK64FN1M0 (FRDM-K64F)	MCUXpresso SDK 2.3

See [Table 4](#) for pin compatibility between the SIGFOX freedom board and selected MCUs.

Table 4. Compatibility of the SIGFOX board with selected MCUs

Pin function	FRDM-KL43Z	FRDM-KL27Z	FRDM-KL25Z	FRDM-K64F
ACK	PTD2	PTA5	PTD5	PTC4
CS	PTD4	PTC4	PTD0	PTD0
MOSI(SDI)	PTD6	PTC6	PTD2	PTD2
MISO(SDO)	PTD7	PTC7	PTD3	PTD3
SCK	PTD5	PTC5	PTD1	PTD1

## 6.2.4 The SIGFOX software driver

This section provides an overview of the functionality, configuration and usage of the driver. For additional information, see the API programmer's guide (included in the SIGFOX software driver zip file) and the comments in the code.

### 6.2.4.1 Configuring the driver

The configuration structure shown below serves as the user interface for configuring the driver. Users must add the appropriate values into the structure to reflect the desired driver setup. The configuration structure is passed directly to the initialization function at startup.

```

/! @brief This structure is used by the user to initialize the SIGFOX device.
* It contains a configuration of the SIGFOX device only (no SPI,
* etc. configuration). */
typedef struct
{
    sf_net_standard_t netStandard; /*!< Selection of a standard defining network
                                     communication parameters. */
    uint8_t txAttSteps; /*!< Amount of 0.25 dB steps that needs to be
                                     subtracted from TX power. */
    sf_xtal_type_t xtal; /*!< Selection of XTAL type used in the
                                     device. */
    sf_tx_repeat_t txRepeat; /*!< Amount of transmissions that will happen
                                     on one time invoking of SF_SendPayload
                                     function. */
    sf_pa_type_t paType; /*!< Selection of internal PA type used in
                                     the device. */
} sf_user_config_t;

```

The user's configuration structure includes the following variables:

- **netStandard** selects a standard for defining network communication parameters
- **txAttSteps** contains number of 0.25 dB steps that needs to be subtracted from the TX power
- **xtal** selects between usage of 55.2 MHz and 27.6 MHz TCXO XTAL
- **paType** selects between internal PA being used: 14 dBm or 0 dBm
- **txRepeat** selects number of transmissions that will happen when invoking SF\_SendPayload function (1TX or 3TX).

For a more detailed description of the user configuration structure, refer to the API programmer's guide included in the SIGFOX software driver zip file.

### 6.2.4.2 Driver API

This software driver provides an API that allows user application code to configure the device in real time. For a summary of the available functions, see [Table 4](#).

**Table 5. SIGFOX software driver API**

Function	Description
SF_Init	Initializes the SIGFOX driver based on user configuration
SF_GetDefaultConfig	Loads the user configuration structure with default values
SF_SendCommand	Sends a command to the device. This function waits for an acknowledgment (if any)
SF_WakeUp	Sends the <b>Send_Wakeup</b> command to wake up the device from power down mode
SF_Sleep	Puts the device in power off mode by means of the <b>Send_To_Sleep</b> command
SF_SetUIFrequency	Sets the uplink frequency of the SIGFOX device
SF_GetUIFrequency	Gets the uplink frequency of the SIGFOX device
SF_SendPayload	Sends the <b>Send_Payload</b> command to the device, which sends user data to SIGFOX network
SF_SendBit	Sends the <b>Send_Bit</b> command, which transmits a single bit to the SIGFOX network. This is the shortest frame that the SIGFOX library can generate. This command is useful for network testing.
SF_SendOutOfBand	Sends the <b>Send_out_of_band</b> frame, which transmits data to the SIGFOX network. Data is composed of information about the chip itself (voltage, temperature). This function must either be called at least once every 24 hours or never, depending on whether an application has some energy critical constraints.
SF_ReceiveMessage	Receives a frame from SIGFOX network using the <b>Receive_Frame</b> command. Available for the OL2385 device only
SF_TriggerWatchdog	Resets the SIGFOX device watchdog using the <b>Trigger_Watchdog</b> command.
SF_GetDeviceInfo	Reads Device ID, PAC, SIGFOX library version and device version using the <b>Get_info</b> and <b>Get_Device_Version</b> commands
SF_GetState	Returns the current state of the SIGFOX device in the software state machine. The state is updated every time an ACK SPI frame is received.
SF_GetErrorCode	Returns an error code received in a SPI frame from the SIGFOX device. The error code is updated every time an ACK SPI frame is received.
SF_TestSpiCon	Tests if the SPI bus is working by using the <b>Echo</b> command to send data (see SF_ECHO_DATA macro) and then checking if the device replies with the inverted payload
SF_TestDevice	Executes a test of uplink and downlink connectivity using the <b>Send_Test_Mode</b> command. The returned status indicates the success or failure of the test. The user can obtain details using the <b>SF_Get_Error_Code</b> function.
SF_GenerateContWave	Sends the <b>Continuous_Wave</b> command to generate a continuous transmission at the last set frequency. The signal can be then analyzed using a spectrum analyzer.