APPLICATION NOTE

AVR2131: Lightweight Mesh Getting Started Guide

Atmel MCU Wireless

Features

- Atmel® Lightweight Mesh Software Development Kit (SDK)
- WSNDemo sample application
- Custom applications

Description

The purpose of this application note is to introduce users to the Lightweight Mesh network protocol stack and typical application development process from Atmel. This document describes how to start quickly with the Lightweight Mesh SDK, by setting up the development environment and programming devices with sample applications.

To find more detailed information about the Lightweight Mesh architecture and application development process, refer to [1].

Table of Contents

1. Introduction

Atmel Lightweight Mesh is an easy to use proprietary low power wireless mesh network protocol. Lightweight Mesh was designed to address the needs of a wide range of wireless connectivity applications. Some of these applications include:

- Remote control
- Alarms and security
- Automatic Meter Reading (AMR)
- Home and commercial building automation
- Toys and educational equipment

Lightweight Mesh is designed to work with all Atmel IEEE® 802.15.4 transceivers and SoCs. Currently the stack works with AVR®- and ARM®-based MCUs, but given extreme portability and low resource requirements, it can be run on almost any Atmel MCU. Table 1-1 gives a summary of the currently supported hardware platforms.

Table 1-1. Supported Hardware Platforms

Board or module	Ordering code	MCU	Radio Transceiver
ZigBit® 2.4GHz Module with Balanced RF Output ZigBit 2.4GHz Module with Dual Chip Antenna	ATZB-24-B0 ATZB-24-A2	ATmega1281	AT86RF230B
ZigBit 2.4GHz Module with Chip Antenna	ATZB-X0-256-3-0-C	ATxmega256A3U	AT86RF233
ATmega128RFA1 Evaluation Kit	ATAVR128RFA1-EK1	ATmega128RFA1	ATmega128RFA1
XMEGA®-B1 Xplained and RZ600 radio modules	ATXMEGAB1-XPLD ATAVRRZ600	ATxmega128B1	AT86RF212, AT86RF231
RCB128RFA1	Part of the ATRF4CE-EK	ATmega128RFA1	ATmega128RFA1
RCB231	http://www.dresden-elektronik.de	ATmega1281	AT86RF231
ATmega256RFR2 Xplained Pro	ATMEGA256RFR2-XPRO	ATmega256RFR2	ATmega256RFR2
RCB256RFR2	http://www.dresden-elektronik.de	ATmega256RFR2	ATmega256RFR2
SAMD20 Xplained Pro and RZ600 radio modules	ATSAMD20-XPRO ATAVRRZ600	ATSAMD20 J18	AT86RF231
SAMD20 Xplained Pro and REB233-XPRO radio module	ATSAMD20-XPRO ATREB233-XPRO	ATSAMD20 J18	AT86RF233
SAMR21 Xplained Pro	ATSAMR21-XPRO	ATSAMR21G18 (via ATSAMD21J18)	ATSAMR21G18 (via AT86RF233)

All demonstrations in this document will use the RCB128RFA1 board [3] and the WSNDemo sample application as an example, but the same techniques can be applied to any other development kit, or a custom board and application.

2. Development Tools

A development toolchain consists of:

- An integrated development environment (for example, Atmel Studio or IAR Embedded Workbench®), where sample applications may be modified, compiled, and debugged,
- a corresponding compiler toolchain (AVR-GCC, IAR™), which provides everything necessary to compile application source code into binary images, and
- a programming device (for example, JTAG), which may be used to program and debug the application on a target platform

IAR Embedded Workbench for Atmel AVR [4] can be used to develop and debug applications for AVR-based platforms. The IAR IDE support's editing of application source code, compilation, linking object modules with libraries, and application debugging.

Atmel Studio 6 [6] can be used to develop and debug applications for AVR-based platforms. Atmel Studio is equipped with the GCC toolchain and does not require external tools to compile Lightweight Mesh applications.

3. WSNDemo Sample Application

The WSNDemo application implements a typical wireless sensor network scenario, in which one central node collects the data from a network of sensors and passes this data over a serial connection for further processing. In the case of the WSNDemo this processing is performed by the WSNMonitor PC application. The BitCloud® Quick Start Guide [2] provides a detailed description of the WSNDemo application scenario, and instructions on how to use WSNMonitor.

The majority of the information in [2] applies to the WSNDemo application running on top of Lightweight Mesh stack. However, since BitCloud is a ZigBee® PRO stack, there are a few differences in the protocol:

- Device types (Coordinator, Router, and End Device) are simulated on the application level; there is no such separation in Lightweight Mesh on the stack level
- The value of the extended address field is set equal to the value of the short address field
- For all frames, the LQI and RSSI fields are filled in by the coordinator with the values of LQI and RSSI from the received frame. This means that nodes that are not connected to the coordinator directly will have the same values as the last node on the route to the coordinator
- Sensor data values are generated randomly on all platforms

4. Using Provided Projects

4.1 Overview

Applications are located in the *apps* directory in the SDK. All sample applications in the Lightweight Mesh SDK come with the project files for Atmel Studio, IAR Embedded Workbench, and GNU make utility.

All Lightweight Mesh applications include a configuration file *config.h*. This file contains settings for the application and the stack. WSNDemo application settings are listed in Table 4-1. For system settings mentioned in the configuration file see [1].

Table 4-1. WSNDemo Application Settings

Parameter	Description
APP_ADDR	Node network address. This parameter also determines emulated device type: • 0x0000 – Coordinator • 0x0001-0x7fff – Router • 0x8000-0xfffe – End Device
APP_CHANNEL	Radio transceiver channel. Valid range for 2.4GHz radios is 11 – 26 (0x0b – 0x1a)
APP_PANID	Network identifier
APP_SENDING_INTERVAL	This parameter has a different meaning for different device types: • Coordinator: Interval between sending sensor values to the UART • Router: Interval between reporting sensor values to the coordinator • End Device: Sleep interval
APP_SECURITY_KEY	Security encryption key

Note: For normal network operation all devices should have different network addresses. There is no automatic address assignment mechanism, so it is the developer’s responsibility to ensure that addresses are unique.

Refer to the respective development environment documentation for the information on how to compile and debug projects.

Before programming compiled application into the chip make sure that Fuse bits are set correctly. Table 4-2 shows correct fuse bits settings for various platforms.

Table 4-2. Recommended Fuse Bits Settings

Board or module	MCU	Extended	High	Low
RCB128RFA1	ATmega128RFA1	0xFE	0x9D	0xC2
RCB231	ATmega1281	0xFE	0x9D	0xC2
ZigBit	ATmega1281	0xFE	0x9D	0xC2
ATmega256RFR2 Xplained Pro	ATmega256RFR2	0xFE	0x9D	0xC2
RCB256RFR2	ATmega256RFR2	0xFE	0x9D	0xC2

4.2 Running the Application

After all boards are programmed connect coordinator board to the PC and run the WSNMonitor application. Observe the coordinator and other node icons appearing on the screen. Refer to [2] for details on how to use the hardware and PC software.

5. Creating a New Application

5.1 Starting from a Template Application

The best way to start a new standalone application is to use the provided *Template* application as a base, and make custom modifications. Using template project files will ensure that all necessary components are included in the build, and that all required definitions are present. The template application can be found in the *<SDK Root>/apps/Template* directory.

5.2 Starting from Scratch

If Lightweight Mesh has to be integrated into a larger existing project, it is recommended to include all required files and definitions into the existing project. Table 5-1, Table 5-2, and Table 5-3 present a lists of files, include paths and definitions that are required for normal Lightweight Mesh operation. If platform with a standalone transceiver is used, then files, paths and definitions for the transceiver should be added to files, paths and definitions for the MCU.

Table 5-1. Required Files

MCU / RF transceiver	Files
All	<SDK Root>\nwk\src\nwk.c <SDK Root>\nwk\src\nwkDataReq.c <SDK Root>\nwk\src\nwkSecurity.c <SDK Root>\nwk\src\nwkFrame.c <SDK Root>\nwk\src\nwkGroup.c <SDK Root>\nwk\src\nwkRoute.c <SDK Root>\nwk\src\nwkRouteDiscovery.c <SDK Root>\nwk\src\nwkRx.c <SDK Root>\nwk\src\nwkTx.c <SDK Root>\sys\src\sys.c <SDK Root>\sys\src\sysTimer.c <SDK Root>\sys\src\sysEncrypt.c
ATmega1281	<SDK Root>\hal\atmega1281\src\hal.c <SDK Root>\hal\atmega1281\src\halPhy.c <SDK Root>\hal\atmega1281\src\halTimer.c

ATxmega128B1	<SDK Root>\hal\atxmega128b1\src\hal.c <SDK Root>\hal\atxmega128b1\src\halPhy.c <SDK Root>\hal\atxmega128b1\src\halTimer.c
ATxmega256A3U	<SDK Root>\hal\atxmega256a3u\src\hal.c <SDK Root>\hal\atxmega256a3u\src\halPhy.c <SDK Root>\hal\atxmega256a3u\src\halTimer.c
ATSAMD20J18	<SDK Root>\hal\atsamd20\src\hal.c <SDK Root>\hal\atsamd20\src\halPhy.c <SDK Root>\hal\atsamd20\src\halTimer.c <SDK Root>\hal\atsamd20\src\halStartup.c
ATSAMR21G18 (via ATSAMD21J18)	<SDK Root>\hal\atsamd21\src\hal.c <SDK Root>\hal\atsamd21\src\halPhy.c <SDK Root>\hal\atsamd21\src\halTimer.c <SDK Root>\hal\atsamd21\src\halStartup.c
ATmega128RFA1	<SDK Root>\hal\atmega128rfa1\src\hal.c <SDK Root>\hal\atmega128rfa1\src\halTimer.c <SDK Root>\phy\atmegarfa1\src\phy.c
ATmega256RFR2	<SDK Root>\hal\atmega256rfr2\src\hal.c <SDK Root>\hal\atmega256rfr2\src\halTimer.c <SDK Root>\phy\atmegarfr2\src\phy.c
AT86RF230B	<SDK Root>\phy\at86rf230\src\phy.c
AT86RF231	<SDK Root>\phy\at86rf231\src\phy.c
AT86RF212	<SDK Root>\phy\at86rf212\src\phy.c
AT86RF233 (including ATSAMR21G18)	<SDK Root>\phy\at86rf233\src\phy.c

Table 5-2. Required Include Paths

MCU / RF transceiver	Include Paths
All	<SDK Root>\nwk\inc <SDK Root>\sys\inc <Application Root> (required to locate *config.h* file)
ATmega1281	<SDK Root>\hal\atmega1281\inc
ATxmega128B1	<SDK Root>\hal\atxmega128b1\inc
ATxmega256A3U	<SDK Root>\hal\atxmega256a3u\inc
ATSAMD20J18	<SDK Root>\hal\atsamd20\inc
ATSAMR21G18 (via ATSAMD21J18)	<SDK Root>\hal\atsamd21\inc
ATmega128RFA1	<SDK Root>\hal\atmega128rfa1\inc <SDK Root>\phy\atmegarfa1\inc
ATmega256RFR2	<SDK Root>\hal\atmega256rfr2\inc <SDK Root>\phy\atmegarfr2\inc
AT86RF230B	<SDK Root>\phy\at86rf230\inc
AT86RF231	<SDK Root>\phy\at86rf231\inc
AT86RF212	<SDK Root>\phy\at86rf212\inc
AT86RF233 (including ATSAMR21G18)	<SDK Root>\phy\at86rf233\inc

Table 5-3. Required Definitions

MCU / RF transceiver	Definitions
All	F_CPU=<MCU Operating Frequency> Note that if MCU frequency is different from the supported by default then you may need to change frequency depended code in the <*SDK Root*>*\hal* directory.
ATmega1281	HAL_ATMEGA1281
ATxmega128B1	HAL_ATXMEGA128B1
ATxmega256A3U	HAL_ATXMEGA256A3U
ATSAMD20J18	HAL_ATSAMD20J18
ATSAMR21G18 (via ATSAMD21J18)	HAL_ATSAMD21J18
ATmega128RFA1	PHY_ATMEGARFA1 HAL_ATMEGA128RFA1
ATmega256RFR2	PHY_ATMEGARFR2 HAL_ATMEGA256RFR2
AT86RF230B	PHY_AT86RF230
AT86RF231	PHY_AT86RF231
AT86RF212	PHY_AT86RF212
AT86RF233 (including ATSAMR21G18)	PHY_AT86RF233

The execution environment should ensure that:

- *SYS_Init()* function is called before any other Lightweight Mesh API call
- *SYS_TaskHandler()* function is called as often as possible
- *SYS_TaskHandler()* function is only called from the main *while (1)* {} loop

Note: The *HAL_Init()* function (called from *SYS_Init()* function) will perform low level hardware initialization. If such initialization already is performed by the existing project environment, then it should be removed from the *HAL_Init()* function.

6. References

[1] Atmel AVR2130: Lightweight Mesh Developer Guide.
[2] Atmel AVR2052: Atmel BitCloud Quick Start Guide.
[3] Atmel AVR2044: RCB128RFA1 – Hardware User Manual.
[4] IAR Embedded Workbench for Atmel AVR.
[5] Studio Archive (AVR Studio installer downloads).
[6] Atmel Studio 6.

7. Revision History

Doc. Rev.	Date	Comments
42029G	03/2014	Added ATSAMR21-XPRO information
42029F	03/2014	Added ATZB-X0-256-3-0-C and ATREB233-XPRO information Removed OTA information
42029E	08/2013	Added ATSAMD20 information Changed product line abbreviation from AVR to Wireless in the document footer
42029D	05/2013	Updated directory structure, removed precompiled binaries information
42029C	03/2013	ATmega256RFR2-XPLD has been replaced by ATmega256RFR2-XPRO
42029B	02/2013	Added ATmega256RFR2 information
42029A	09/2012	Initial document release

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370