UM0433
User manual

SNDEV-260 ZigBee® kit

Introduction

This document describes the SNDEV-260 ZigBee® kit which is a full-featured tool kit which offers all the hardware and software needed for deployment of wireless networks for remote monitoring, sensing, and control-network applications based on the ZigBee® standard.

The kit provides a complete hardware/software development environment and network system (3 nodes) that uses the SN260 ZigBee® network processor and the EmberZnet™ ZigBee® advanced wireless protocol stack.

The kit facilitates application development with sample applications and a complete software tool set that includes Ember's InSight™ Desktop with Network Traffic Analyzer to display network and node activity in real time.

This user guide is intended to provide an overall description of the software and hardware requirements for the SNDEV-260 ZigBee® kit, instructions for setting up the hardware and building the wireless application examples as well as describing known limitations and issues at release time.

Due to the total compatibility between the SN260 and EM260, and the partnership between the two companies, some documents have been kindly supplied by Ember.

List of key words

EmberZNet:	ZigBee® stack running over the SN260 silicon
EZSP:	Ember ZigBee® Serial Protocol
HAL:	Hardware Abstraction Layer
ISA:	InSight™ Adapter
ISD:	InSight™ Desktop
POE:	Power Over Ethernet
RCM:	Radio Communication Module
RIDE:	Raisonance Integrated Development Environment

Contents

1 Release notes

The SNDEV-260 kit library package supports STMicroelectronics' STM32F103x and ST7LITE39 microcontrollers whose hardware daughter boards are supplied with the kit.

It also supports STMicroelectronics' STR71xF, STR75xF and STR91xF microcontrollers which are not part of the kit.

2 SNDEV-260 kit hardware and software content

This section describes the SNDEV-260 kit contents.

2.1 Hardware components

The SNDEV-260 ZigBee® kit includes 3 hardware nodes.

Each individual hardware node in SNDEV-260 kit consists of:

- 1 radio communication module SN260 RCM
- 1 REva board for ZigBee + Rlink adapter attached + USB cable
- 1 ST7LITE39 ST daughter board
- 1 STM32F103RBT6 ST daughter board
- 1 InSight Adapter (plus a USB connector) to monitor any type of exchanges
- 1 InSight Port cable
- 1 small bag of jumpers
- 1 (straight) RJ45 Ethernet cable
- 1 USB-mini cable

Note: An Ethernet switch with POE is also provided.

2.1.1 SN260 radio communication module (RCM)

The SN260 radio communication module (RCM) offers a complete ZigBee wireless solution for development and deployment of a low-data-rate, low-power ZigBee application. The SN260 RCM works combined with a ST host microcontroller (STM32F103RBT6 or ST7LITE39), using the kit application board.

The radio communication module includes these components:

- The SN260, a 2.4 GHz, IEEE 802.15.4-2003 Network Processor running the Ember ZigBee-compliant EmberZNet stack
- Two 6-pin RCM interface connectors to access the EmberZNet Serial Protocol (EZSP)
- An MC-Card RF switch
- A ceramic SMT antenna
- A low-profile crystal (24 MHz)
- A ceramic balun
- 10-pin InSight Port connector which allows connectivity to InSight Desktop for debug

The RCM directly attaches to the kit application board which contains the ST microcontroller daughter board. The ZigBee application runs from the ST microcontroller and communicates with the SN260 through the EZSP APIs. Each radio communication module is assigned a unique IEEE 64-bit identifier (EUI-64). This number is printed on a label affixed to the bottom of the module and can also be viewed in InSight Desktop. For detailed information about the radio communication module, see the technical specification in your kit.

Note: The SN260 Network Processor is provided with the Ember ZigBee-compliant EmberZNet stack image already downloaded on it. To update the stack image stack, follow the instructions in *Section 4: Updating the EmberZNet stack image on page 41**.*

2.1.2 REva board for ZigBee

The REva ZigBee Raisonance board is provided with the kit. This board hosts the SN260 RCM modules and the ST microcontroller daughter board allowing running ZigBee applications.

The REva ZigBee Raisonance board is made up of a generic mother board with embedded RLink in-circuit programmer and debugger, a daughter board featuring a target microcontroller and a ZigBee network area. The REva ZigBee Raisonance board has the following components:

- ZigBee area for hosting a SN260 RCM module
- Digital and analog I/O evaluation features including on-board LEDs, buttons, switches, buzzer, etc.
- On-board RS232 driver and DB9 connector
- SPI, CAN and USB connections (depending on the target device)
- Embedded RLink for in-circuit debugging and in-circuit programming
- VDD settings for 1.8V, 3.3V and 5V microcontrollers
- USB powered, no external power required

In addition, an USB cable for programming the ST microcontroller using the RIDE toolset (and also to power the overall board) is provided.

Note: It is also possible to power the REva board using an external power supply (9V) and setting the REva power jumper to the specific position (in particular, this configuration is used with the IAR toolset).

2.1.3 ST7LITE39 daughter board

The ST7LITE3 daughter board includes the following components:

- 1 ST7LITE39 microcontroller
- 1 six-position jumper for selecting the clock source
- Certain passive devices for decoupling and managing reset priorities

2.1.4 STM32F103RBT6 daughter board

The STM32F103RBT6 daughter board includes the following components:

- 1 STM32F103 microcontroller
- various jumpers for the chip configuration (boot mode, etc.)
- a 8-MHz crystal clock and USB clock generation
- a 32-kHz crystal for RTC operation
- 1 USB mini-B connector
- 1 CAN bus driver

2.1.5 InSight adapter

The SNDEV-260 kit includes InSight Adapter which connects the application board to the Ethernet. Each adapter transmits network data collected by InSight Port and conveys it over its Ethernet connection to InSight Desktop tool. It also picks up any messages or new software that is addressed to this board, and processes emulation and debug commands. ISA allows the host PC interfacing to the hardware node during debugging and programming.

An Ethernet switch with POE for connection of multiple adapters to a host PC is also provided.

The InSight Adapter has the following components:

- 10-pin InSight Port for interfacing to the radio communication module (providing programming and debugging services)
- TCP/IP 10/100 Ethernet interface with Power-over-Ethernet functionality.

The SNDEV-260 kit contains a 16-port Power-over-Ethernet switch that supplies power to an InSight adapter over a standard Ethernet cable. The InSight adapter can, in turn, supply power to an application board and the radio communication module that is mounted on it. Thus, you can place the application boards wherever an Ethernet cable connection is available.

The single ISA can be also powered through a DC power jack, for applications not using Power-over-Ethernet.

Note: Power-over-Ethernet is disabled when the InSight Adapter is powered with the12V DC power input. However, the Ethernet data is still available.

For detailed information about the InSight Adapter, see the relative document in your kit.

2.1.6 External equipment

To setup a ZigBee network, using the SNDEV-260 hardware and software components, the minimum external equipment required is a PC (Windows 2000 or XP) with the Raisonance RIDE and Ride7 toolsets. For more information, refer to *Section 2.2.3: Ride7 toolset on page 11*.

When using the IAR toolset, the following equipment is required for programming (and/or debugging):

- IAR J-Link JTAG emulator hardware for programming/debugging
- External power supply (9 V) for powering the REva board

Cabling:

- Serial cable (for connecting the REva board serial SER1 to a PC RS232 port)
- USB mini cable (for connecting the STM32F103x or STR71x-9x USB mini-B connector to a PC USB port)

2.2 Software components

The SNDEV-260 ZigBee kit includes the following software components:

- EmberZnet™ ZigBee compliant networking stack
- EZSP library for driving the SN260 (by STMicroelectronics) or EM260 (by Ember) ZigBee devices
- HAL library for addressing some of the hardware platform devices and capabilities
- Certain wireless application examples:
  - Sensor and sink
  - Light switch
  - Range test
- RIDE toolset (version 06.10.22) used for the ST7LITE39 microcontroller libraries and applications building and running
- Ride7 toolset (Ride7 IDE version 7.01.0002 and RKit ARM for Ride7 version 1.03.0003) used for the STM32F103x and STR7x-91x ARM microcontroller family libraries and applications building and running
- InSight Desktop DEVeloper License (3 nodes, 10k events)

2.2.1 EmberZnet™ ZigBee compliant networking stack

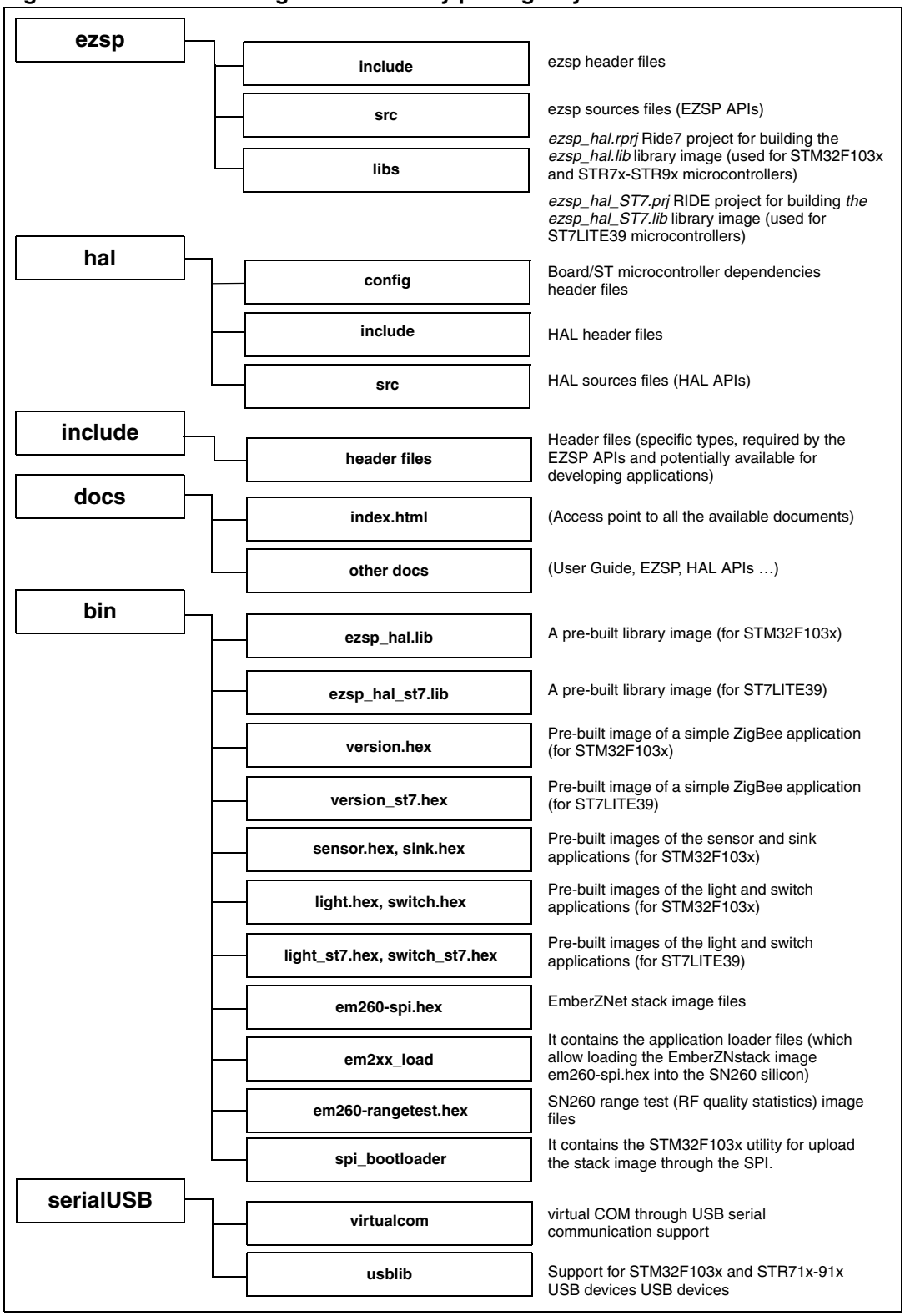
The EmberZnet™ ZigBee compliant networking stack is an advanced wireless protocol stack that runs on the SN260 network processor, and provides algorithms for creating reliable, flexible, and secure networks.

For information about the EmberZnet™ ZigBee stack, see the relative EmberZNet application developer's guide in your kit.

2.2.2 EZSP and HAL libraries

Figure 1 describes the SNDEV-260 ZigBee® kit library package layout.

Figure 1. SNDEV-260 ZigBee® kit library package layout

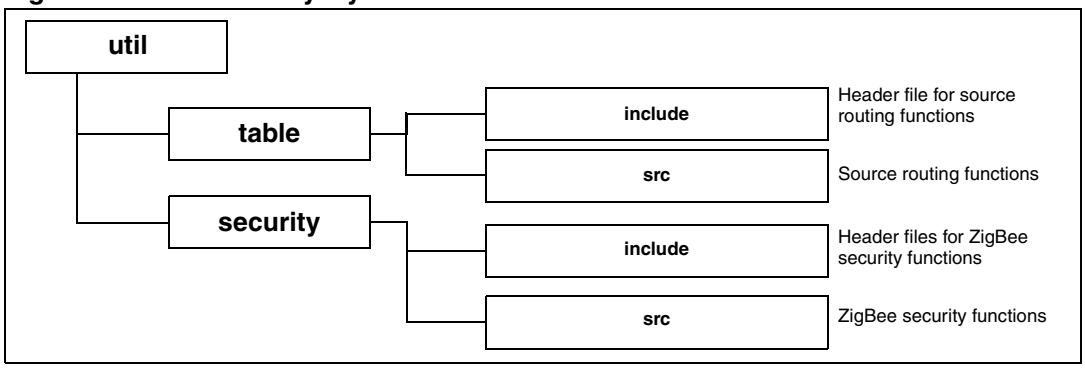


Note: The EmberZNet 2.5.x stack version is delivered in the XPV and XDV format (same for the related range test application).

If using the Ember ZNet 3.x stack, a new *util* directory is provided. It includes functions for handling the ZigBee source routing and security.

Figure 2 describes the *util* directory layout.

Figure 2. *util* directory layout

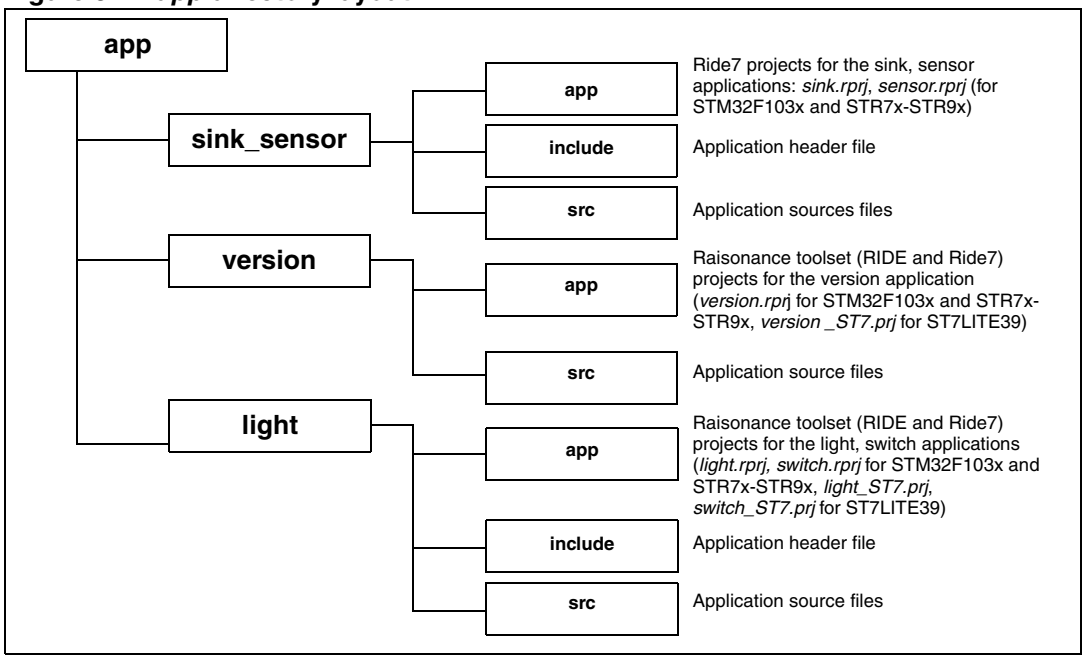


Certain application examples are provided to show ZigBee network functions. The kit includes two simple applications (light and switch) to control a light source. There are also two more complex applications (sink and sensor) to configure a distributed sensors network and a prebuilt range test application to perform a simple send/receive test on the device to determine its range and generally test its radio function.

An *app* directory is provided within the SNDEV-260 kit for hosting applications built over the EZSP and HAL libraries.

Figure 2 describes the *app* directory layout.

Figure 3. *app* directory layout



Furthermore, for each application, the corresponding IAR workspace is provided to enable the user to directly build the application using the IAR toolset.

Note: 1 The ezsp_hal library does not provide an IAR workspace.

2 This version of the SNDEV-260 ZigBee package provides the IAR workspaces only for STM32F103x, STR71x and STR91x microcontrollers.

3 The version application is a simple single node application which enables basic ZigBee operations to be performed: stack initialization, stack version and EUI64 Node ID. It can be used just for checking that the application board, ST microcontroller and SN260 RCM modules are correctly connected and communicating. It does not enable the use of any ZigBee networking features.

4 STM32F103x microcontrollers are only supported by the new Raisonance Ride7 environment, which still provides support for STR7x-91x microcontrollers. The RkitST7 for the Ride7 has some limitations/issues (it does not support the Cosmic toolchain and its library manager does not build a library). A RIDE project is still provided for the ezsp_hal library version for the ST7LITE39 microcontroller (and related applications). Consequently, two different Ride7 and RIDE projects are provided to target the library/applications based on the microcontroller family being used. The _ST7 suffix identifies the microcontroller family and the rprj (Ride7) or prj (RIDE) suffix identifies the Raisonance environments to be used. (For example: use the version.rprj Ride7 project for STM32F103x and STR7x-91x microcontrollers and the version_ST7.prj RIDE project for ST7LITE39 microcontrollers).

The sensor and sink applications are not supported for the ST7LITE39 microcontroller due to its smaller Flash memory (only 8 Kbytes) and RAM (only 384 bytes).

The available pre-built images of version.hex, light.hex, switch.hex, sensor.hex, sink.hex and ezsp_hal.lib are provided only for STM32F103x microcontrollers (not the STR7x or STR91x).

The available built images version_st7.hex, light_st7.hex, switch_st7.hex and ezsp_hal_ST7.lib are created for ST7LITE39 microcontrollers.

The libraries and applications are distributed through a specific link in the STMicroelectronics web site.

2.2.3 Ride7 toolset

The Raisonance Ride7 toolset (Ride7 IDE version 7.01.0002 and RKit ARM for Ride7 version 1.03.0003) is used for the library, applications building and running with the STM32F103x, STR7x-91x ARM microcontroller family. Ride7 is the new integrated development environment, designed for the development of ST microcontroller projects. This tool automatically manages file dependencies so that a makefile is not necessary.

The toolset is available by selecting “Download the latest CD-ROM image” at http://www.stm32circle.com/resources/tools.php

Note: The Ride7 toolset has the following known limitations:

The Ride7 RKit for the ST7 microcontrollers does not support the Cosmic toolchain. Raisonance does not plan to add Cosmic toolchain support to the Ride7 environment. The solution is to use the RIDE toolset (version 06.10.22) or the Raisonance toolchain provided inside the Ride7, RKitST7 for the ST7LITE39 microcontrollers.

The current library manager of the Ride7, RKitST7 Raisonance toolchain does not work. It does not build a library: Raisonance support cannot provide a proper fix in a short time. For the ST7LITE39 microcontroller, the RIDE (version 06.10.22) toolset with the Cosmic toolchain is used.

2.2.4 RIDE toolset

The Raisonance RIDE toolset (version 06.10.22) is provided for the library, applications building and running with the ST7LITE39 microcontroller. RIDE is an integrated development environment, designed for the development of ST microcontroller projects. This tool automatically manages file dependencies so that no makefile is necessary.

The toolset is also available from the Raisonance web site. Download (for free, after a simple account creation request) the RKit-ST7+STRx+uPSD package (following the reported instructions if some other patches are required) and install it. This installation comes with some documentation about the RIDE tool and Raisonance development boards (including the ST microcontroller's daughter boards).

*Note: The current RIDE toolset has the following known limitation (when used for the ST7LITE39 microcontroller): The RIDE default library manager incorrectly builds the *.lib file format. It does not use the clib utility even if the RIDE has been configured for using the Cosmic toolchain (see below).*

Raisonance support recommends using a script (make_library.wsc) for running the clib utility and build the library image. The ezsp_hal_st7.prj project is already configured for using this make_library.wsc script, so the user is required to double-click on this script after executing the "Project Build all" command.

Cosmic C toolchain for ST7

RIDE-ST7 can be used together with a number of third party tools such as the Cosmic C toolchain.

The Cosmic C toolchain for ST7 is composed of a C compiler, an assembler and linker and can be used without leaving RIDE's graphical user interface (GUI).

Configuring RIDE and the COSMIC C toolchain to work together

In order to use RIDE together with the Cosmic C toolchain, the Cosmic software package must be installed. The Cosmic tools are not included in the distribution of RIDE-ST7. The free ST7 compiler 16K evaluation version is available for download at the Cosmic Software website after completing a request form. To use the Cosmic C toolchain it is also required to register with Cosmic Software: the installation procedure will send a message to Cosmic Software.

Once the two packages have been properly installed and registered, it is necessary to inform RIDE where the Cosmic tools are located. By default, RIDE looks for the Cosmic tools under c:\cosmic\cxst7, which is the default directory proposed by the Cosmic installation program. If this directory is changed during the Cosmic tools installation, this must be indicated to RIDE. To do this:

1. From **Options**, **Target**, **ST7 family** menu, open **Properties**, **Tools** and select **Cosmic Tools.**
2. From **Options**, **Target**, **ST7 family** menu, open **Properties**, **Settings** and select the Cosmic installation path.

2.2.5 IAR toolset

The IAR Embedded Workbench IDE for ARM toolset (version 4.42A which supports the ARM Cortex processor) is also used for building and running applications. The IAR Embedded Workbench IDE for ARM is a very powerful integrated development environment used for developing and managing complete embedded applications projects.

Note: 1 For programming (and/or debugging) using the IAR toolset, an IAR J-Link JTAG emulator is required. The IAR J-Link is a JTAG emulator designed for ARM cores. It connects via a USB port to a PC running Windows 2000 or XP. It has a built-in 20-pin JTAG connector, which is compatible with the standard 20-pin connector defined by ARM. Using the IAR J-Link, an external power supply is required for powering the REva board. The 9V power supply should output 9V DC and have a 2.1 x 5 mm jack with the ground signal on the outside.

*2 The first time the IAR J-link is plugged in the PC USB port, the user is requested to provide the relative J-Link driver, browse to the IAR installation directory and select the folder **ARM**, **drivers**, and **Jlink**.*

3 When disconnecting the IAR J-Link from the PC USB port, also unplug the IAR 20-pin JTAG connector from the REva on-board 20-pin JTAG ISD connector.

4 The IAR toolset and the IAR J-Link are not provided with the kit. For detailed information about the IAR products, refer to the www.iar.com web site.

2.2.6 InSight™ Desktop

InSight™ Desktop is a graphical tool that displays network and node activity in real time. It provides a rich and flexible interface to embedded networks, which helps you develop and debug new network applications. InSight Desktop includes these features:

- Multiple editor panes that provide tiered views of network activity, letting you drill down from a high-level map of node interactions to the details of each packet.
- Customizable filters that let you specify exactly which network activities to display.
- Log files that save captured data, so you can step through transactions and events for detailed analysis.
- A file browser that lets you easily upload new applications to any connected node.
- A browser-based interface for automatic discovery of Ethernet-connected adapters, and easy management of adapter applications.

The InSight Desktop DEVeloper license, included in the SNDEV-260 kit, supports up to 3 nodes and 10,000 radio events and includes a 6-month support and maintenance agreement.

For detailed information about InSight Desktop, refer to the *InSight Desktop User's Guide* in your developer kit.

Note: When using InSight Desktop with Windows® 2000, problems may occur when uploading the firmware into the SN260 processor. In this case, the em2xx_load.exe utility can be directly used by opening a Windows command prompt.

2.3 Hardware setup

2.3.1 Module/cable connections for applications building and uploading

Please ensure that the target hardware is connected as described below:

1. Plug the EM260 RCM module on the REva Raisonance two 6-pin, single row, and 0.1-inch pitch sockets present in the wrapping zone (they allow direct connection of the RCM module to the REva ZigBee platform).
2. Plug the ST microcontroller daughter board into the specific REva socket.
3. If using the RIDE or Ride7 toolset, set the REva power jumper according to the settings described in *Section 2.3.2* and plug the REva USB cable into the PC USB port and the RLink port.
4. If using the IAR toolset, set the REva power jumper according to the settings described in *Section 2.3.2* and plug an external power supply connector into the REva 9V DC input jack.
5. Plug the IAR J-Link 20-pin JTAG connector to the REva on-board 20-pins ISD connector and the J-Link USB cable to a PC USB port.
6. If the application requires displaying debugging messages and/or interaction with the user, the following serial communication channels are supported:
   - Serial COM through RS232 (for all microcontrollers)
     Connect a serial cable to the PC serial port and to the REva SER1 port.
   - Virtual COM through USB (only for STM32F103x and STR71x-91x microcontrollers)
     Connect the USB-Mini cable to the STM32F103x or STR71x-9x USB mini-B connector.

Caution: When unplugging the USB cable from the mini-B connector of the STM32F103x or STR71x-9x daughter board, always hold the daughter board with one hand while removing the cable with the other. Otherwise, the daughter board may be torn off from the SO-DIMM connector.

2.3.2 REva board jumper settings

Table 1 lists the jumper settings for REva board functional areas (inputs, outputs, analog, and com) when using STM32F103x, STR71x, STR75x and STR91x daughter boards.

Table 1. Settings when using STM32F103x, STR71x, -75x and -91x daughter boards

Jumper	Setting	Purpose
D7	Fitted	Enable LED D7 (available for application use)
D6	Fitted	Enable LED D6 (available for application use)
D5	Fitted	Enable LED D5 (available for application use).
D4	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D3	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D2	Fitted	Enable LED D2 (available for application use).

Table 1. Settings when using STM32F103x, STR71x, -75x and -91x daughter boards

Jumper	Setting	Purpose
D1	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D0	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
BT5	Fitted	Enable the BT5 button (available for application use).
BT6	Fitted	Enable the BT6 button (available for application use). Note that the POL jumper also has to be fitted to _| |_ position.
POL	Fitted as |_| for STM32F103x and STR91x. Fitted as _| |_ for STR71x and STR75x	Enable the BT6 button.
BUZZ	Fitted	Enable buzzer (available for application use).
TX	Fitted	Enable serial transmission (available for application use).
RX	Fitted	Enable serial reception (available for application use).

Table 2 lists the jumper settings for REva board functional areas (inputs, outputs, analog, and com) when using the ST7LITE39 daughter board.

Table 2. Settings for ST7LITE39 daughter boards

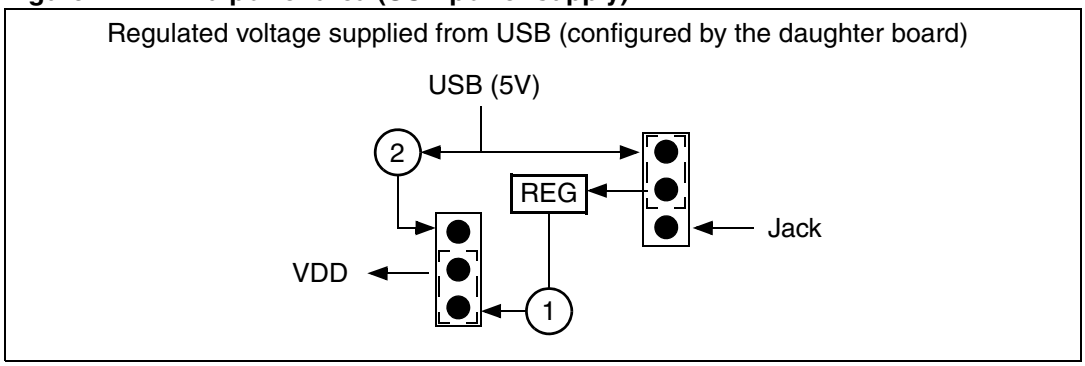
Jumper	Setting	Purpose
D7	Unfitted	Exclusively used for the driver handling communication over the UART.
D6	Unfitted	Exclusively used for internal driver purposes.
D5	Unfitted	Exclusively used for internal driver purposes.
D4	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D3	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D2	Fitted	Enable led D2 (available for application use).
D1	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
D0	Unfitted	Exclusively used for the SN260 to the ST microcontroller serial communication over the SPI interface.
BT5	Fitted	Enable the BT5 button (available for application use).
BT6	Fitted	Enable the BT6 button (available for application use). Note that the POL jumper also has to be fitted to |_| position.
POL	Fitted as |_|	Enable the BT6 button usage.
TX	Fitted	Enable serial transmission (available for application use).
RX	Fitted	Enable serial reception (available for application use).

2.3.3 REva power area

RIDE and Ride7 environments

When using the REva power jumper settings in a RIDE or Ride7 environment, keep the default configuration (USB power supply) as shown in *Figure 4*.

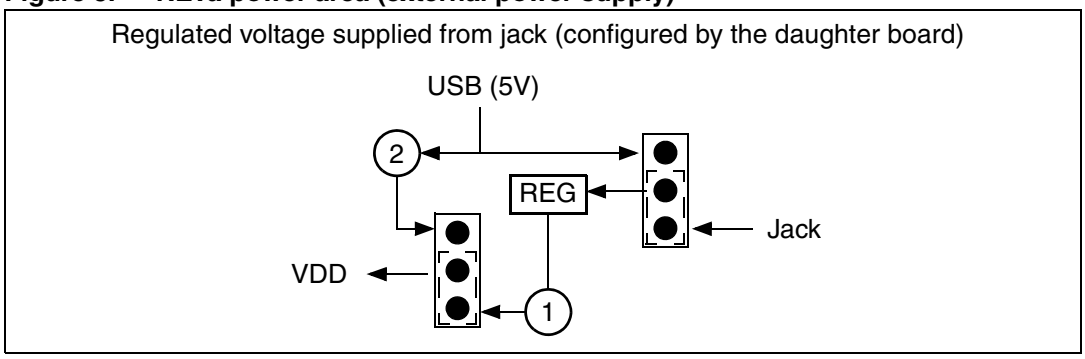
Figure 4. REva power area (USB power supply)



IAR environment (IAR J-Link JTAG emulator and external power supply)

When using the REva power jumper settings in an IAR environment, set the jumper for an external power supply as shown in *Figure 5*.

Figure 5. REva power area (external power supply)



2.3.4 Daughter board jumper settings

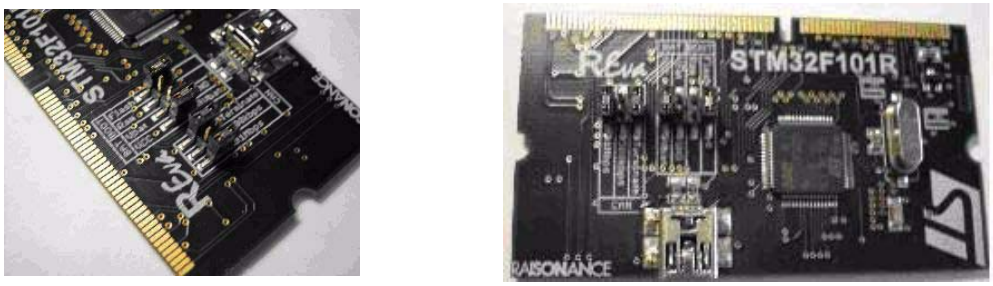
REva STM32F103x daughter board

For the REva STM32F103x daughter board, *Table 4* and *Figure 7* show the default settings.

Table 3. REva STM32F103x daughter board

Jumper	Setting	Purpose
FLASH	Fitted	Flash boot mode
RAM	Unfitted	RAM boot mode (the Flash boot mode is used)

Figure 6. Boot mode setting for STM32F103x daughter board



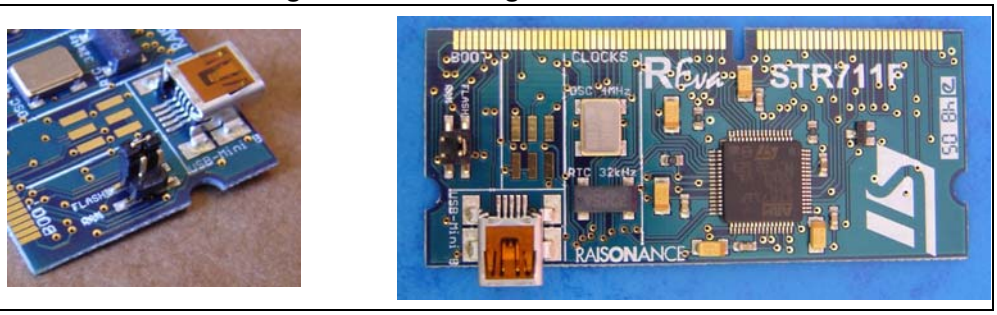
REva STR71x daughter board

For the REva STR71x daughter board, *Table 4* and *Figure 7* show the default settings.

Table 4. REva STR71x daughter board

Jumper	Setting	Purpose
FLASH	Fitted	Flash boot mode
RAM	Unfitted	RAM boot mode (the Flash boot mode is used)

Figure 7. Boot mode setting for STR71x daughter board



REva STR75x daughter board

For the REva STR75x daughter board, *Table 5* shows the default settings.

Table 5. REva STR75x daughter board

Jumper	Setting	Purpose
Boot0 = 0	Fitted	Flash boot mode
Boot0 = 1	Unfitted	Flash boot mode
Boot1 = 0	Fitted	Flash boot mode
Boot1 = 1	Unfitted	Flash boot mode

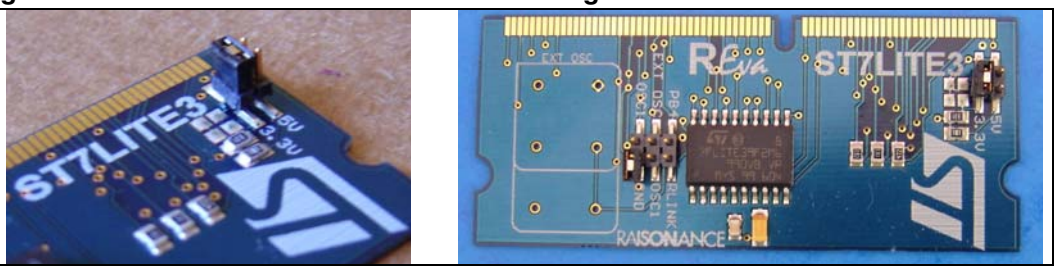
REva ST7LITE39 daughter board

For the REva ST7LITE39 daughter board, *Table 6* and *Figure 8* show the default settings.

Table 6. REva ST7LITE39 daughter board

Jumper	Setting	Purpose
5V	Unfitted	5V power supply
3.3V	Fitted	3.3V power supply

Figure 8. Power area for REva ST7LITE39 daughter board



Note: The STR91x cannot boot from RAM. As a consequence, no specific boot jumpers are present on the STR91x daughter board.

2.3.5 Raisonance RLink jumper settings

When using STM32F103x, STR71x, STR75x and STR91x daughter boards, just keep the default jumper settings as shown in *Figure 9*.

Figure 9. Power area for Raisonance RLink (STM32F103x, STR71x, -75x and -91x)

When using the ST7LITE39 daughter board, set the jumpers as shown in *Figure 10*.

Figure 10. Power area for Raisonance RLink (ST7LITE39)

2.3.6 Setting up the InSight adapter

As shipped, InSight adapters are configured with DHCP mode enabled and a preassigned host-name. Each InSight adapter carries a label with its host-name. Hostnames for all InSight adapters in the kit also appear on a printed hardware ID list, which are mapped to their EUI-64 identifiers, and the serial numbers of matching radio communication modules.

Note: InSight adapters are accessible through their assigned hostnames only if your DHCP system populates DNS records automatically. Otherwise, you must access them through their IP addresses. Alternatively, you can set up a hosts file on your local computer that maps host names to IP addresses. If you are using static IP addresses, make sure that they do not conflict with other network addresses.

Configure the InSight adapter for static IP addresses

In order to access to the administration interface on the ISA, via the USB connector, a driver is needed for the USB<->Serial converter used. This driver can be acquired through the FTDI website (http://www.ftdichip.com/Drivers/VCP.htm).

If your site is configured for static IP addresses the InSight Adapter can be configured for static IP address as follows:

- Connect the USB cable from your workstation to the InSight adapter's USB connector.
- From your workstation, run a standard terminal emulation program to connect to the InSight adapter (the InSight adapter appears as a COM port).
- Configure your terminal emulator with these settings:
  - 115200 (baud)
  - 8 (data bits)
  - n (no parity)
  - 1 (stop bit)
  - Flow Control = None
- Issue the following InSight adapter commands through the terminal emulator:

```
ip_static <ipaddress> <netmask> <gateway>
ip_dhcp off
```

- Reboot the application board by pressing the red RESET button on the front of the InSight adapter.

At the same way, it is possible to change the default hostnames assigned to the ISA modules following the same steps previously described and issuing this command:

```
hostname set <new-hostname>
```

To verify the new host name, issue the InSight adapter command config, which returns with the new hostname.

ISA hardware connections

To attach the InSight adapter follows these steps:

1. Plug the InSight port cable into the InSight adapter.
2. Plug the InSight port cable into the radio communication module.
3. Verify that the InSight adapter's RCM Power Select switch is set to `Int`.
4. Power on the Insight adapter (it obtains its power from the supplied AC adapter or the Power-over-Ethernet switch).

To remove the InSight adapter follows these steps:

1. Unplug the InSight port cable from the radio communication module.
2. Unplug the InSight port cable from the InSight adapter.
3. Power off the Insight adapter.

To perform a hardware reset of an InSight adapter, press its red reset button in front. You can reset the software for the InSight adapter simply connecting or disconnecting an Ethernet cable from the application board.

Note: To avoid communication problems, verify that each radio communication module is firmly seated in its application board connector, and its InSight port cable is properly seated.

2.4 Building the EZSP and HAL libraries

The library software relies on the software and build system framework introduced in the library layout section. What follows is a short introduction, aimed at getting enough knowledge to be able to build the software library using the RIDE or Ride7 toolset.

STM32F103x, STR7x-9x microcontrollers specific library building steps (Ride7 toolset)

1. Open the Ride7 toolset.
2. From the **Project**, **Open Project** menu, open the *ezsp\libs\ezsp_hal.rprj* Ride7 project.
3. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller, or STR912FW44 as STR91x microcontroller).
4. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Boot mode** and select **Flash** (the ST microcontroller daughter board Flash jumper has to be fitted).
5. From the **Options**, **Project Properties**, **GCC Compiler**, **Defines** menu, select **Defines** and add the ST_STM32 define value for the STM32F103x microcontroller or the ST_STR71X define value for the STR71x microcontroller or the ST_STR75X define value for the STR75x microcontroller or the ST_STR91X define value for the STR91x microcontroller.
6. From the **Options**, **Project Properties**, **GCC Compiler, Compiler Output** menu, open **Debugging Information** and select **No Debug.**
7. From the **Project** menu, select **Build Project**: a library *ezsp_hal.lib* is produced into the *ezsp\libs* directory.

ST7LITE39 microcontroller specific library building steps (RIDE toolset)

1. Open the RIDE toolset.
2. From the **Project**, **Open** menu, open the *ezsp\libs\ezsp_hal_ST7.prj* RIDE project.
3. From the **Options**, **Target**, **ST7 family** menu, open **Device** and select **ST7LITE39.**
4. From the **Options**, **Target**, **ST7 family** menu, open **Properties**, **Settings**, **Memory Model** and select **Stack Long.**
5. From the **Options**, **Project** menu, open **CXST7**, **Defines** and add the ST_ST7;CODECUT define values.
6. From the **Options**, **Project** menu, open **CXST7**, **More** and add: `-ga -oc +compact -pcp +split` to the More Controls window.
7. From the **Options**, **Project**, **CLNK**, **Hex File Generation** menu, open **Generate a HEX file** and select **Intel HEX format.**
8. From the **Options**, **Debug** menu, open **Real machine**, and select **Tools**: **RLink.**
9. From the **Project** menu, select **Build All.**
10. On the **Project** tab, double-click on the *make_library.wsc* script file: a library *ezsp_hal_ST7.lib* will be produced into the *ezsp\libs* directory.

2.5 Wireless application examples

Certain wireless applications are provided for allowing a ZigBee network to be set up using the included hardware and software components. The SNDEV-260 ZigBee kit components allow to setup and monitoring the following ZigBee networks:

- 1 sink and 2 sensors network
- 1 light and 2 switches network
- 2 or 3 nodes network performing the Range test application for RF evaluation

To set up a ZigBee network, the following steps are required:

1. Load each SN260-DEV kit node with the specific application.
2. Select a 3-node ZigBee network using the kit hardware and software tools (InSight Desktop).
3. Setup the serial communication channel as described in *Section 2.5.1*.

2.5.1 Setup the application serial communication channel

The provided application examples require displaying debugging messages and/or interaction with the user. Two serial communication channels are supported: serial COM through RS232 or virtual COM through USB.

Setup a serial COM through RS232 (for all microcontrollers)

1. Connect a serial cable between the PC serial port and the REva SER1 port.
2. Open a HyperTerminal on the serial COM port with the following configuration (the application messages and /or interactions come through the serial HyperTerminal):
   - **Bit rate:** 9600
   - **Data bits:** 8
   - **Parity:** None
   - **Stop bits:** 1
   - **Flow control:** None

Setup a virtual COM communication through USB (only for STM32F103x, and STR71x-91x microcontrollers)

1. Connect the USB-Mini cable to the STM32F103x or STR71x-91x USB mini-B connector and to a PC USB port. (The first time a STM32F103x or STR7x-STR9x USB device is plugged to the PC USB port, the user is required to install the relative USB software driver: select file stmcdc.inf in the serialUSB directory).
2. Using the mouse, right click on My Computer, select Manage, Device Manager, and open Ports (COM and LPT) to display a STR71x-91x CDC communication port on a specific COMx port. The STM32F103x or STR71x-91x USB device has been recognized.
3. Open a HyperTerminal on the corresponding USB virtual COMx port with the following configuration:
   - **Bit rate:** 9600
   - **Data bits:** 8
   - **Parity:** None
   - **Stop bits:** 1
   - **Flow control:** None

Reset a virtual COMx communication channel

1. Disconnect the COMx HyperTerminal.
2. Reset the REva board (STM32F103x or STR91x case) or Power OFF/ON the board (STR71x case).
3. Make a call on the COMx HyperTerminal: the application messages and /or interactions come through the COMx HyperTerminal.

Note: 1 When resetting the STR711FR2 microcontroller (through the REva RESET button), the PC is not able to enumerate again the STR71x USB device. To reset the STR71x USB device, power OFF/ON the REva board.

2 The PC is able to recognize only a single STM32F103x or STR71x-STR91x USB device when plugged on a PC USB port. When connecting a second STM32F103x or STR71x-STR91x USB device to another PC USB port, the PC will not recognize it.

3 If both the connectors (USB-mini and RS232) are connected, the virtual COM through USB communication is automatically selected by the application.

2.5.2 Sensor and sink applications

The sensor and sink applications set a distributed sensors network. They show how a single device collects data from multiple devices. In these simple applications, the 2 sensor nodes collect data and send it periodically to a sink node.

Build and download the sensor application on a single SNDEV-260 kit node

STM32F103x, STR7x-9x microcontrollers specific building steps (Ride7 toolset)

1. Open the Ride7 toolset and open the Ride7 ezsp_hal.rprj project.
2. Compile the library following the instructions on *Section 2.4: Building the EZSP and HAL libraries on page 21*.
3. From the **Project**, **Open Project** menu, open the *app\sink_sensor\app\sensor.rprj* Ride7 project.
4. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).
5. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Boot mode** and select **Flash** (the ST microcontroller daughter board Flash jumper has to be fitted).
6. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
7. From the **Options**, **Project Properties**, **GCC Compiler**, **Defines** menu, select **Defines** and add the ST_STM32 define value for the STM32F103x microcontroller, the ST_STR71X define value for the STR71x microcontroller, the ST_STR75X define value for the STR75x microcontroller or the ST_STR91X define value for the STR91x microcontroller.
8. From the **Options**, **Project Properties**, **GCC Compiler**, **Defines** menu, select **Defines** and add the SENSOR_APP; EZSP_HOST define values.
9. From the **Options**, **Project Properties**, **GCC Compiler**, **Compiler Output** menu, open **Debugging Information** and select **No Debug.**

10. From the **Options**, **Project Properties**, **LD Linker**, **Scripts** menu, open **Starter Kit Limitation** and select **NO**.
11. From the **Project** menu, select **Build Project**: a binary file is produced into the *app\sink_sensor\app*directory.

ST7LITE39 microcontroller specific building steps (RIDE toolset)

The sink and sensor applications are not supported by the ST7LITE39 microcontroller.

To download the sensor application on a single SNDEV-260 kit node

1. Connect the REva USB cable to the PC and to the RLink.

STM32F103x, STR7x-9x microcontrollers specific running steps (Ride7 toolset)

2. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open **Advanced Options, Options dialog box and** select **Debug** for the STM32F103x microcontroller and unselect **Debug** for the STR7x-STR9x microcontrollers.
3. From the **Debug** menu, select **Start**. The sensor application image is then downloaded.
4. Using the STM32F103x microcontroller, open the **Debug** menu, select **Run** and then **Terminate.**

ST7LITE39 microcontroller specific running steps (RIDE toolset)

The sink, sensor applications are not supported for the ST7LITE39 microcontroller.

Further steps for all ST microcontrollers

5. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
6. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.
7. Follow the messages displayed in the HyperTerminal window for interacting with the sensor node and to see how this node communicates with a sink node (the sensor node expects a sink node to send data to).

Note: For building and downloading the sink application just open the app\sink_sensor\app\sink.rprj Ride7 project. Follow the same steps as for the sensor application, adding the SINK_APP define value instead of the SENSOR_APP one. Furthermore, another PC serial communication channel (serial RS232 or virtual COM through USB) is required for user interaction with the sink application.

For a description of the sink and sensor applications, refer to the relative documentation.

The sensor and sink applications are not supported by the ST7LITE39 microcontroller due to its smaller Flash (only 8 Kbytes) and RAM (only 384 bytes).

Load two sink and sensor pre-built images (RIDE or Ride7 toolset)

Two pre-built images of the subscribed sensor and sink applications images are already present in the bin directory. To download the sensor.hex image into the ST microcontroller Flash, follows these steps:

1. Connect the REva USB cable between the PC and the RLink.

STM32F103x, STR7x-9x microcontrollers specific steps

1. Open the Ride7 toolset.
2. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x

microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).

3. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
4. From the **Options**, **Project Properties**, **Cortex RLink**, **Advanced Options** menu, open the options dialog box and deselect **Debug**.
5. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box and select **Erase target now!**
6. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box**,** select **Write Target Flash Now** and choose the *bin\sensor.hex* image file. Wait for the image to download into the Flash memory.

ST7LITE39 microcontroller specific steps

The sensor, sink applications are not supported for the ST7LITE39 microcontroller

Further steps for all ST microcontrollers

7. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
8. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

Follow the same steps for configuring another REva board as sink node (using the bin\sink.hex pre-built image). Additionally, another PC serial communication channel (serial RS232 or virtual COM through USB) is required for user interaction with the sink node.

STM32F103x and STR71x-STR9x microcontrollers specific building steps (IAR toolset)

1. Open the IAR toolset.
2. From the **File**, **Open**, **Workspace** menu, open the app\sink_sensor\app\IAR\sensor.eww IAR workspace.
3. From the **View** menu, select **Workspace** to display the supported projects. (A window is displayed on the left side of the IAR environment.)
4. From the **Workspace** selector window, select the application configuration according to the microcontroller to be addressed: sensor_STM32 for the STM32F103x microcontroller configuration, sensor_STR71x for the STR71x microcontroller configuration or sensor_STR91x for the STR91x microcontroller configuration.
5. From the **Project** menu, select **Rebuild All**. An Intel HEX file, sensor_STM32.hex, sensor_STR71x.hex or sensor_STR91x.hex, is built in the directory app\sink_sensor\app\IAR.

To download the sensor application

1. From the **Project** menu, select **Debug**. Wait for the image to download into the Flash memory.
2. From the **Debug** menu, select **Stop Debugging**.
3. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
4. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

When building and running the sink application, just open the app\sink_sensor\app\IAR\sink.eww IAR workspace and follow the same steps as for the sensor application.

2.5.3 Light and switch applications

The light and switch applications control the switching on/off of a light.

Build and download the light application on a single SNDEV-260 kit node

STM32F103x, STR7x-9x microcontrollers specific building steps (Ride7 toolset)

1. Open the Ride7 toolset and open the Ride7 ezsp_hal.rprj project.
2. Compile the library following the instructions in *Section 2.4: Building the EZSP and HAL libraries on page 21* and adding the following define value EMBER_SECURITY_LEVEL=0 (from **Options**, **Project Properties**, **GCC compiler**, and **Defines** menu).
3. From the **Project**, **Open Project** menu, open the *app\light\app\light.rprj* Ride7 project.
4. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).
5. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Boot mode** and select **Flash** (the ST microcontroller daughter board Flash jumper has to be fitted).
6. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
7. From the **Options**, **Project Properties**, **GCC Compiler**, **Defines** menu, select **Defines** and add the ST_STM32 define value for the STM32F103x microcontroller or the ST_STR71X define value for the STR71x microcontroller or the ST_STR75X define value for the STR75x microcontroller or the ST_STR91X define value for the STR91x microcontroller.
8. From the **Options**, **Project Properties**, **GCC Compiler**, **Compiler Output** menu, open **Debugging Information** and select **No Debug**.
9. From the **Options**, **Project Properties**, **LD Linker**, **Scripts** menu, open **Starter Kit Limitation** and select **NO**.
10. From the **Project** menu, select **Build Project**: a binary file is produced into the *app\light\app* directory**.**

ST7LITE39 microcontroller specific building steps (RIDE toolset)

1. Open the RIDE toolset and open the RIDE ezsp_hal_ST7.prj project.
2. Compile the library following the instructions on *Section 2.4 on page 21* and adding the following new define value: EMBER_SECURITY_LEVEL=0 (from Options, Project, CXST7, Defines)
3. From the **Project**, **Open** menu, open the *app\light\app\light_ST7.prj* RIDE project.
4. From the **Options**, **Target**, **ST7 family** menu, open **Device** and select **ST7LITE39.**
5. From the **Options**, **Target**, **ST7 family** menu, open **Properties**, **Settings**, **Memory Model** and select **Stack Long.**
6. From the **Options**, **Project** menu, open **CXST7**, **Defines** and add the ST_ST7;CODECUT define values.
7. From the **Options**, **Project** menu, open **CXST7**, **More** and add: `-ga -oc +compact -pcp +split` to the More Controls window.

8. From the **Options**, **Project**, **CLNK**, **Hex File Generation** menu, open **Generate a HEX file** and select **Intel HEX** format.
9. From the **Options**, **Debug** menu, open **Real machine**, and select **Tools**: **RLink.**
10. From **Project** menu, select **Build All:** a binary file version is produced in the *app\light\app* directory.

To download the light application on a single SNDEV-260 kit node

STM32F103x, STR7x-9x microcontrollers specific running steps (Ride7 toolset)

1. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the **options dialog box** and select **Debug** for the STM32F103x microcontroller and unselect **Debug** for the STR7x-STR9x microcontrollers.
2. From the **Debug** menu, select **Start**. The light application image is then downloaded.
3. Using the STM32F103x microcontroller, from the **Debug** menu, select **Run** and then **Terminate**.

ST7LITE39 microcontroller specific running steps (RIDE toolset)

4. From the **Debug** menu, select **Start** *light_st7.cos*. The light application image is then downloaded.

Further steps for all ST microcontrollers

5. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
6. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

Note: To build and download the switch application, open the app\light\app\switch.rprj (for STM32F103x or STR7x-9x Ride7 projects) or app\light\app\switch_ST7.prj (for ST7 RIDE projects). Follow the same steps as for the light application. Furthermore, another PC serial communication channel (serial RS232 or virtual COM through USB) is required for user interaction with the switch application.

For a description of the light and switch applications refer to the related documentation.

Load two light and switch pre-built images (RIDE or Ride7 toolset)

Two pre-built images of the subscribed light and switch applications images are already present in the *bin* directory. To download the *light.hex* (STM32F103x or STR7x-STR9x case) or the *light_st7.hex* (ST7 case) image into the ST microcontroller Flash and run the light application, follow these steps:

1. Connect the REva USB cable between the PC and the RLink.

STM32F103x and STR7x-9x microcontrollers specific steps

1. Open the Ride7 toolset.
2. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).
3. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
4. From the **Options**, **Project Properties**, **Cortex RLink**, **Advanced Options** menu, open the options dialog box and deselect **Debug**.

5. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box and select **Erase target now!**
6. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box**,** select **Write Target Flash Now** and choose the *bin\light.hex* image file. Wait for the image to download into the Flash.

ST7LITE39 microcontroller specific steps

7. Open the RIDE toolset.
8. From the **Options**, **Target**, **ST7 family** menu, open **Device** and select **ST7LITE39**.
9. From the **Options**, **Debug** menu, open **Real machine**, **Tools** and select **RLink**.
10. From the **Options**, **Debug** menu, open **Advanced Options**, deselect **Debug**, click on **Write Target Flash Now** and select the *bin\light_st7.hex* image file. Wait for the image to download into the Flash.

Further steps for all ST microcontrollers

11. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
12. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

Follow the same steps for configuring another REva board as switch node, using the *bin\switch.hex* (for STM32F103x or STR7x-STR9x devices) or *bin\switch_st7.hex* (for ST7 devices) pre-built image. Additionally, another PC serial communication channel (serial RS232 or virtual COM through USB) is required for user interaction with the switch node.

STM32F103x and STR71x-STR9x microcontrollers specific building steps (IAR toolset)

1. Open IAR toolset.
2. From the **File**, **Open**, **Workspace** menu, open the app\light\app\IAR\light.eww IAR workspace.
3. From the **View** menu, select **Workspace** to display the supported projects. (A window is displayed on the left side of the IAR environment.)
4. From the **Workspace** selector window, select the application configuration according to the microcontroller to be addressed: light_STM32 for the STM32F103x microcontroller configuration, light_STR71x for the STR71x microcontroller configuration, light_STR91x for the STR91x microcontroller configuration.
5. From the **Project** menu, select **Rebuild All**. An Intel HEX file, light_STM32.hex, light_STR71x.hex or light_STR91x.hex, is built in the directory app\light\app\IAR.

To download the light application

1. From the **Project** menu, select **Debug**. Wait for the image to download into the Flash memory.
2. From the **Debug** menu, select **Stop Debugging**.
3. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
4. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

When building and running the switch application, just open the app\light\app\IAR\switch.eww IAR workspace and follow the same steps as for the light application.

2.5.4 Range test application

The range test application is a special application which allows performing a simple send/receive test on the SN260 device to determine its range and generally test its radio functions.

The range tests application is provided only in binary format and it is included in the kit bin subdirectory. It requires a minimum of two nodes. The range test application runs directly over the SN260 module (no interaction with the ST microcontroller is required) so the application needs to be downloaded on it. The following sections describe how download the range test application inside the SN260 processor.

Preliminary steps

1. Unplug the ST microcontroller daughter board from the application board (do not power it).
2. Connect the SN260 RCM module to the application board.
3. Set each of the ISA Power Select to `Int` position.
4. Connect the InSight Port cable to the SN260 RCM InSight port connector and to the InSight Adapter connector.
5. Make sure that your InSight Adapter is receiving power and is accessible via your Local Area Network (LAN). For ISA powering, *Section 2.1.5: InSight adapter on page 7*.

At this point the range test application can be downloaded as follows:

Download the *em260-rangetest.hex* application (provided with the EmberZNet 3.x stack)

1. Open a new command prompt on the PC.
2. Determine the IP address of the InSight Adapter attached to the SN260 device being programmed.
3. Execute the em2xx_load command, specifying the IP address of the ISA attached to the target SN260 device. For example, to load a SN260 device connected to an ISA with IP 192.168.0.1, execute the following command:

```
em2xx_load.exe -sid 192.168.0.1 em260-rangetest.hex
```

After completing the device programming, a success message is displayed.

Download the *em260-haltest.xpv/xdv* application (provided with the EmberZNet 2.5.x stack)

Loading firmware via the InSight Desktop

1. Open the InSight Desktop toolset.
2. Verify that the Application loader path refers to the directory where the *em2xx_load.exe* file can be found (it is provided in the bin subdirectory).
3. Click on **Discover Adapter** button to refresh the list of available devices.
4. On the InSight desktop, right-click on the node and select **Connect** (wait for node connection).
5. On the InSight desktop, right-click on the node and **select uploading network coprocessor firmware**, select the *em260-haltest.xpv* file and wait for image download.

Loading firmware via the em2xx_load utility

1. Open a new command prompt on the PC.
2. Determine the IP address of the InSight Adapter attached to the SN260 device being programmed.
3. Execute the em2xx_load command, specifying the IP address of the ISA attached to the target SN260 device prefaced by the -sid argument. For example, to load a SN260 device connected to an ISA with IP 192.168.0.1, execute the following command:

```
em2xx_load -sid 192.168.0.1 C:\SNDEV-260\bin\em260-haltest.xpv
```

After the device programming complete, a success message is displayed.

2.5.5 Version application

The version application is a simple application which allows some basic ZigBee operations to be performed:

- EmberZnet Stack initialization
- ezsVersion for getting the EmberZnet stack version running on the SN260 silicon
- Get the Eui64 node address

As consequence, it could be used only for checking that the application board, ST microcontroller and SN260 RCM modules are correctly connected and communicating. It doesn't allow exploiting any ZigBee networking feature.

Build and run the version application

A short introduction follows about how to build and run the application.

STM32F103x and STR7x-9x microcontrollers specific building steps (Ride7 toolset)

1. Open the Ride7 toolset and open the Ride7 ezsp_hal.rprj project.
2. Compile the library following the instructions on *Section 2.4: Building the EZSP and HAL libraries on page 21*.
3. From the **Project**, **Open Project** menu, open the *app\version\app\version.rprj* Ride7 project.
4. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).
5. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Boot mode** and select **Flash** (the ST microcontroller daughter board Flash jumper has to be fitted).
6. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
7. From the **Options**, **Project Properties**, **GCC Compiler**, **Defines** menu, select **Defines** and add the ST_STM32 define value for the STM32F103x microcontroller or the ST_STR71X define value for the STR71x microcontroller or the ST_STR75X define value for the STR75x microcontroller or the ST_STR91X define value for the STR91x microcontroller.
8. From the **Options**, **Project Properties**, **GCC Compiler**, **Compiler Output** menu, open **Debugging Information** and select **No Debug**.

9. From the **Options**, **Project Properties**, **LD Linker**, **Scripts** menu, open **Starter Kit Limitation** and select **NO**.
10. From the **Project** menu, select **Build Project**: a binary file is produced into the *app\version\app* directory.

ST7LITE39 microcontroller specific application building steps (RIDE toolset)

1. Open the RIDE toolset and open the RIDE *ezsp_hal_ST7.prj* project.
2. Compile the library following the instructions on *Section 2.4 on page 21*.
3. From the **Project**, **Open** menu, open the *app\version\app\version_ST7.prj* RIDE project.
4. From the **Options**, **Target**, **ST7 family** menu, open **Device** and select **ST7LITE39.**
5. From the **Options**, **Target**, **ST7 family** menu, open **Properties**, **Settings**, **Memory Model** and select **Stack Long.**
6. From the **Options**, **Project** menu, open **CXST7**, **Defines** and add the ST_ST7; CODECUT define values.
7. From the **Options**, **Project** menu, open **CXST7**, **More** and add: `-ga -oc +compact -pcp +split` to the More Controls window.
8. From the **Options**, **Project**, **CLNK**, **Hex File Generation** menu, open **Generate a HEX file** and select **Intel HEX** format.
9. From the **Options**, **Debug** menu, open **Real machine**, and select **Tools**: **RLink**.
10. From the **Project** menu, select **Build All**: a binary file version is produced into the *app\version\app* directory.

To run the version application

1. Connect the REva USB cable between the PC and the RLink.

STM32F103x and STR7x-9x microcontrollers specific running steps (Ride7 toolset)

1. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box and select **Debug** for STM32F103x microcontrollers and unselect **Debug** for STR7x-STR9x microcontrollers.
2. From the **Debug** menu, select **Start**. The version application image is then downloaded.
3. Using the STM32F103x microcontroller, from the **Debug** menu, select **Run** and then **Terminate**.

ST7LITE39 microcontroller specific running steps (RIDE toolset)

4. From the **Debug** menu, select **Start** *version_st7.cos*. The version application image is then downloaded.

Further steps for all ST microcontrollers

5. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
6. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

Certain messages are displayed in the PC HyperTerminal window.

Below is a log of a version application execution:

```
***** Stack initialization performed with success
***** Stack Version = 3059
***** EUI64 Node ID = 000D6F000009A340.
```

Note: The most recent EmberZNet 3.x stack version is 3059. The Ember ZNet 2.5.x stack version is 2553.

Load and run a version pre-built image (RIDE or Ride7 toolset)

A pre-built image of the version application is already present in the *bin* directory. To download the *version* image into the ST microcontroller Flash and run the version application, follow these steps:

1. Connect the REva USB cable between the PC and the RLink.

STM32F103x and STR7x-9x microcontrollers specific steps

1. Open the Ride7 toolset.
2. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Processor** menu, open **Device** and select the specific STM32F103x or STR7x-9x microcontroller (STM32F103RBT6 as STM32F103x microcontroller, STR711FR2 as STR71x microcontroller, STR750FV2 as STR75x microcontroller or STR912FW44 as STR91x microcontroller).
3. From the **Options**, **Project Properties**, **Advanced ARM Options**, **Debug environment** menu, open **Debug tool** and select **RLink**.
4. From the **Options**, **Project Properties**, **Cortex RLink**, **Advanced Options** menu, open the options dialog box and deselect **Debug**.
5. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box and select **Erase target now!**
6. From the **Options, Project Properties**, **Cortex RLink**, **Advanced Options** menu**,** open the options dialog box**,** select **Write Target Flash Now** and choose the *bin\version.hex* image file. Wait for the image to download into the Flash.

ST7LITE39 microcontroller specific steps

1. Open the RIDE toolset.
2. From the **Options**, **Target**, **ST7 family** menu, open **Device** and select **ST7LITE39**.
3. From the **Options**, **Debug** menu, open **Real machine**, **Tools** and select **RLink**.
4. From the **Options**, **Debug** menu, open **Advanced Options**, deselect **Debug**, click on **Write Target Flash Now** and select the *bin\version_st7.hex* image file. Wait for the image to download into the Flash.

Further steps for all ST microcontrollers

5. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
6. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

STM32F103x and STR71x-STR9x microcontrollers specific building steps (IAR toolset)

1. Open the IAR toolset.
2. From the **File**, **Open**, **Workspace** menu, open the app\version\app\IAR\version.eww IAR workspace.
3. From the **View** menu, select **Workspace** to display the supported projects. (A window is displayed on the left side of the IAR environment.)
4. From the **Workspace** selector window, select the application configuration according to the microcontroller to be addressed: version_STM32 for the STM32F103x

microcontroller, version_STR71x for the STR71x microcontroller configuration or version_STR91x for the STR91x microcontroller configuration.

5. From the **Project** menu, select **Rebuild All**. An Intel HEX file, version_STM32.hex, version_STR71x.hex or version_STR91x.hex, is built in the directory app\version\app\IAR

To download the version application

1. From the **Project** menu, select **Debug**. Wait for the image to download into the Flash memory.
2. From the **Debug** menu, select **Stop Debugging**.
3. Set up the serial communication channel as described in *Section 2.5.1 on page 22*.
4. If using the virtual COM, reset the corresponding communication channel as described in *Section 2.5.1 on page 22*.

3 Setting up a network

3.1 Setting up a sink and sensor ZigBee network

Using the SNDEV-260 kit, it is possible to setup a 1-sink, 2-sensor ZigBee network by following these steps:

1. Build the sink application following the HW/SW instructions included in *Section 2.5.2: Sensor and sink applications on page 23* and download it in one of the three SNDEV-260 ZigBee Kit nodes.
2. Build the sensor application following the HW/SW instructions included in *Section 2.5.2: Sensor and sink applications on page 23* and download it into the other two SNDEV-260 ZigBee Kit nodes.
3. Set each of the ISA Power Select to `Int` position.
4. Connect the InSight Port cable to the SN260 RCM InSight port connector and to the InSight Adapter connector.
5. Connect each of the 3 ISAs to the Ethernet switch through standard Ethernet cables.
6. Power on the Ethernet switch; the connected ISAs are powered.
7. Open the InSight Desktop on your PC: if the applications are using an encryption key (such as the sensor and sink one) set the key as follows:

   From File Preferences Decoding Decryption Key, insert your encryption key.
8. On the ISD, Adapters window select **Default Group**: the 3 ISA connected to the Ethernet are displayed.
9. Right click on each adapter and select **Connect**.
10. Right click on each adapter and select **Connect Start Capture**.
11. Reset each node (see *Section 2.5.1 on page 22*). The sink and sensor applications start execution.

The distributed sensor network runs as follows:

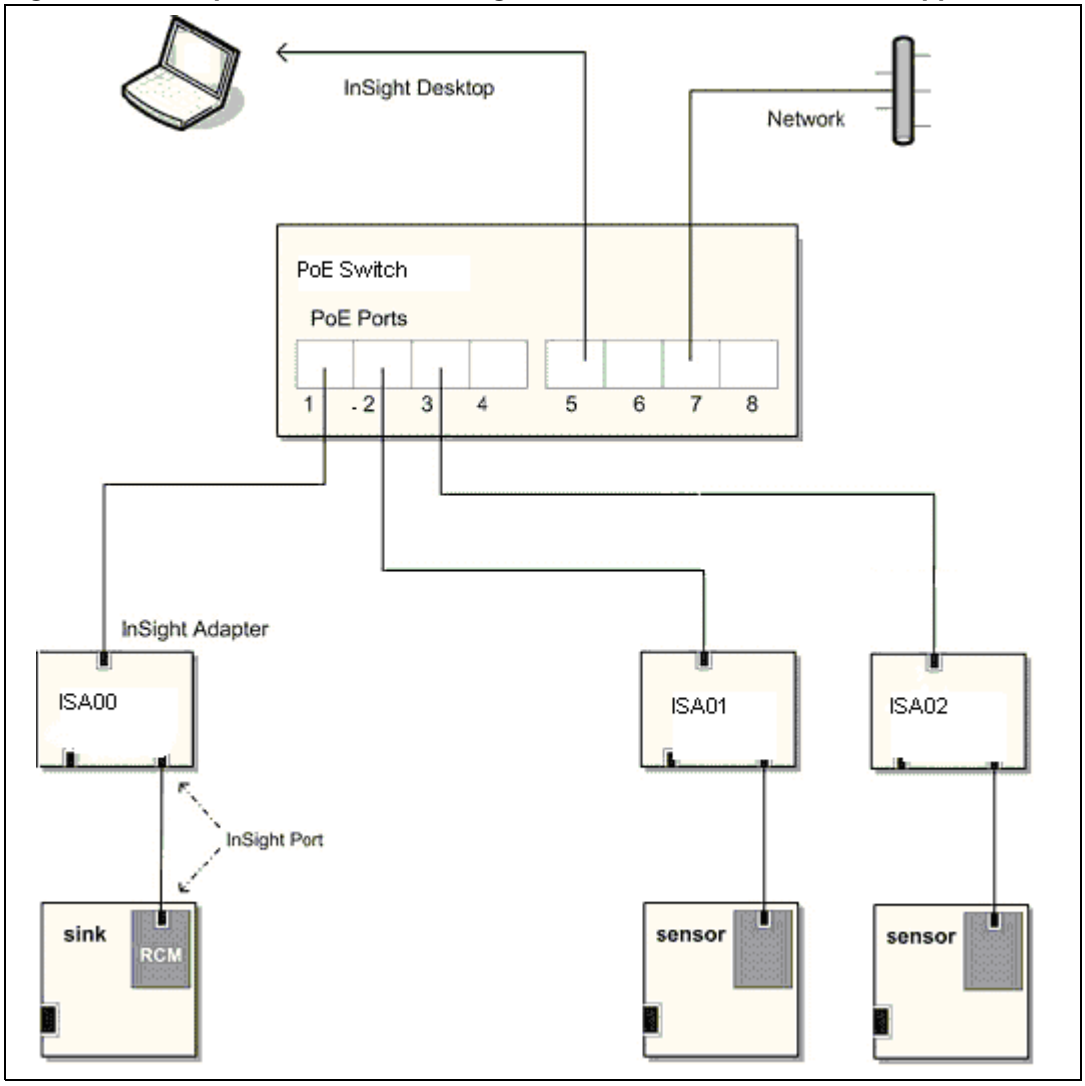
- A ZigBee network is created by the sink node.
- After the network if formed, the sink node sends its identity (EUI64 address) to the available sensors that can set up a binding to it and sending a request for communicate with it. After the sensor receives the confirmation message from the sink, it will begin transmitting some data to the sink.

*Note: The sink and sensor applications also allow user interaction through the serial communication channel (see Section 2.5.1 on page 22). Press the **?** key on the HyperTerminal to display the user help menu.*

When the applications start the execution, data are displayed on the InSight Desktop windows (map, transactions, events) showing different levels of details about the ongoing networking communication.

Figure 11 provides a high level view of the SNDEV-260 hardware components and connections when using with the sink, sensors wireless application examples.

Figure 11. Setup of the SNDEV-260 ZigBee network: sink and sensors applications



3.2 Setting up a light and switch ZigBee network

Using the SNDEV-260 kit, it is possible to setup a 1-light, 2-switch ZigBee network by following these steps:

1. Build the light application following the HW/SW instructions in *Section 2.5.3: Light and switch applications on page 26* and download it into one of the three SNDEV-260 ZigBee Kit nodes.
2. Build the switch application following the HW/SW instructions in *Section 2.5.3: Light and switch applications on page 26* and download it into the other two SNDEV-260 ZigBee Kit nodes.
3. Set each of the ISA Power Select to `Int` position.
4. Connect the InSight Port cable to the SN260 RCM InSight port connector and to the InSight Adapter connector.
5. Connect each of the three ISAs to the Ethernet switch through standard Ethernet cables.
6. Power on the Ethernet switch; the connected ISAs are powered.
7. Open the InSight Desktop on your PC.
8. On the ISD Adapters window, select **Default Group**; displaying the three ISA connected to the Ethernet.
9. Right click on each adapter and select **Connect**.
10. Right click on each adapter and select **Connect Start Capture**.
11. Reset each node (see *Section 2.5.1 on page 22*). The light and switch applications start execution.

The light and switch network runs as follows:

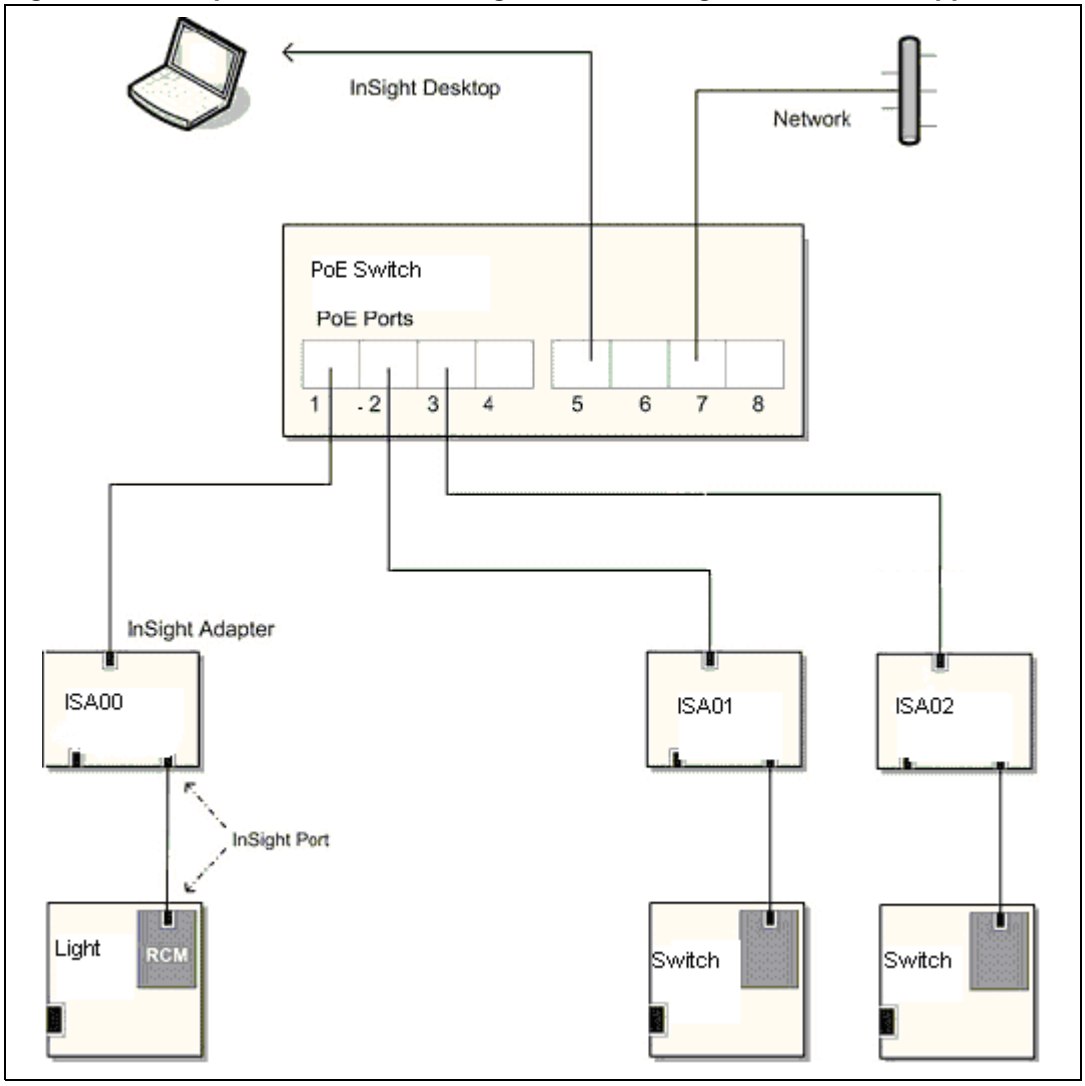
- A ZigBee network is created by the light node.
- After the network is formed, the light node allows a switch node to join the network pressing a specific button on the REva board. Each switch nodes controls the light by sending some messages to switch on or off a led present into the REva light board.

*Note: The light and switches applications also allow the user interaction through the serial communication channel (see Section 2.5.1 on page 22). Press the **?** key on the hyper terminal to display the user help menu.*

When the applications start the execution, data are displayed on the InSight Desktop windows (map, transactions, events) showing different levels of details about the ongoing networking communication.

Figure 12 provides a high level view of the SNDEV-260 hardware components and connections when using with the light, switches wireless application examples.

Figure 12. Setup of the SNDEV-260 ZigBee network: light and switches applications



3.3 Setting up a ZigBee network for RF testing

Using the SN260DEV kit, it is possible to setup a 2- or 3-node (1 transmitter + 1/2 receivers) ZigBee network by following these steps:

1. Download the range test application image into both SNDEV-260 Kit modules following the instructions in *Section 2.5.4: Range test application on page 29*.
2. On the ISD Adapters window, select **Default Group**: the 2 or 3 ISAs connected to the Ethernet are displayed.
3. Right-click on each adapter and select **Connect**.
4. Right-click on each adapter and select **Connect Start Capture**.
5. For each network node, open a telnet connection to the TCP port 4900 on the InSight Adapter where the node is connected. For example, for an SN260 device connected to an InSight Adapter with the IP address 192.168.0.1, you would enter the following text at a Windows command prompt:

   ```
   telnet 192.168.0.1 4900
   ```
6. Press **Enter** in each telnet windows to start the range test application.
7. Press **?** to display the available applications commands.
8. Type `calchannel 15` (just an example) on each telnet terminal.
9. Type `receive` on one terminal (it will be ready to receive packets and display statistics for each one).
10. Type `transmit 64` (just an example) on the other terminal: 100 packets will be transmitted.

The range test network runs as follows:

- Packets are sent by the transmit node.
- The receiver terminal will display the statistics of the received packets.

Note: For a description of the range test statistics refers to the application description in directory docs/Applications_Description.

3.4 Monitoring network activity

After the ZigBee network is formed, the InSight Desktop tool can be used to monitor and debug the network activity. The InSight Desktop displays the overall node and network activities in realtime and in a non intrusive way. It provides a rich and flexible interface to Ember embedded networks, which helps you develop and debug new network applications.

The InSight Desktop access to the network nodes using the relative InSight Adapter hardware.

InSight Desktop displays data in the following views and panels:

- **Adapters** list all InSight adapters that are connected to the specified Ethernet subnet. Map provides a graphical view of the network, where nodes are displayed with their network identifiers. The GUI interface also displays network activity.
- **Transactions** displays line items for all event transactions. For example, all events that pertain to a node joining a network are regarded as a single Associate transaction. These events include all requests and replies that are sent by the node offering permission, and the node petitioning to join.
- **Events** displays line items for packets that are transmitted during a capture session. All events that belong to the selected transaction are marked with a red dot.
- **Event Detail** displays the decoded header data of the selected packet.
- **Hex Dump** displays the hexadecimal header data of the selected packet

Filtering events

By default, the **Events** pane displays all session events. You can build and apply filters that constrain InSight Desktop to show only those events that are of interest. By including only events that are of interest, you can analyze results more efficiently.

You apply filters to each session individually; so different sessions can have their own filter settings. When you change a session's filters, InSight Desktop immediately refreshes the display. When you exit InSight Desktop, all session filters are cleared and must be reapplied when you restart.

For example, to filter out neighbor exchanges where nodes negotiate efficient routes between them in a series of messages, simply open the **Filters** menu and select **Hide Neighbor Exchange**. This filter is defined in the Filter Manager as follows:

```
!isPresent(neighborExchange)
```

The InSight Desktop immediately applies this filter, showing only those events that it evaluates as “True” for this expression (all events that are not neighbor exchanges).

Note: The Filters menu is preset to show several common event types. You can customize this menu and saved filter definitions using the Filter Manager.

Customizing the InSight Desktop View

InSight Desktop provides a number of options that you can use to customize your view of devices. For example, you can create a logical group of devices.

When InSight Desktop initially detects devices connected to the Ethernet, it displays them in the Default folder. You can create custom device folders and move devices in them:

1. From the Adapters toolbar, click **Add group**.
2. If desired, rename the New Group folder.

Note: Custom device folders have no functional impact on devices or their interaction.

InSight Desktop creates a new adapters folder New Group.

1. Select the folder name.
2. Type the folder's new name, then press **Enter**.
3. Click on a node to be moved and drag it to the new folder. For multiple selections, use Shift-Click and Ctrl-Click for contiguous and non-contiguous selections, respectively.

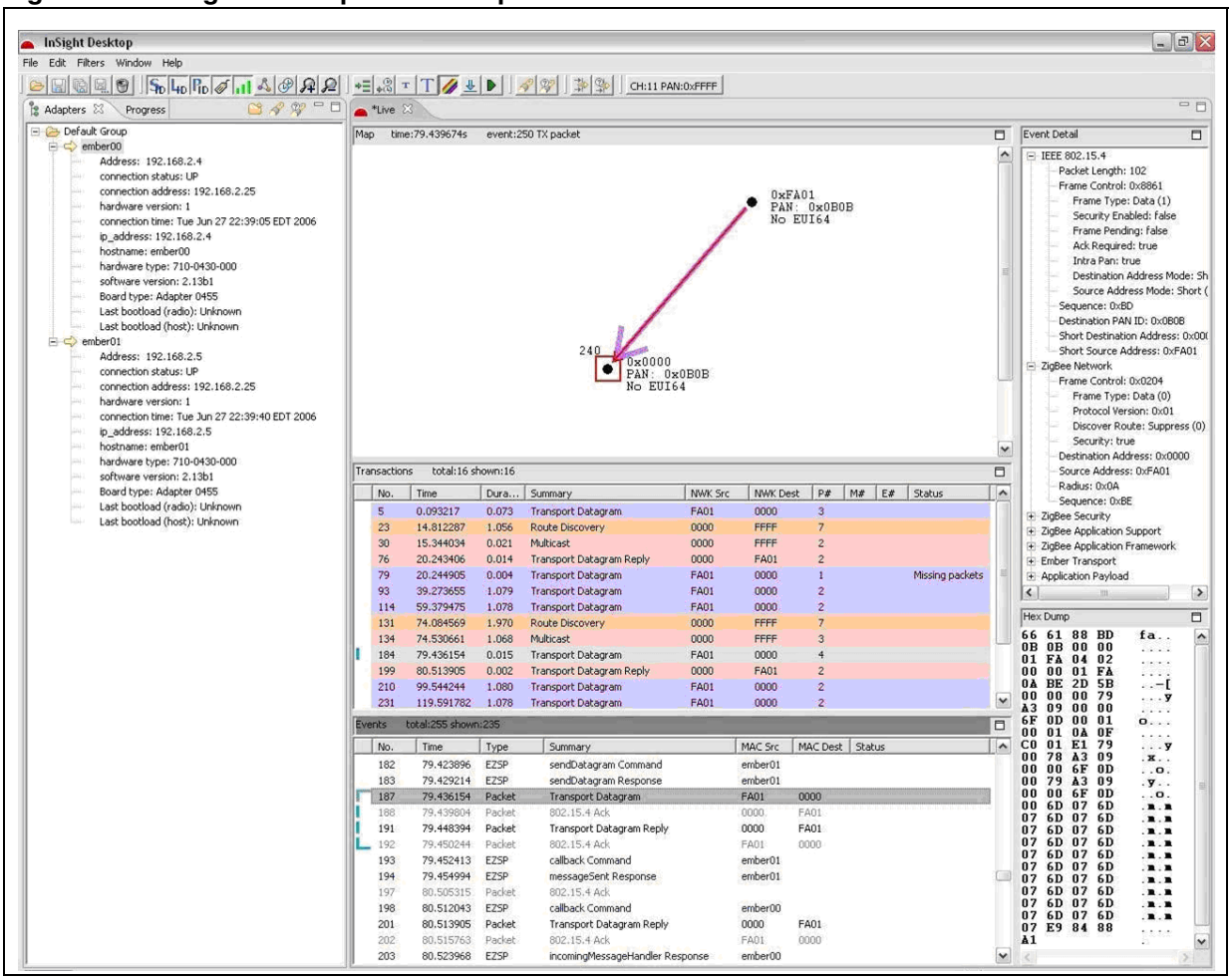
To remove a group folder:

1. In the **Adapters** menu, right-click on the folder to remove.
2. From the context menu, select **Delete InSight Desktop** to remove the group folder and move its devices to the Default group.

For a detailed description of the InSight Desktop and InSight Adapter, refer to the related technical documentation provided with the kit.

Figure 13 shows the InSight Desktop views and panels.

Figure 13. InSight Desktop views and panels



4 Updating the EmberZNet stack image

The following sections describe how update the EmberZNet stack image inside the SN260 network processor.

4.1 Update the stack image using the Insight adapter tool

Preliminary steps

1. Unplug the ST microcontroller daughter board from the application board (do not power it).
2. Connect the SN260 RCM module to the application board.
3. Set each of the ISA Power Select to `Int` position.
4. Connect the InSight Port cable to the SN260 RCM InSight port connector and to the InSight Adapter connector.
5. Make sure that your InSight Adapter is receiving power and is accessible via your Local Area Network (LAN). For ISA powering information, see *Section 2.1.5: InSight adapter on page 7*.

EmberZNet 3.x stack image update

The Ember ZNet 3.x stack is provided in the hex format (*em260-spi.hex* file). To update the SN260 processor with the Ember ZNet 3.x stack image, follows these steps:

6. Open a new command prompt on the PC.
7. Determine the IP address of the InSight Adapter attached to the SN260 device being programmed.
8. Execute the em2xx_load command, specifying the IP address of the ISA attached to the target SN260 device. For example, to load a SN260 device connected to an ISA with IP 192.168.0.1, execute the following command:

```
em2xx_load.exe -sid 192.168.0.1 em260-spi.hex
```

After the device programming is complete, a success message is displayed.

EmberZNet 2.5.x stack image update

The Ember ZNet 2.5.x stack is provided in XPV/XDV format (*em260.xpv/xdv* files). To update the SN260 processor with the Ember ZNet 2.5.x stack image, follow these steps:

Loading firmware via the InSight Desktop

1. Open the InSight Desktop toolset.
2. Verify that the Application loader path refers to the directory where the *em2xx_load.exe* file can be found (it is provided in the bin subdirectory).
3. Click on discover adapter button to refresh the list of available devices.
4. On the InSight desktop, right clicking on the node and select **Connect** (wait for node connection).
5. On the InSight desktop, right-click on the node and select **uploading network coprocessor firmware** and select the *em260.xpv* file and wait for image download.

Loading firmware via the em2xx_load utility

1. Open a new command prompt on the PC.
2. Determine the IP address of the InSight Adapter attached to the SN260 device being programmed.
3. Execute the em2xx_load command, specifying the IP address of the ISA attached to the target SN260 device prefaced by the -sid argument. For example, to load a SN260 device connected to an ISA with IP 192.168.0.1, execute the following command:

```
em2xx_load -sid 192.168.0.1 C:\SNDEV-260\bin\em260.xpv
```

After the device programming complete, a success message is displayed.

4.2 SPI bootloader

The EmberZNet 3.0.2 stack supports the bootloader through the SPI interface feature. Basically, it is possible to update the SN260 ZigBee stack image by running a simple application on the STM32F103RBT6 microcontroller which is able to put the SN260 in bootloader mode and then sending the new stack image packets through the SPI interface. This application requires the SN260 previously loaded with a stack version (EmberZNet 3.0.1 or more recent) supporting the bootloader.

The upload process consists of the following steps:

1. Open a DOS command window.
2. Go to the directory bin/spi_bootloader/
3. Type the following command to load the STM32_upload_stack_3_0_2_spi.hex application into the STM32F103RBT6 microcontroller Flash memory:

```
run.bat STM32 STM32_upload_stack_3.0.2_spi.hex
```

During the upload process, LEDs D7 and D6 on the REva board indicate the process status:

- LED D7 is ON: Process initialization OK
- LED D6 blinks every 2 seconds and LED D7 is ON: Upload IN PROGRESS
- LED D7 is OFF and LED D6 blinks every quarter second: Upload FAILED
- LED D7 and D6 blink alternatively every quarter second: Upload process completed with SUCCESS.

The user can also plug a serial cable to the REva SER1 port to receive upload status messages over the corresponding serial HyperTerminal.

The HyperTerminal settings are:

- Bit rate: 9600
- Data bits: 8
- Parity: None
- Stops bits: 1
- Flow control: None

5 Limitations and support

For any questions about the library and issue notifications, refer to the specific link on the STMicroelectronics web site.

6 Revision history

Table 7. Document revision history

Date	Revision	Changes
31-May-2007	1	Initial release.
17-Sep-2007	2	Power selection jumper on the ST7LITE39 daughter board set to 3.3V (Default configuration at delivery) and not to 5V. Added *Figure 7: Boot mode setting for STR71x daughter board on page 17* and *Figure 8: Power area for REva ST7LITE39 daughter board on page 18*.
17-Oct-2007	3	Document updated adding full scope support for the most recent 3.x stack with the SPI bootloader feature. Added IAR toolset support and virtual COM through USB communication. Reviewed REva power area description and library/applications building/running steps.
7-Jan-2008	4	Document updated adding full scope support for the STM32F103RBT6 microcontroller.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com