FTF FREESCALE TECHNOLOGY FORUM 2015

MCR20A 802.15.4 2.4GHz Standalone Transceiver Integration with Kinetis K or L Series MCUs

FTF-SNT-F1227

Tudor Stănescu | Software Development Manager

J U N E . 2 0 1 5

freescale™

External Use

Session Introduction

This hands-on session aims to teach how to use the Freescale MCR20A IEEE® 802.15.4 wireless transceiver with Freescale software and development boards.

- This presentation is relevant to all who want to develop IEEE® 802.15.4-based connectivity and Internet of Things (IoT)-enabling applications or add connectivity capabilities to existing designs:
  - This presentation is important because it shows how to use a readily-available, easy to use development environment for IoT.
  - The attendees will be become familiar with Freescale 802.15.4 technology and will have a great starting point in their application and product development.
  - One major problem addressed in this presentation is porting/migrating 802.15.4 connectivity capabilities from one MCU to another.
- Presenter: Tudor Stănescu – manager of the IEEE® 802.15.4 and Bluetooth® software team.
- Time allocated to this presentation is 2 hours, out of which 15 minutes will be allotted for questions. Questions will also be taken during the session itself.

Session Objectives

- After completing this session you will be able to:
  - Understand how the MCR20A Freescale offering can benefit your product design by enhancing it with connectivity capabilities and provide you a competitive edge in the IoT space.
  - Configure the various MCR20A-based evaluation board setups.
  - Install and configure the MCR20A IEEE® 802.15.4 software package on top of Kinetis SDK.
  - Understand the IEEE 802.15.4 software architecture and functionality.
  - Appreciate the ease with which an IEEE® 802.15.4-based IoT application or stack can be migrated from one MCU to another, using the MCR20A wireless transceiver.

Agenda

- Hardware Introduction – 20 minutes
  - MCR20A Silicon Overview
  - MCR20A Development Board Overview
  - KL46Z, KL26Z and K64F FRDM Boards Overview
- Software Introduction – 40 minutes
  - Kinetis SDK as Connectivity Foundation
  - IEEE® 802.15.4 MAC/PHY Software and Applications
- Porting the MCR20A IEEE® 802.15.4 Software to FRDM-KL26Z (Hands-on Session with Examples) – 45 minutes
  - Clock Configurations
  - Transceiver Driver and SPI Interfacing
  - Serial Connectivity
  - GPIOs
  - Linker configurations
- Conclusions and Q&A – 15 minutes

Hands-On Session Infrastructure

- The following software and hardware is provided for this session:
  - 18 x training laptops
  - Development Boards:
    - 36 x FRDM-CR20A shields
    - 18 x FRDM-K64F
    - 5 x FRDM-KL46Z
    - 18 x FRDM-KL26Z
  - Kinetis SDK v1.1.0 installation
  - Kinetis SDK v1.1.0 SA for KL26Z and KW01Z
  - MCR20A IEEE® 802.15.4 Software Package
  - IAR Embedded Workbench v7.40.1 installation
  - Scooter Software Beyond Compare
  - PuTTY terminal by Simon Tatham

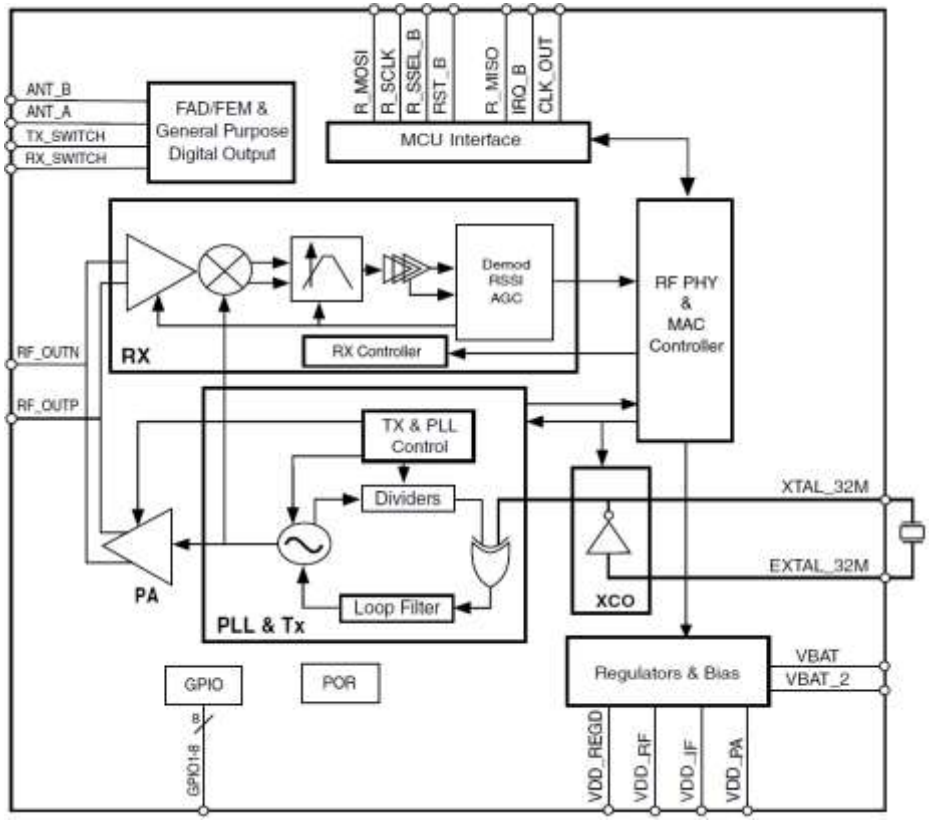
Hardware Introduction

Hardware Introduction - Summary

This section outlines the most important features and competitive advantages of the MCR20A wireless transceiver and the applicable evaluation boards.

- **MCR20AVHM Silicon**
  Main Features and Differentiators
- **FRDM-CR20A Arduino™ Shield**
  Features and Configuration Options
- **MCU FRDM Boards**
  Comparative Presentation of FRDM-KL46Z, FRDM-K64F and FRDM-KL26Z

MCR20A – Silicon Overview



Features:

- High-performance 2.4 GHz IEEE 802.15.4 RF transceiver
- **-102 dBm** sensitivity **+8 dBm** maximum output power reducing the need for external power amplifiers
- Low-power preamble search receive mode (LPPS)
- TX 17 mA @ 0 dBm and RX 19 mA typical
- TX 18 mA @ 0 dBm and RX 19.5 mA max
- Dual PAN support
- Support for MBAN frequencies (2.36-2.48GHz)
- Supports single ended and fast diversity antenna options: single 50 ohm antenna uses single balun to reduce component count and cost
- Packet processor for hardware acceleration PHY/MAC support
- Supports clock output to host microcontroller of either 32 kHz or 4 MHz
- SPI Interface
- 8 GPIOs (+4 outputs via FAD manual control)
- Package: 5x5 32-pin LGA
- Operating parameters: -40 °C to 105°C, 1.8 to 3.6 V

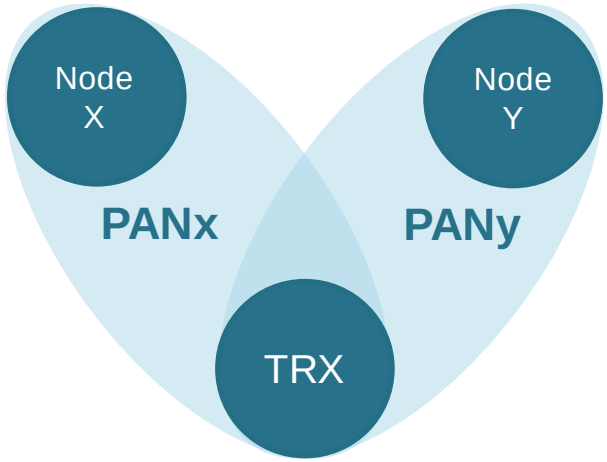
MCR20A Differentiating Features

- **Dual PAN operation:**
  - Duplicate registers for PAN ID, short address, channel, long address and coordinator capability to allow the network node using MCR20A to participate in two 802.15.4 personal area networks
- **Active promiscuous mode:**
  - Promiscuous mode with ACK for packets that are addressed to the node; allows implementation of smart sniffers, that are part of the network and analyze the protocol at the same time
- **Fast Antenna Diversity (FAD):**
  - Control of an external antenna switch that will select between two antennas based on RSSI
- **Low Power Preamble Search (LPPS):**
  - The demodulator and other sub-sections of the receiver can be independently duty cycled at 50%. **1.5 dB** sensitivity traded off for **4 mA** current saving

Dual-PAN Operation

- Allows a single 802.15.4 radio to participate in **2 different PANs** simultaneously, with **2 different stacks** as well

- Maintains **2 sets of network parameters** for each PAN :
  - Channel0/1 – PAN Channel
  - MacPanID0/1 – PAN ID
  - MacShortAddrs0/1 – Device short address
  - MacLongAddrs0/1 – Device extended address
  - PANCORDNTR0/1 – PAN Coordinator capability



- The transition from one PAN to the other one can be **manual** (under software control) or **automatic**
- Automatic transition is done using a programmable timer with a **PAN Dwell Time** from 0.5ms to 3.2s
- If both PAN are defined on the same channel, TRX is able to process both PAN simultaneously (no PAN Dwell Time to define or channel switch time to consider)
- If PANs are on different channels, channel switch is **56 µs** Time to switch, poll, receive packet and switch back is **<10mS**

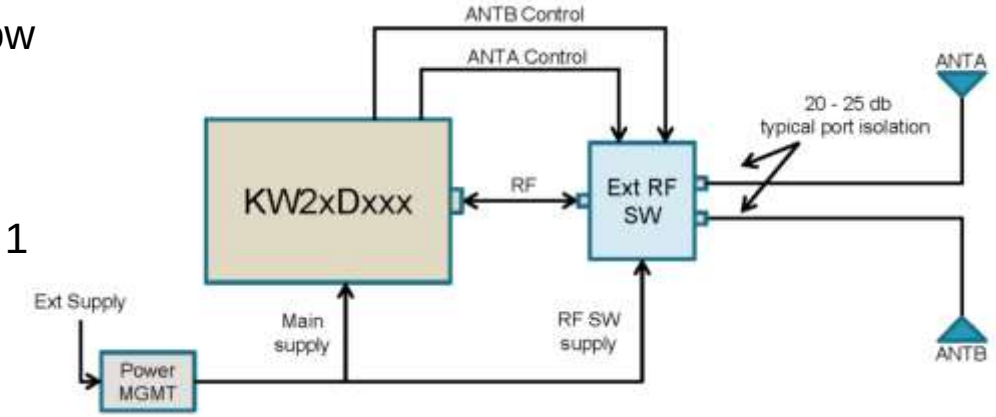
Active Promiscuous Mode

- Works like a typical Promiscuous Mode, except:
- Device will **automatically transmit an ACK** packet for frames that are addressed to it.
- Allows for building **smart sniffers** that can be used as commissioners in various networks
- The active promiscuous device can **monitor** the network **and configure** it at the same time
- The active promiscuous device can operate as a node in the network and as a sniffer at the same time

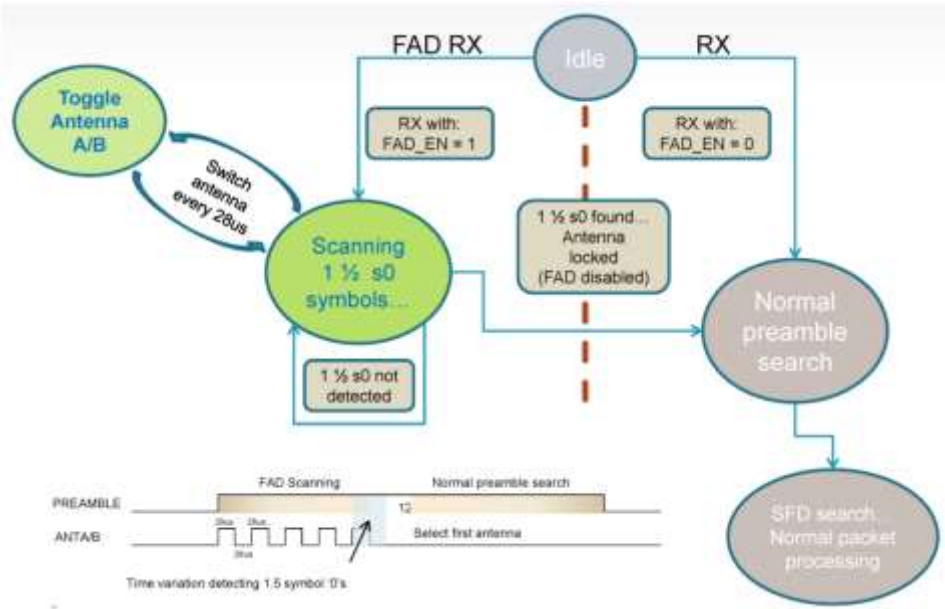
Fast Antenna Diversity

- Sensitivity performance increases the low end limit of the dynamic range
- Little cost of enabling FAD
- No software overhead
- Operates on preamble, switch on every 1 + ½ symbols

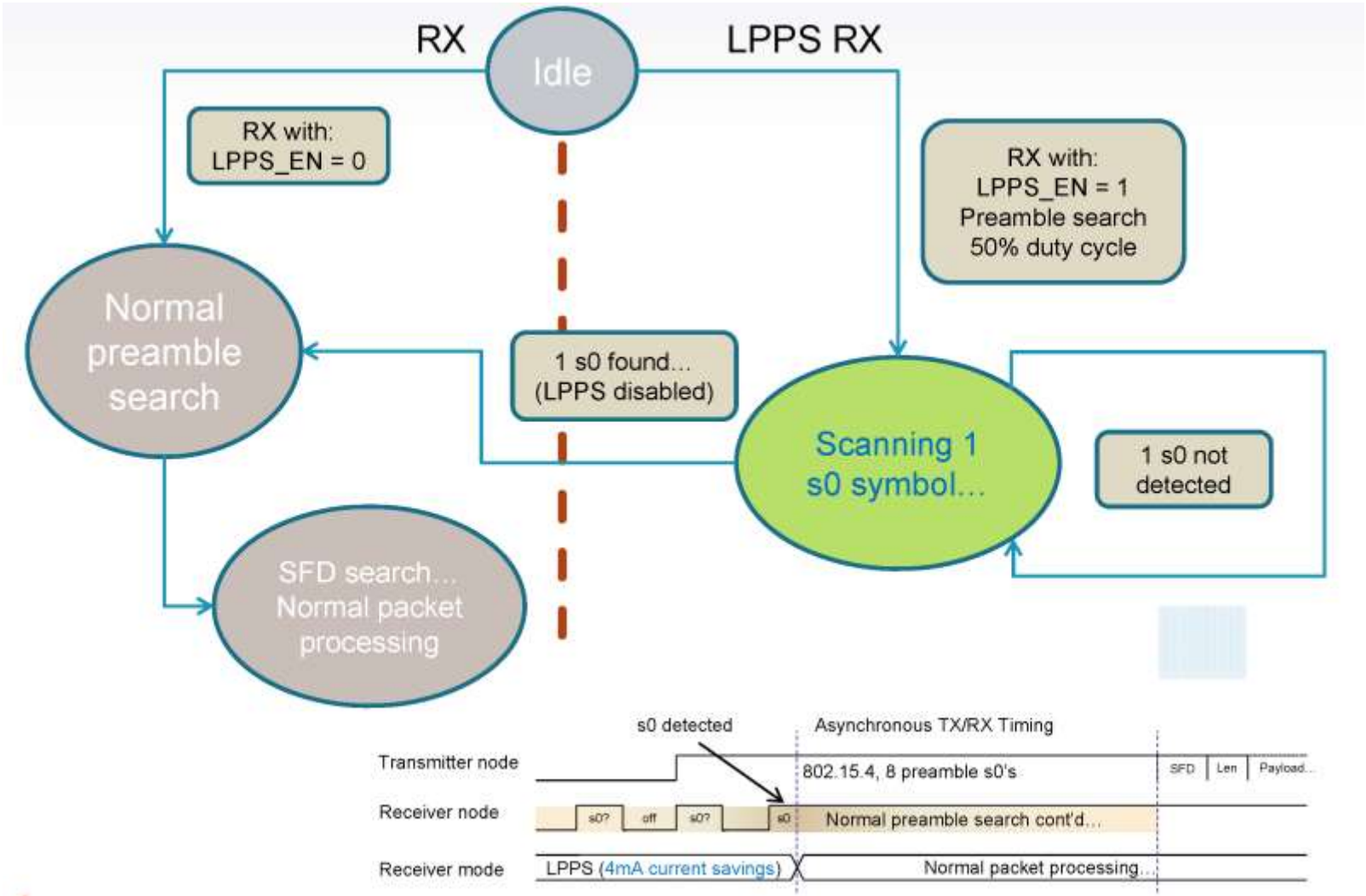
FAD External Circuitry



FAD Flowchart



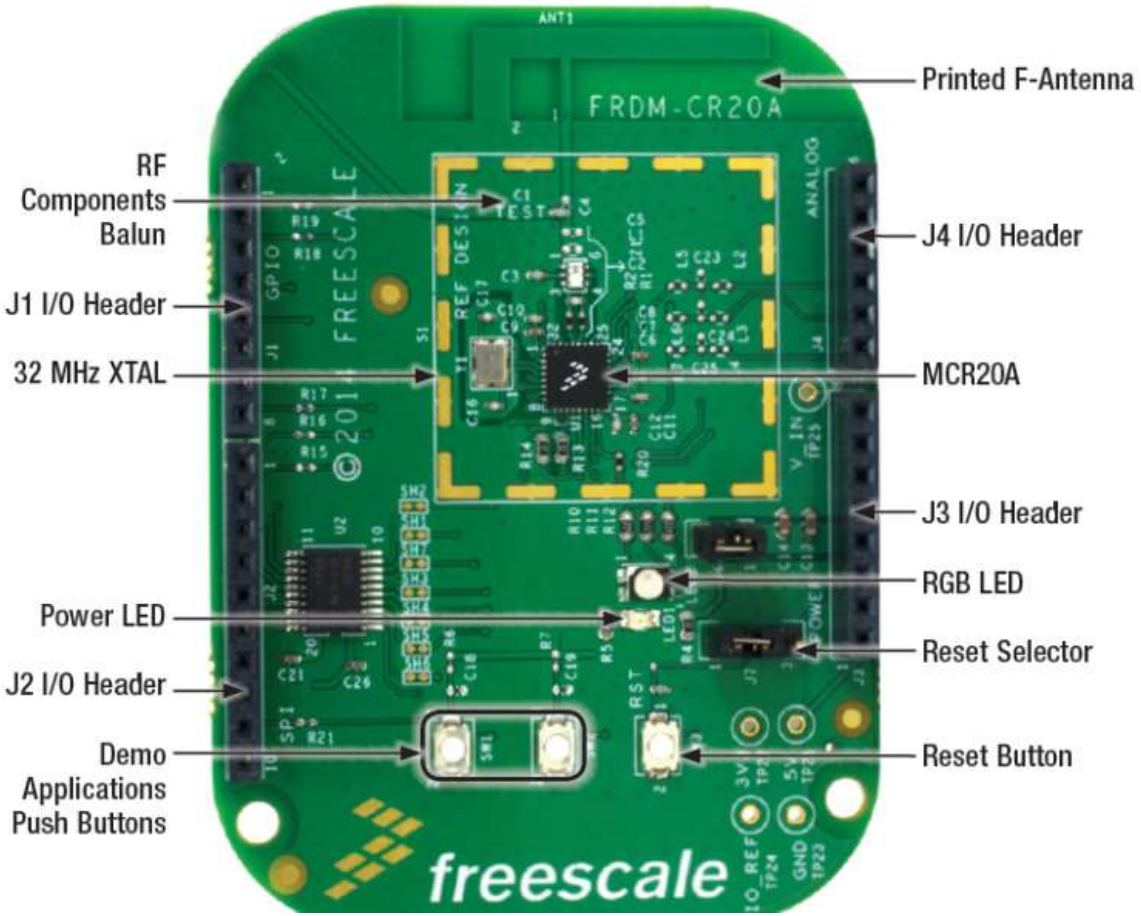
Low Power Preamble Search



Development Hardware

FRDM/ArduinoTM Form Factors for MCR20A and MCUs

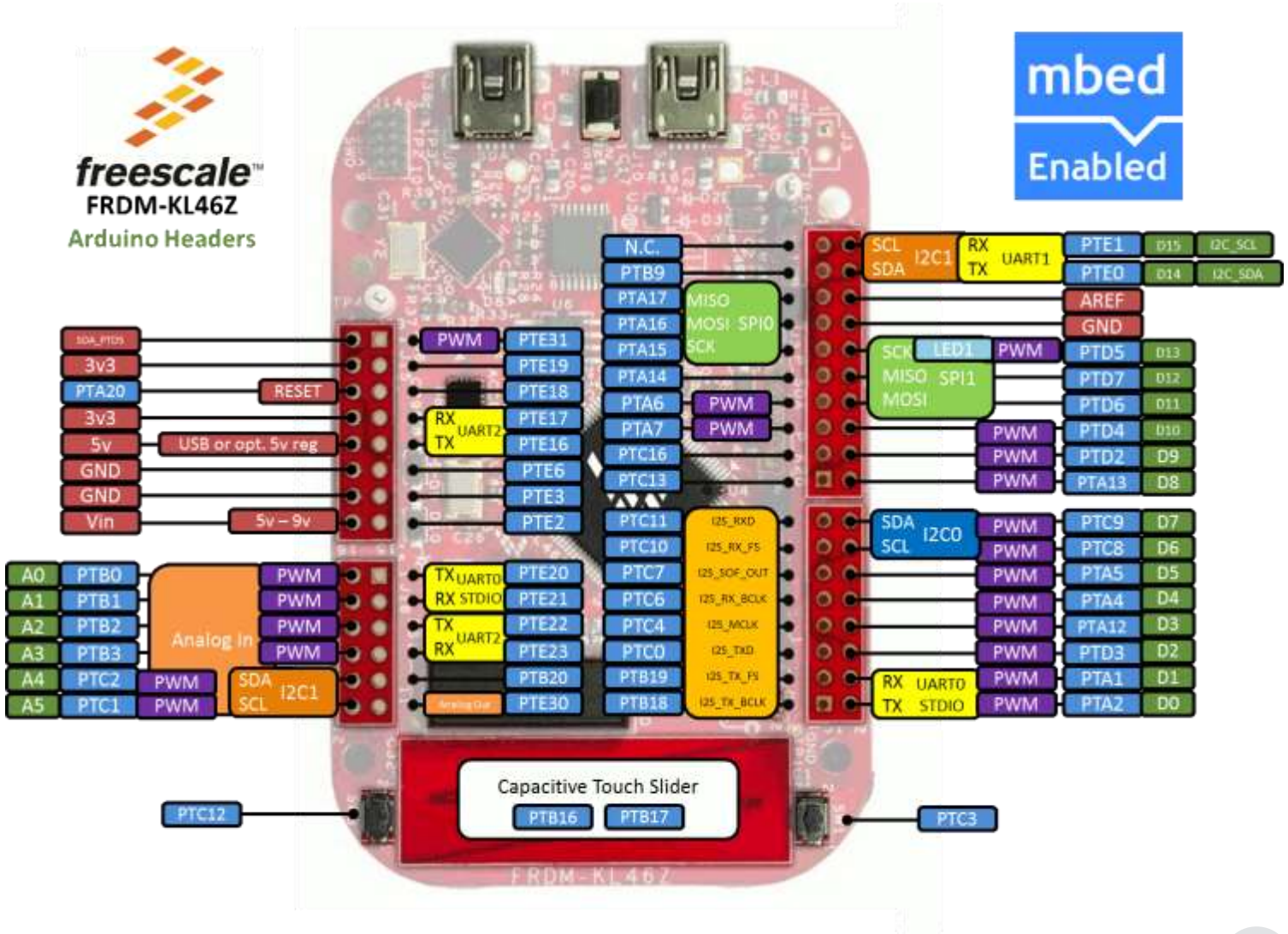
FRDM-CR20A - Overview



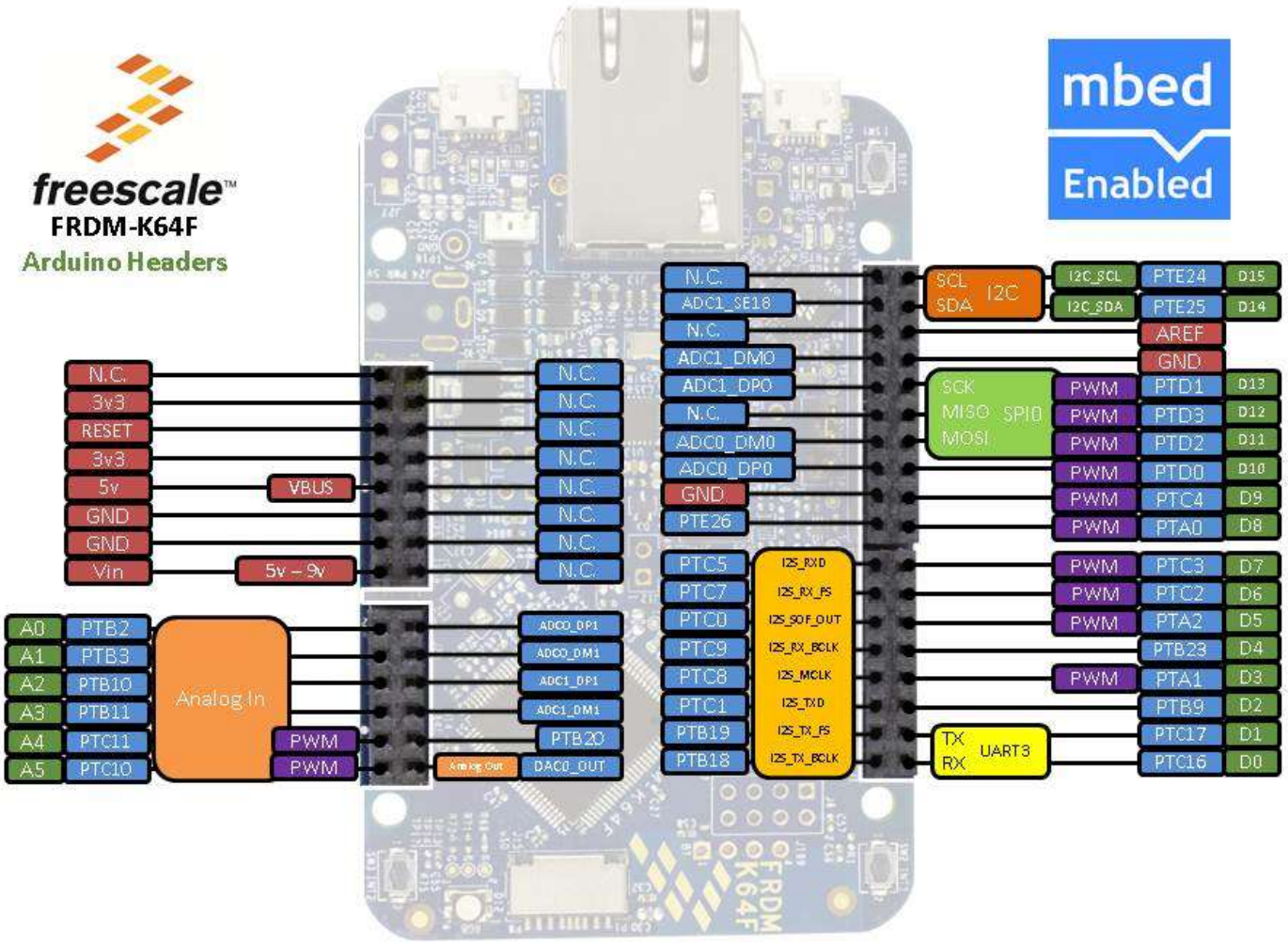
Features:

- MCR20AVHM 802.15.4 Transceiver – 2.36-2.48GHz frequencies
- SPI interface
- Two (2) user push-button switches for interrupts (SW2/SW3) driven by external MCU
- One (1) RGB LED indicator driven by external MCU
- PCB inverted F-type antenna and SMA RF port available
- Minimum number of matching components and external balun
- Form factor compatible with Arduino™ R3 pin layout
- Standard FRDM daughter card mounting interface (Shield)
- Can be directly connected to the FRDM-K64F, FRDM-KL46 and FRDM-KL26Z

FRDM-KL46Z Arduino Headers



FRDM-K64F Arduino Headers

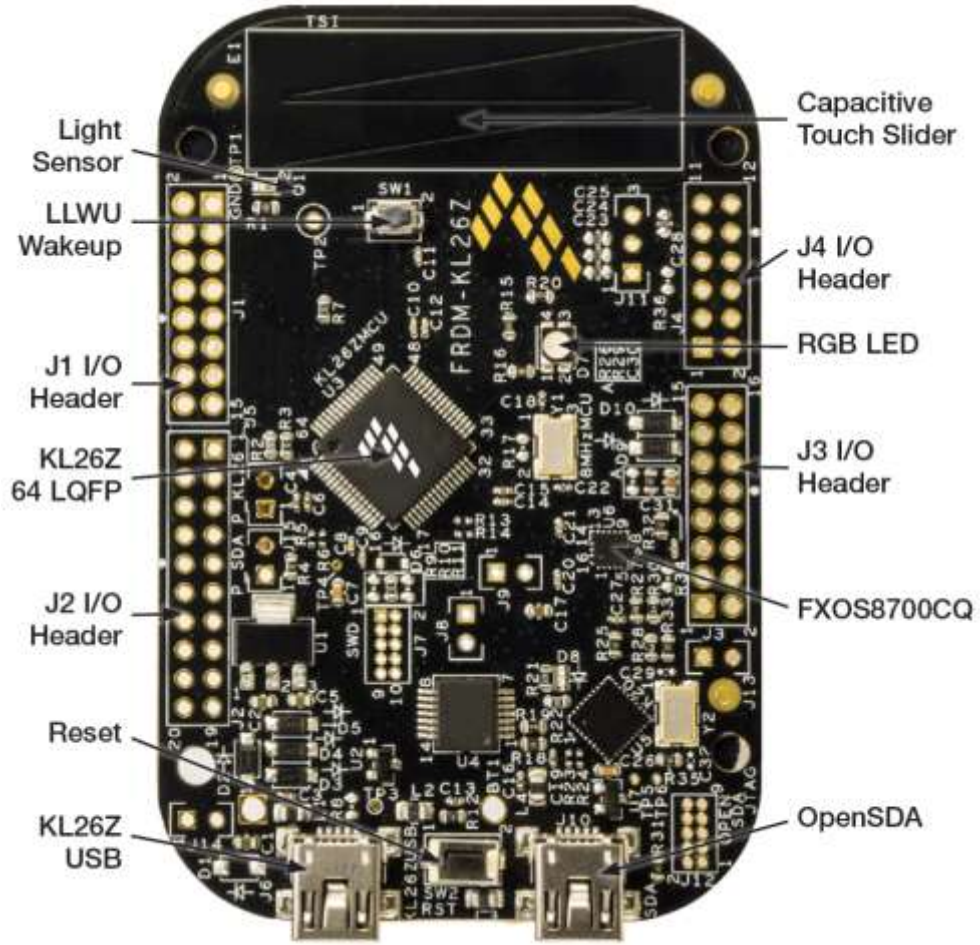


Supported Network Nodes in the Software

FRDM-CR20A + FRDM-K46

FRDM-CR20A + FRDM-K64F

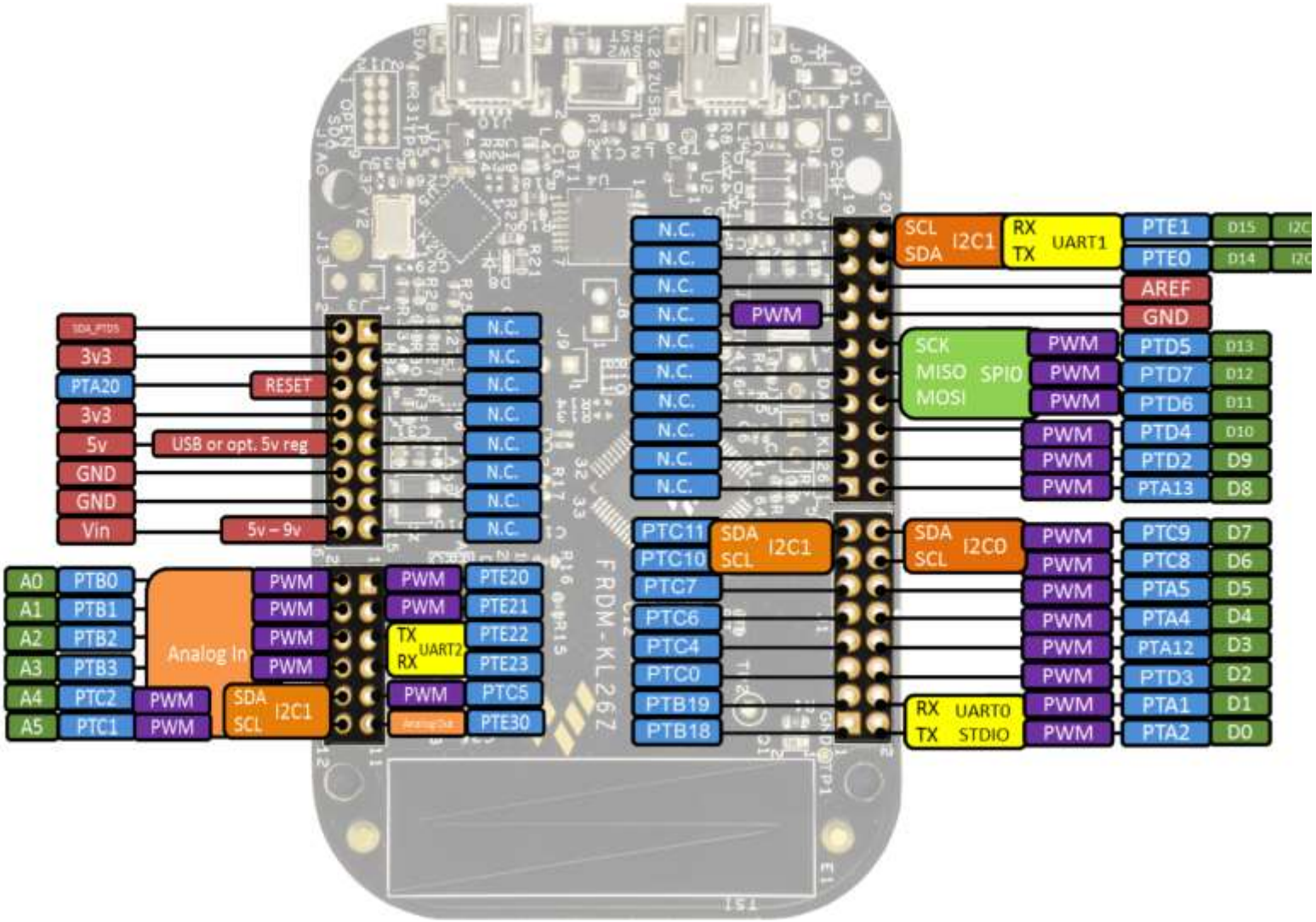
FRDM-KL26Z - Overview



Features:

- MKL26Z128VLH4 MCU – 48MHz, 128KB Flash, 16KB SRAM, USB OTG (FS), 64LQFP
- Capacitive touch slider, FXOS8700CQ accelerometer and magnetometer, Tri-color LED, ambient light sensor
- Flexible power supply options – USB, coin cell battery, external source
- Easy access to MCU I/O
- Battery-ready, power-measurement access points
- Form factor compatible with Arduino™ R3 pin layout
- OpenSDA debug interface
  - Mass storage device flash programming interface (default) – no tool installation required to evaluate demo apps
  - P&E Debug interface provides run-control debugging and compatibility with IDE tools
  - CMSIS-DAP interface: ARM standard for embedded debug interface
  - More information at www.freescale.com/opensda

FRDM-KL26Z Arduino Headers



Newly Supported Network Node After Porting

FRDM-CR20A + FRDM-KL26Z

Software Introduction

Software Introduction - Summary

This section outlines the building blocks of the software enablement for the MCR20A wireless transceiver, running on Kinetis microcontrollers and serving as the essential foundation for IoT applications.

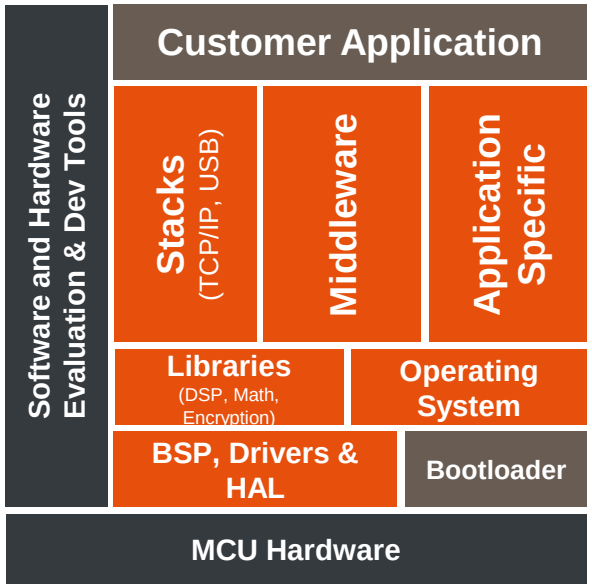
Kinetis SDK
Foundation for Connectivity Solutions

Freescale IEEE® 802.15.4 MAC and PHY
Standard, Software Architecture, Features and Applications

Kinetis Software Development Kit (SDK)

The software framework and reference for Kinetis MCU application development

Hardware abstraction, peripheral drivers, stacks, RTOS's, utilities, and usage examples; delivered in C source



Open Source Initiative

CMSIS COMPLIANT

CERTIFIED USB

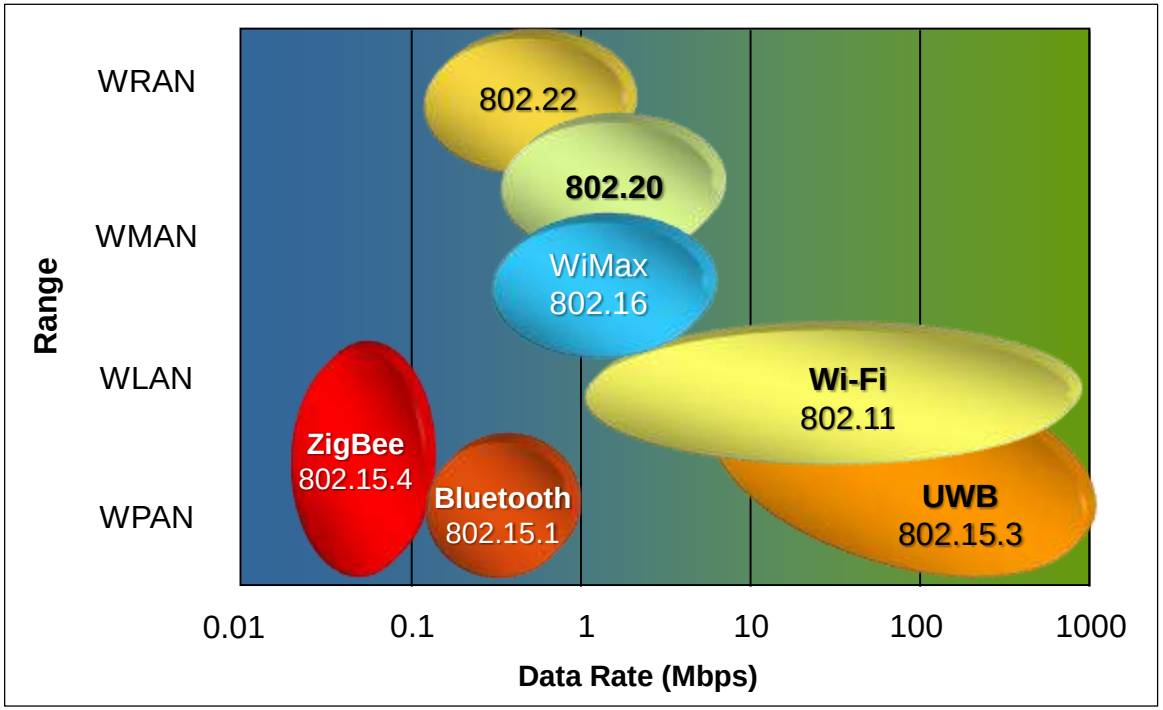
Product Features

- Open source **hardware abstraction layer (HAL)** provides APIs for all Kinetis hardware resources
- BSD-licensed set of **peripheral drivers** with easy-to-use C-language APIs
- Comprehensive HAL and driver **usage examples** and **sample applications** for RTOS and bare-metal
- C**MSIS-CORE** compatible startup plus **CMSIS-DSP** library and examples
- RTOS Abstraction Layer (OSA) with support for **FreeRTOS**, Freescale **MQX**, Micrium **uC/OS**, and **bare-metal**
- Integrates new Freescale unified **USB stack**, open source **FAT file system**, **encryption math/DSP libraries**, and open source **TCP/IP stack** (lwIP).
- Support for **multiple toolchains**: IAR Embedded Workbench, Kinetis Design Studio, Keil uVision and Atollic TrueStudio.

Learn more at: www.freescale.com/KSDK

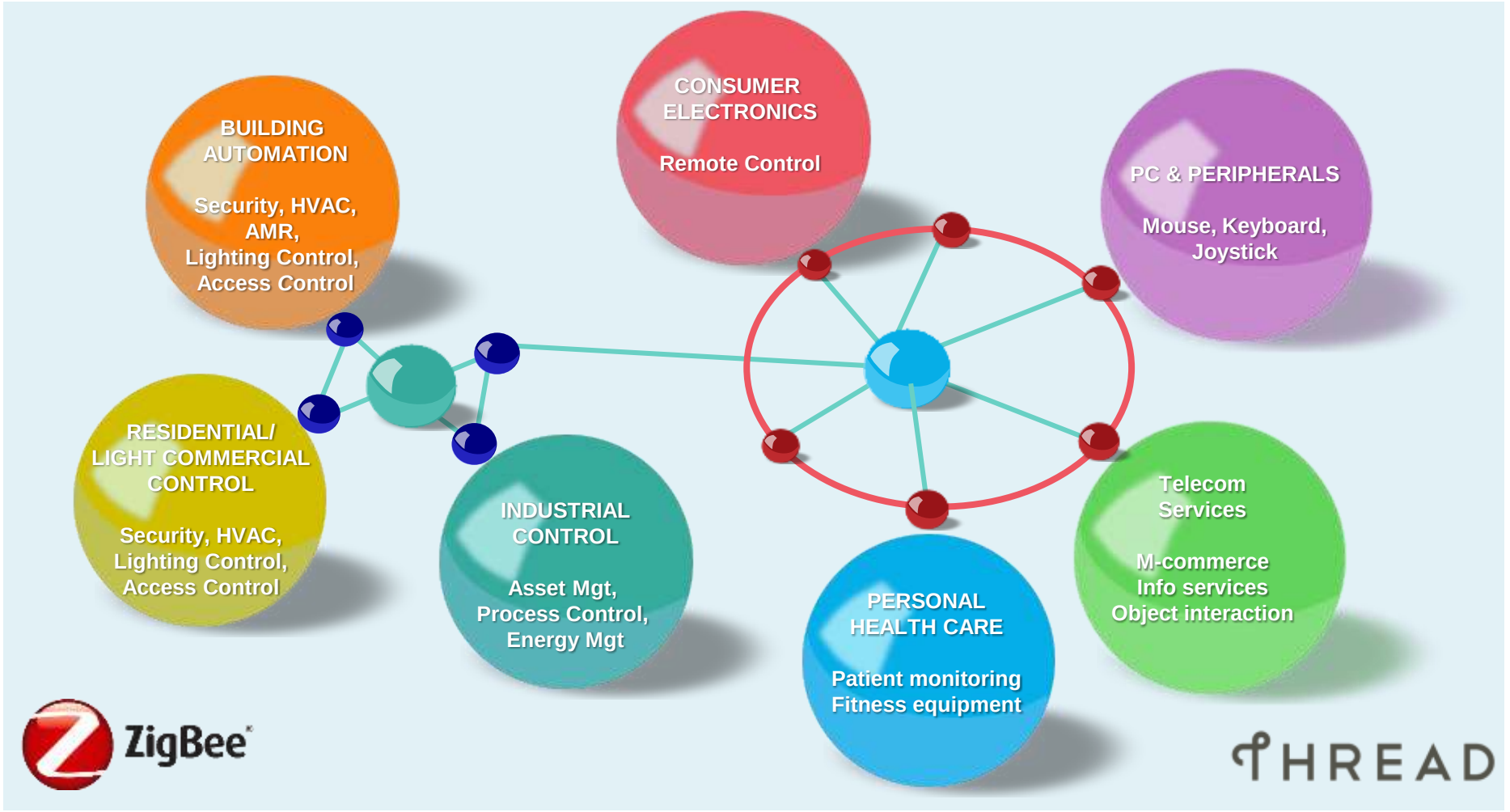
The IEEE® 802.15.4 Standard - Overview

- Stands out in the 802.x wireless space through: low power, low duty cycle, short frame format, low latency and generally low cost/complexity of implementation
- Allows self-forming, self-healing mesh networks
- Multiple RF bands: sub-GHz, 2.4 GHz
- Multiple modulation techniques: FSK, O-QPSK, OFDM



The IEEE® 802.x Wireless Space

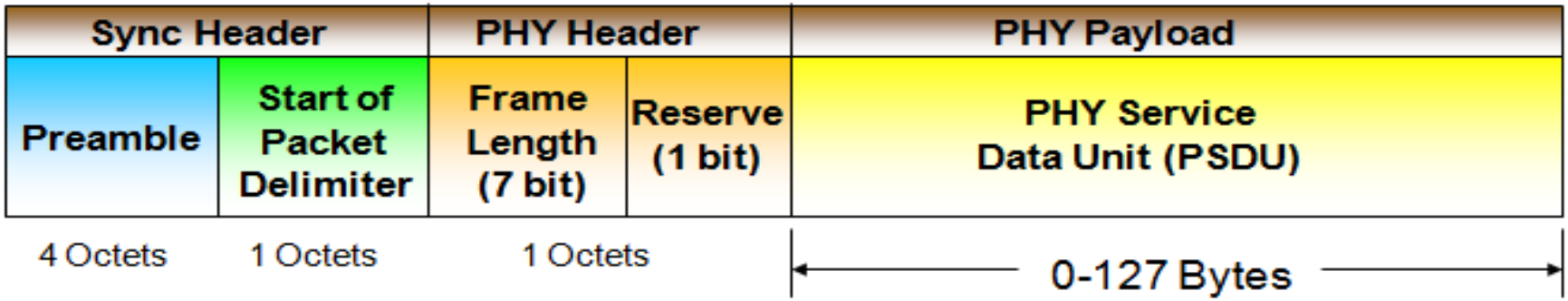
IEEE® 802.15.4-enabled Markets and Applications



The IEEE® 802.15.4 Standard – 2.4 GHz PHY Layer

- Low rate, low power, low cost O-QPSK DSSS modulation
- 16 Channels in the 2.4 GHz ISM band spaced at 5 MHz
- 250 kb/s (4 bits/symbol)
- 127 bytes PHY protocol data unit payload
- Tx power of at least 1 mW (0 dBm)
- Tx center frequency tolerance +/- 40 ppm
- Receiver sensitivity -85 dBm
- Two service access points: data (PD-DATA) and management (PLME)

The IEEE® 802.15.4 PHY Protocol Data Unit



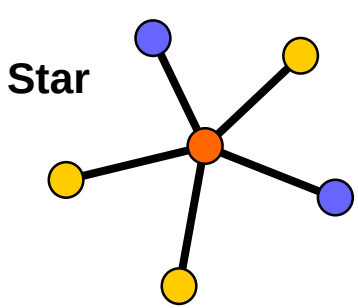
- The 32-bit (8 O-QPSK symbols) all zeros preamble is used for synchronization
- “11100101” SFD indicates start of packet
- 7 out of the 8 PHY header bits are used to indicate the length of the PHY Service Data Unit (PSDU)
- The 2.4 GHz O-QPSK PSDU has a variable length between 0 and 127 bytes

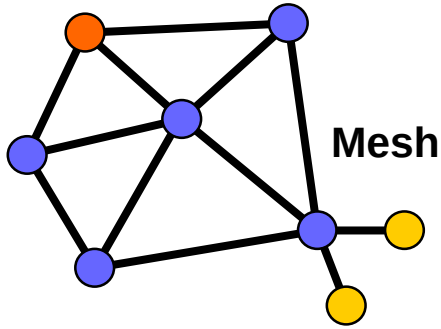
The IEEE® 802.15.4 Standard – MAC Layer

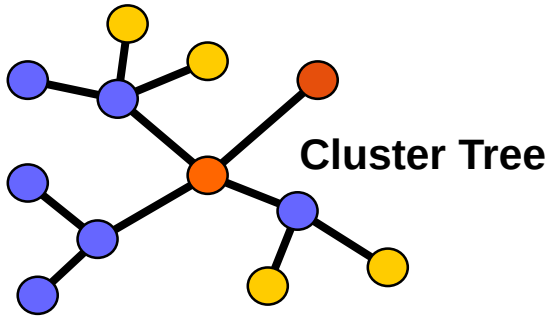
- Ensures reliable and secure data transfers
- Essential foundation for technologies like ZigBee® or Thread
- Collision avoidance algorithm through clear channel assessment
- Acknowledgement-based transmissions and re-transmissions
- Integrity checks with CRC-16
- AES-128 encryption and CCM* block ciphers authentication of data
- Allows star or peer-to-peer topologies
- IEEE standard 64-bit or short, dynamic 16-bit addressing
- Dynamic device addressing allowing routed meshes in upper layers
- Optional slotted mode with superframe-based duty cycles
- Device segregation based on capabilities and roles in a network: coordinator and end device
- Exposes two service access points: management (MLME) and data (MCPS)

IEEE® 802.15.4 MAC Layer-enabled Topologies

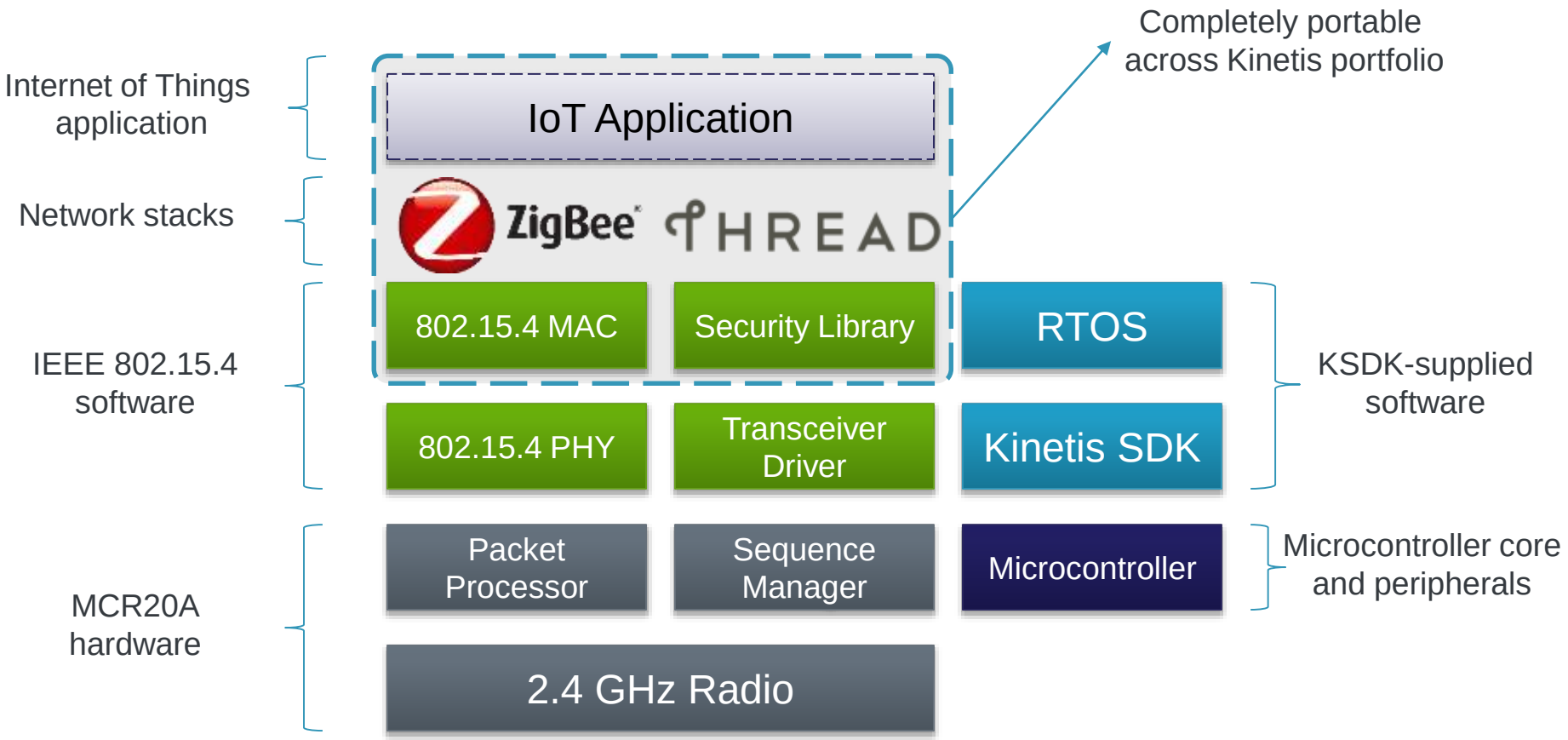
- **PAN Coordinator**
- **Full Function Device (FFD)**
  - •Any topology
  - •Network coordinator capable
  - •Talks to any other device
- **Reduced Function Device (RFD)**
  - •Limited to being leaf devices
  - •Cannot become a network coordinator
  - •Talks only to a network coordinator
  - •Very simple implementation



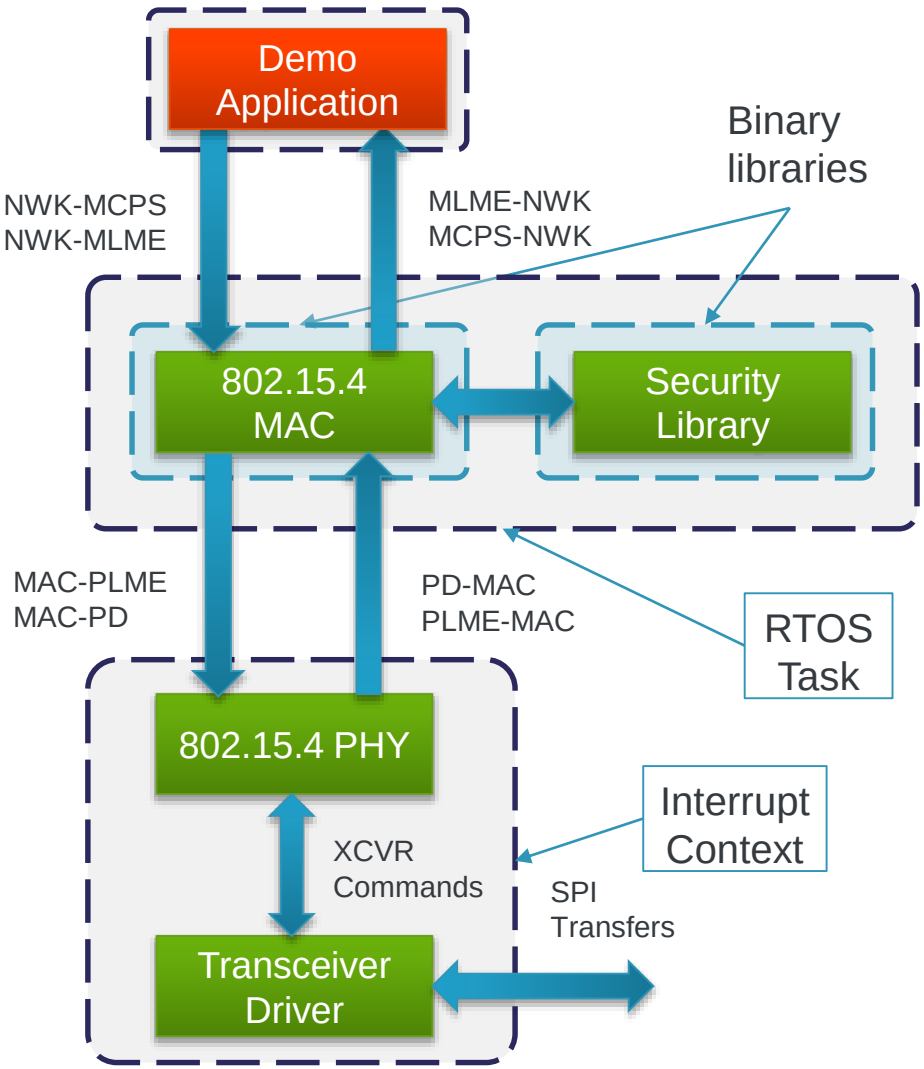




Freescale IEEE® 802.15.4 MCR20A System Architecture



Freescale IEEE® 802.15.4 Software for MCR20A



Features:

- Add-on for the Kinetis SDK
- Uses RTOS services (either FreeRTOS or MQX)
- Needs two contexts: one RTOS task per MAC instance and the interrupt context.
- Contains 4 main modules: MAC library, Security library, PHY module and transceiver driver
- Transceiver driver communicates with any SPI driver of virtually any Kinetis microcontroller connected to the MCR20A.
- The PHY exposes standard service access points to the MAC (PLME and PD-DATA) and can operate in dual PAN mode.
- The MAC and security library are Freescale software IP and are distributed only in binary format.
- The MAC is instantiable and exposes for each instance the standard service access points (MLME and MCPS) for the network (NWK) stack(s) above, which can also be a simple application for demonstration purposes.
- This software is currently used by Freescale for as the basis for the ZigBee and the Thread stacks, which can even coexist on the same microcontroller thanks to the dual PAN feature.

Porting the MCR20A 802.15.4 Software to FRDM-KL26Z

Software Porting - Summary

This section is a hands-on presentation of how to port the MCR20A software to FRDM-KL26Z as an example of the ease of use of the KSDK-Connectivity ecosystem.

- **Premises**
  Leveraging the Software and Hardware Ecosystem for Porting

- **Setting-up the Software Environment**
  Kinetis SDK
  IAR Emebedded Workbench
  Connectivity Package

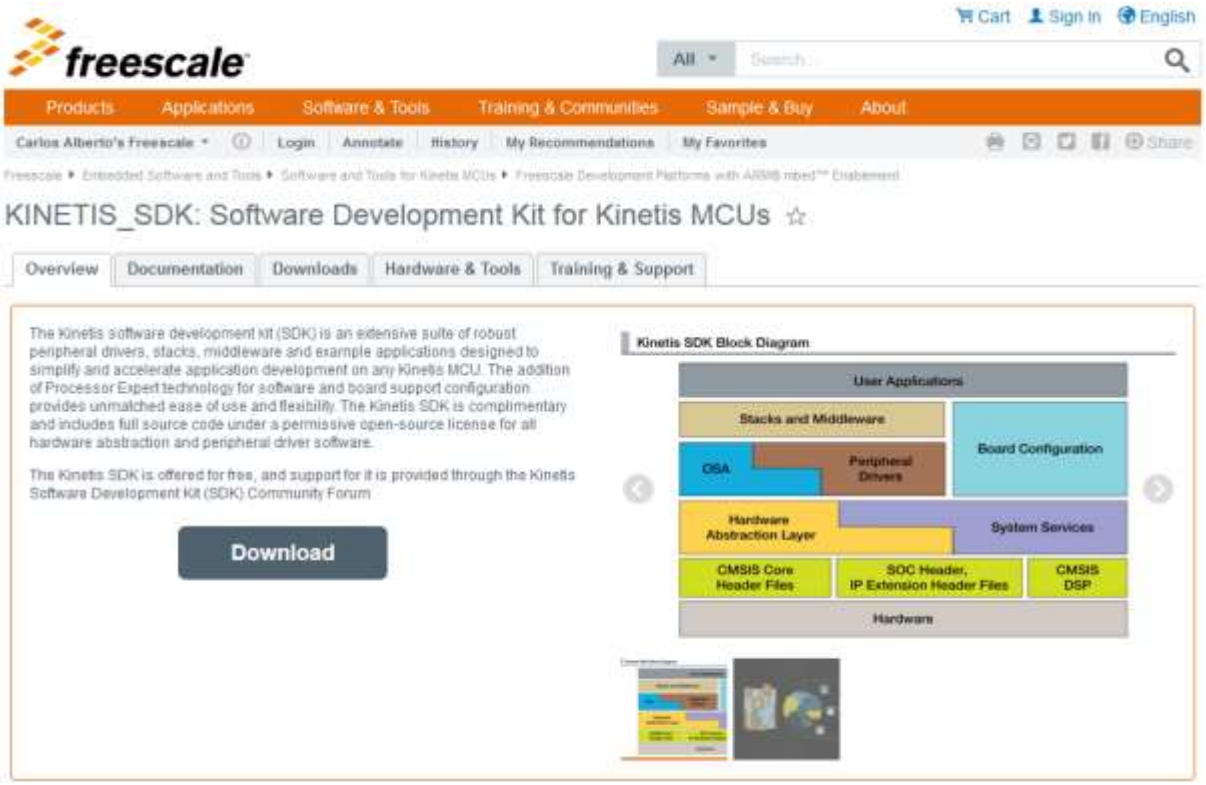
- **Porting**
  Getting Familiar with a Pre-configured MCR20A Application
  Porting a MCR20A Application

Premises – Why it is so Easy to Port

- Kinetis SDK:
  - Ensures availability of Kinetis microcontroller basic enablement: peripheral drivers, clock configurations and linker files.
  - Provides readily configured RTOS libraries and abstraction layers for RTOS services which allow switching from one RTOS to another.
- Arduino standard for FRDM boards:
  - Allows the FRDM-CR20A shield to be paired with virtually any microcontroller board supporting the standard
- Tasks left for the user
  - Pair the FRDM-CR20A shield with the desired MCU motherboard
  - Use the software building blocks to design a connectivity application

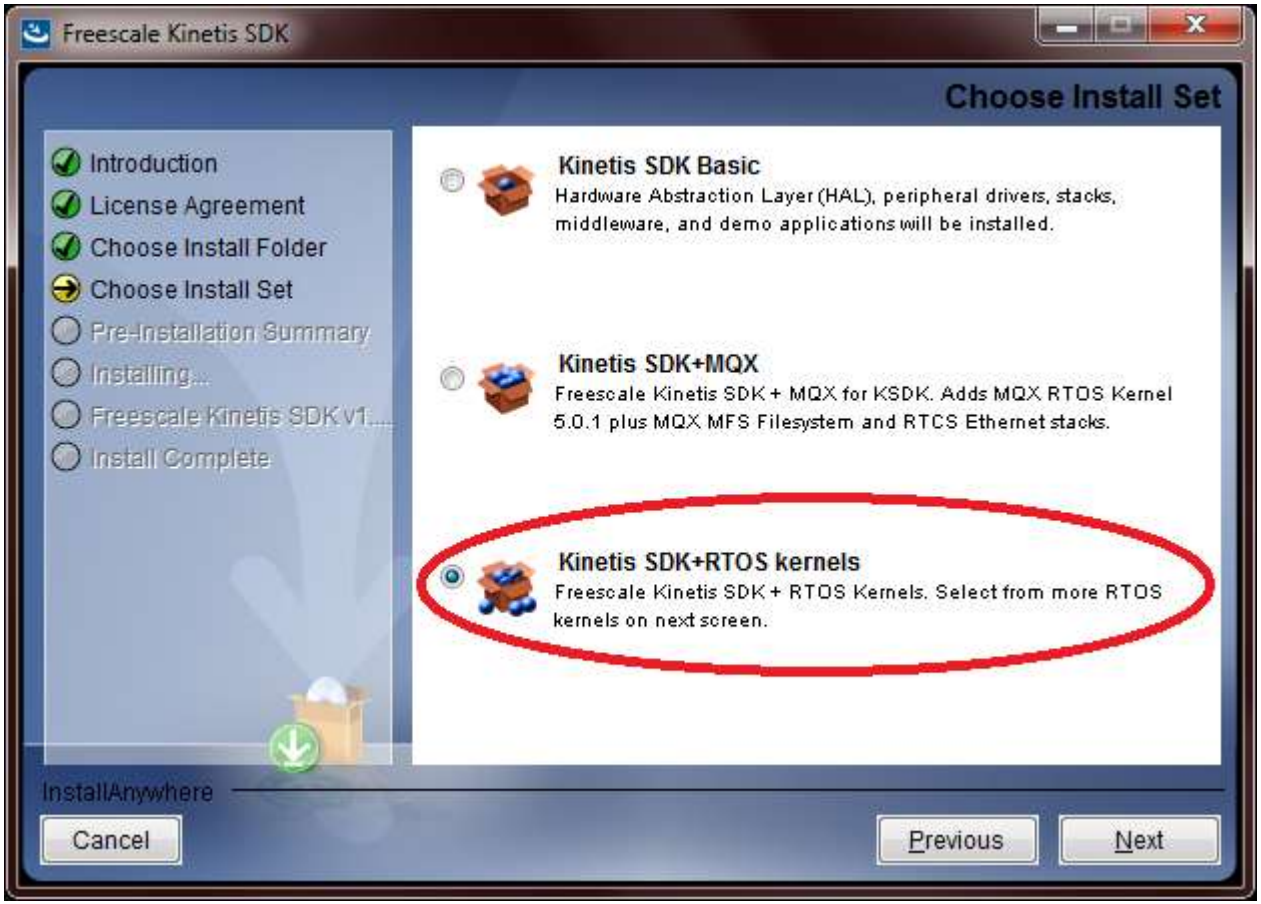
KSDK v1.1 and KSDK v1.1 SA for KL26Z Installation

- Navigate to http://www.freescale.com/ksdk and click “Download”
- Download Kinetis SDK v1.1.0
- Download KSDK v1.1.0 standalone release for the FRDM-KL26Z and MRB-KW01
- Default Install Paths: C:\Freescale\KSDK_1.1.0 and C:\Freescale\KSDK1.1.0_KL26Z_KW01Z_1.0.0



KSDK Installation Package

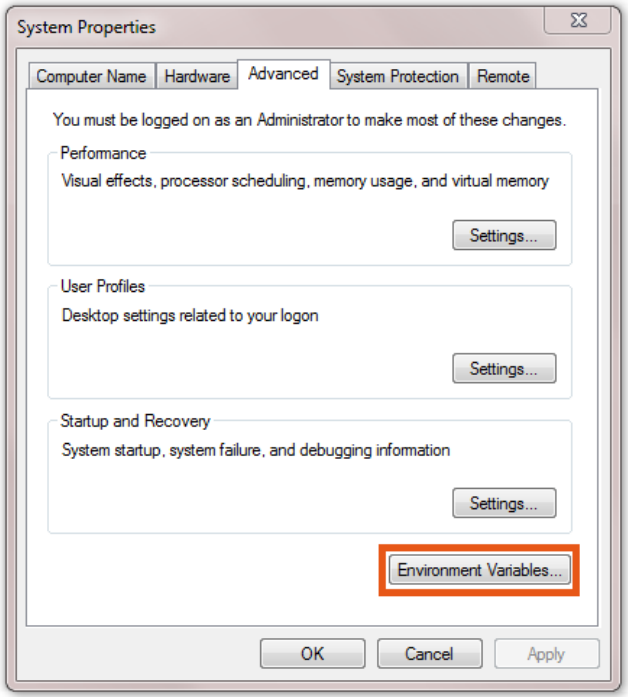
- During installation, select all the RTOSes. This session focuses on FreeRTOS

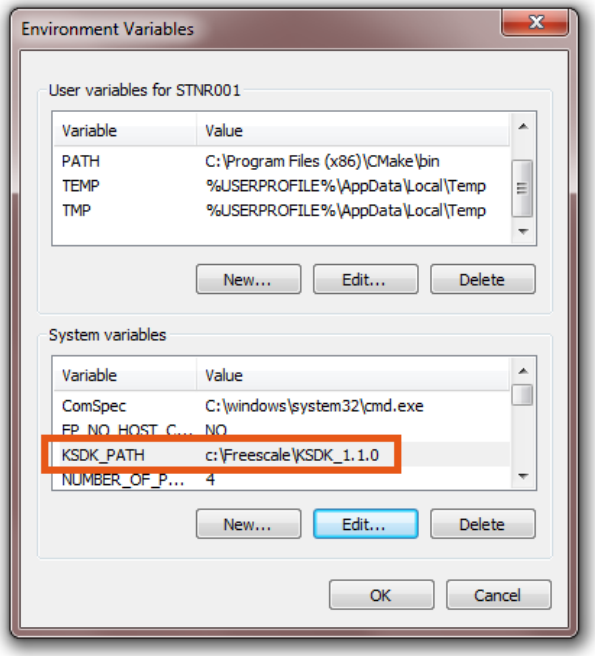


KSDK Environment Variable

- The **KSDK_PATH** system variable is used to point to the desired KSDK installation directory
- Useful in the case when there are multiple versions of KSDK are installed
- Batch files provided on your machine to easily modify this variable
- This variable is used by connectivity projects to refer KSDK files
- Control Panel->System Properties->Advanced Tab

->Environment Variables:





Environment Set-up – IDE/Toolchain

- IAR Embedded Workbench for ARM
  - The connectivity applications and applications are currently limited to the IAR Embedded Workbench for ARM.
  - Your machine comes with version 7.40.1 of this IDE/Toolchain pre-installed.
  - Check your Windows Start Menu for launching it.
  - For more information about this development environment, please visit www.iar.com/ewarm

MCR20A Connectivity Software Package Installation

- MCR20A webpage and software download link: www.freescale.com/MCR20A

freescale

Cart | Sign In | English

All | Search...

Products | Applications | Software & Tools | Training & Communities | Sample & Buy | About

Razvan Tudor's Freescale | Login | Annotate | History | My Recommendations | Subscribe | My Favorites | Share

Freescale ▸ Wireless Connectivity ▸ 2.4 GHz Wireless Solutions

MCR20A: 2.4 GHz 802.15.4 Wireless Transceiver ☆

Overview | Documentation | Software & Tools | Buy / Parametrics | Training & Support

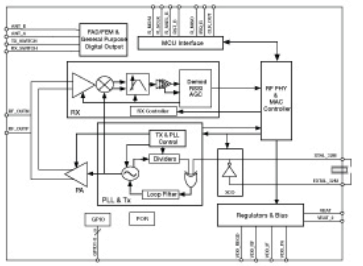
Data Sheet | Buy

The MCR20A expands the Freescale portfolio of wireless connectivity products by delivering a new generation of 2.4 GHz transceiver for the IEEE® 802.15.4 standard. The MCR20A provides a world-class link budget of 110 dB that ensures the longest range of communication. At the same time, the MCR20A is able to receive and transmit at significantly lower peak currents than other competitive devices. This enables mesh networks to run on the same battery for a much longer period. The Dual PAN support allows the system to concurrently participate in two 802.15.4 networks, eliminating the need for multiple radios.

Features

- High-performance 2.4 GHz IEEE 802.15.4 RF transceiver
- 802.15.4 PHY/MAC support
- -102 dBm sensitivity +8 dBm maximum output power reducing the need for external power amplifiers
- Low-power receive mode (LPPS)
- TX 17 mA @ 0 dBm and RX 19 mA typical
- TX 18 mA @ 0 dBm and RX 19.5 mA max
- Dual PAN support
- Supports single-ended and fast diversity antenna options: single 50 ohm antenna uses single balun to reduce component count and cost
- Packet processor for hardware acceleration
- 128-bit random number generator
- 1.8-3.6 V operating range
- Small footprint: 5x5 LGA 32 pin
- –40 °C to +105 °C operational temperature range
- Supports SMAC, Thread Networking Protocol and ZigBee stacks
- Software protocol stacks, tools, and IDE are compatible with the Freescale Kinetis MCU family and integrated in the Kinetis software development kit (SDK)

MCR20AVHM Block Diagram



Featured Documentation

MCR20AVHMFS: MCR20A - Fact Sheet
MCR20RM: MCR20A Reference Manual
MCR20AVHM: MCR20A Data Sheet

Featured Software and Tools

FRDM-CR20A: Freescale Freedom Development Board for MCR20A Wireless Transceiver
MCR20A-IEEE-802-15-4: MCR20A IEEE® 802.15.4 Software v5.0.2

Featured Protocol

Thread Networking Protocol
An ideal way to connect and control products around the home

Read More

The Embedded Beat Blog by Freescale hardware and software experts

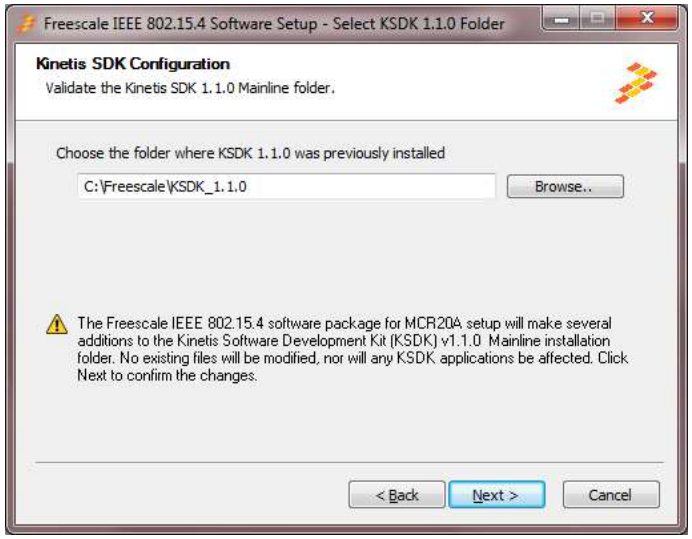
Internet of Tomorrow Blog
Increasing intelligent connections everywhere

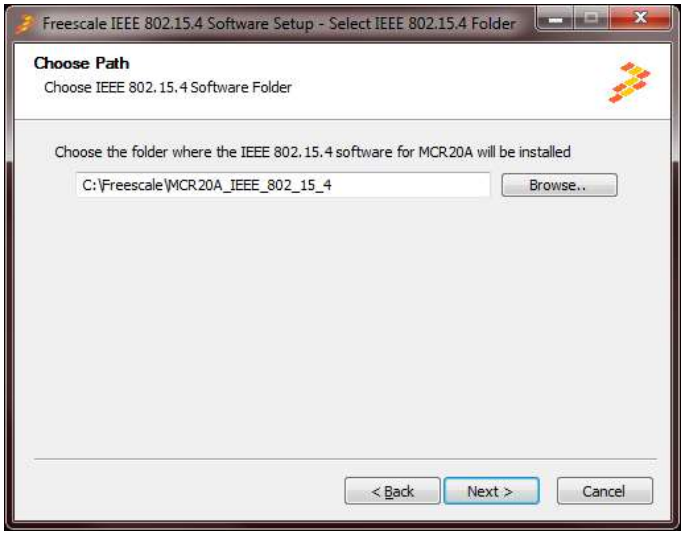
Connect with Us

Wireless Connectivity Community
Freescale Wireless Connectivity solutions forum

Kinetis Software Development Kit (SDK) Community
Get software support and application development expert advice, share ideas and get engaged with the MCU developer community

MCR20A Connectivity Software Package Installation





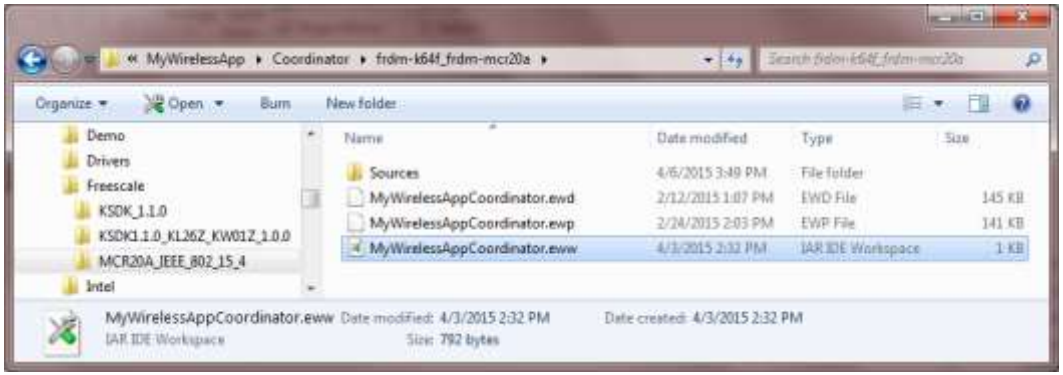
- MCR20A webpage and download link: www.freescale.com/MCR20A
- The package requires KSDK to be previously installed
- Make sure to select the correct KSDK 1.1.0 installation folder when prompted
- Select the default installation folder for the MCR20A software package as: C:\Freescale\MCR20A_IEEE_802_15_4

Getting Familiar with a Pre-configured MCR20A Application

FRDM-K64F “MyWirelessApp” Coordinator

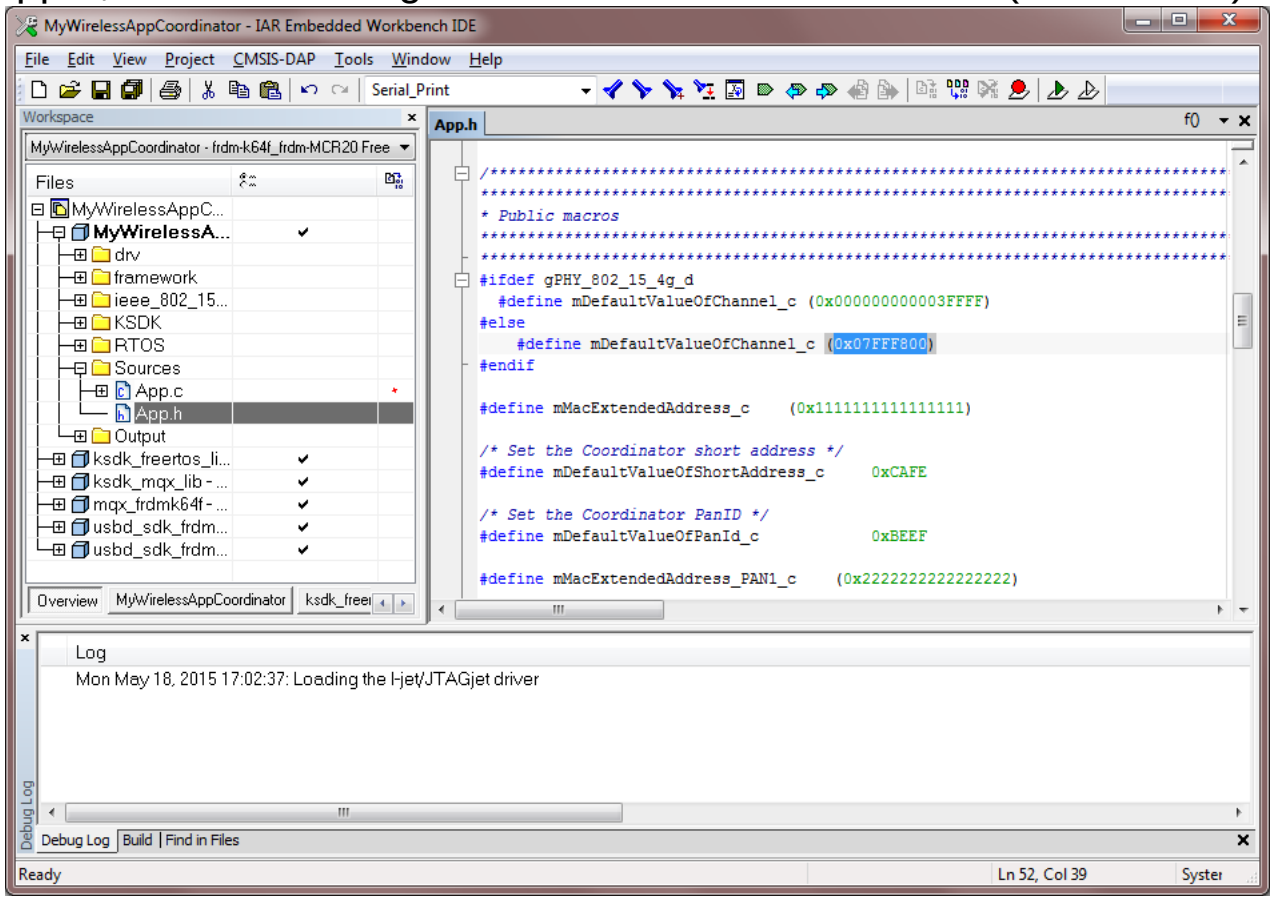
Opening a K64F/MCR20A Application (MyWirelessApp)

- Open the K64F MyWirelessApp Coordinator IAR workspace from C:\Freescale\MCR20A_IEEE_802_15_4\app\ieee_802_15_4\MyWirelessApp\Coordinator\frdm-k64f_frdm-mcr20a\ by double-clicking on the .eww file.



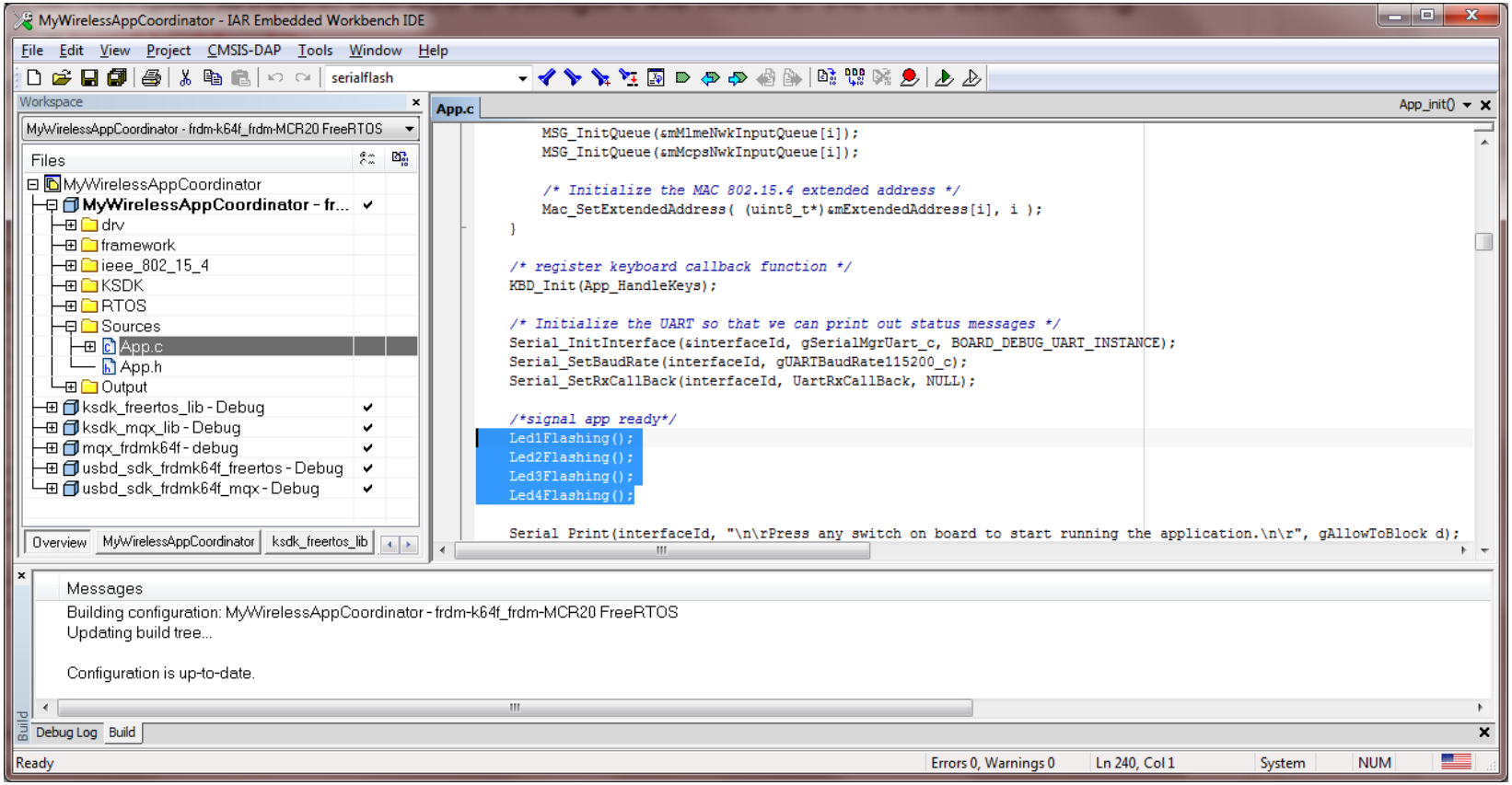
Configuring K64F/MCR20A Application (MyWirelessApp)

- Select the frdm-k64f_frdm-MCR20 FreeRTOS build configuration for the MyWirelessAppCoordinator project from the top drop-down box.
- Navigate to App.h, line 52 to configure the radio channel bit mask (bits 11 - 26).



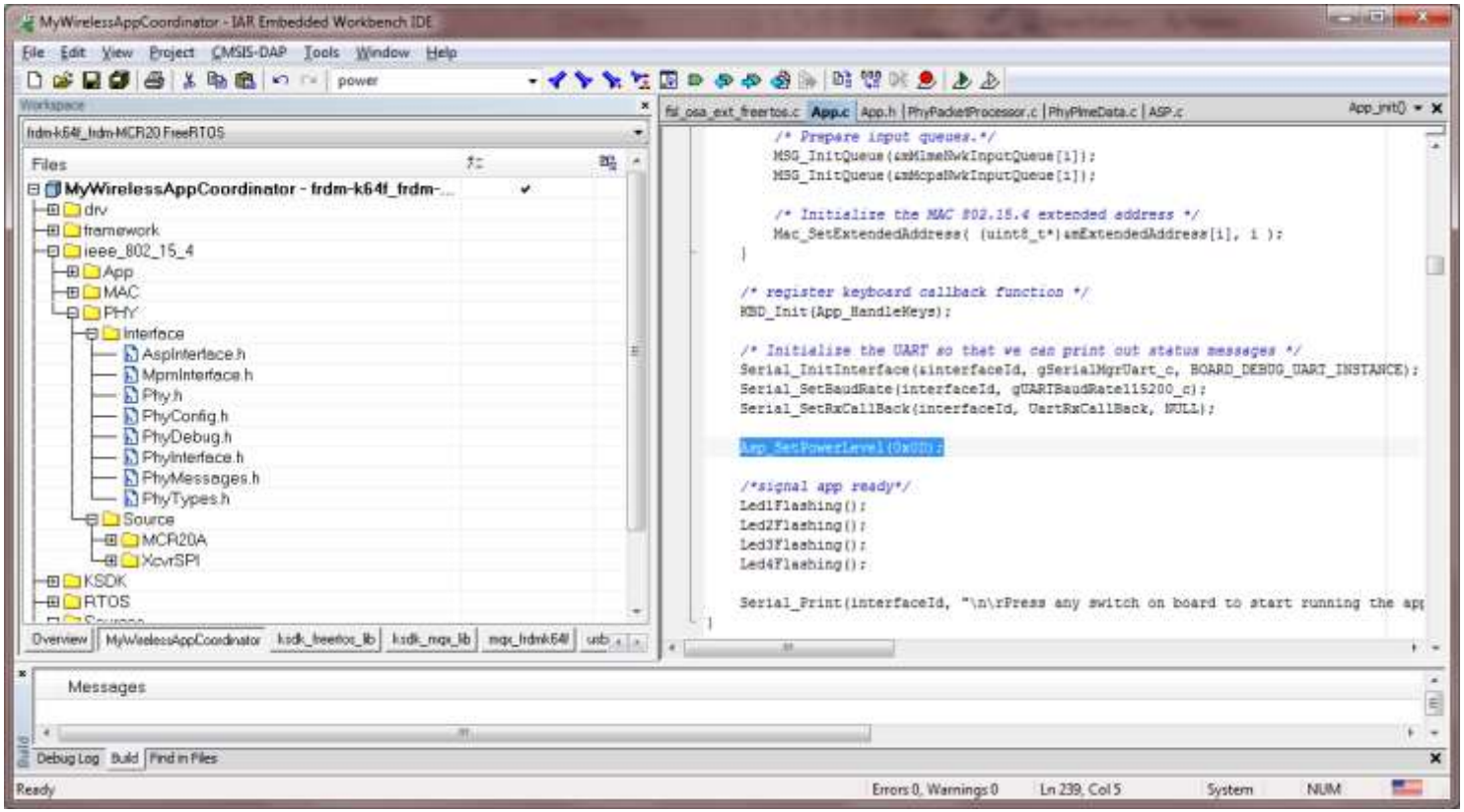
Configuring K64F/MCR20A Application (MyWirelessApp)

- Optional: Navigate to App.c, line 239 to configure the mode of the RGB LED flashing:
  - Default: *very* brightly white (some people might find it bothersome), or
  - Replace the 4 shown lines with `LED_StartSerialFlash;` for RGB flashing.



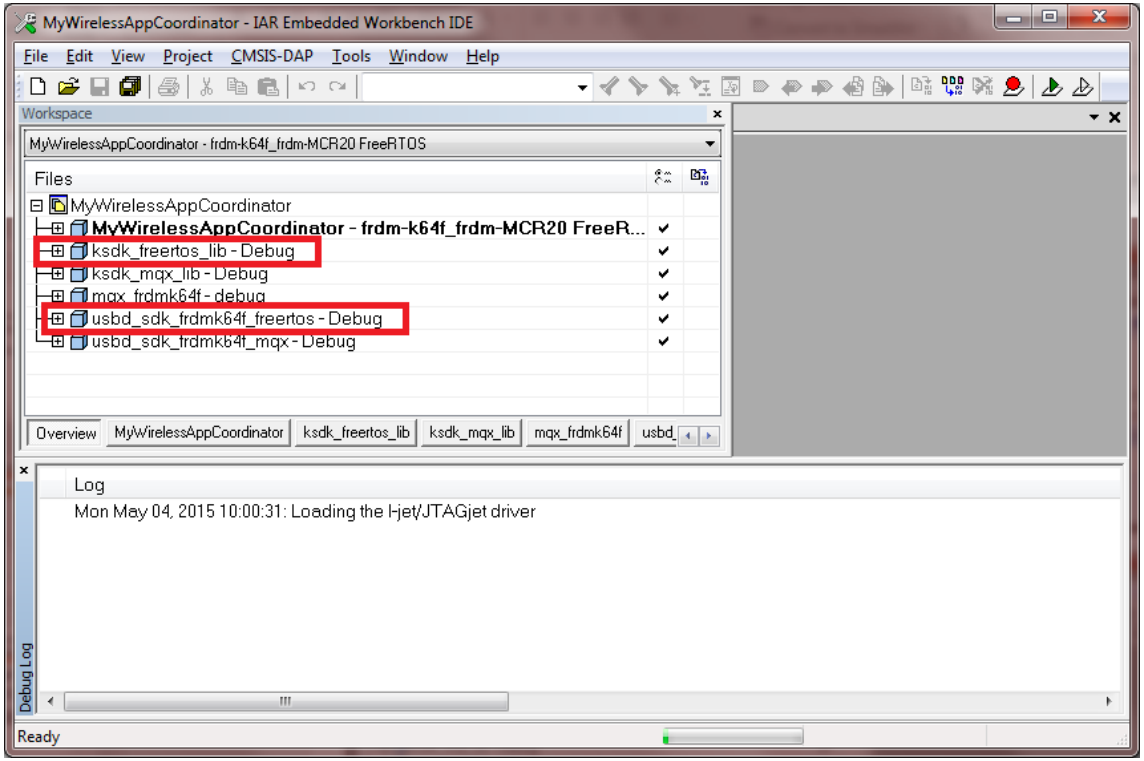
Configuring K64F/MCR20A Application (MyWirelessApp)

- Navigate to App.c, line 239 to configure the transmit power:
  - Default: 0 dBm (1milliwatt) – relatively high power, might interfere with neighbors on the same channel
  - Add the following line above the LED function calls: `Asp_SetPowerLevel(0x0D);` for -20 dBm Tx power



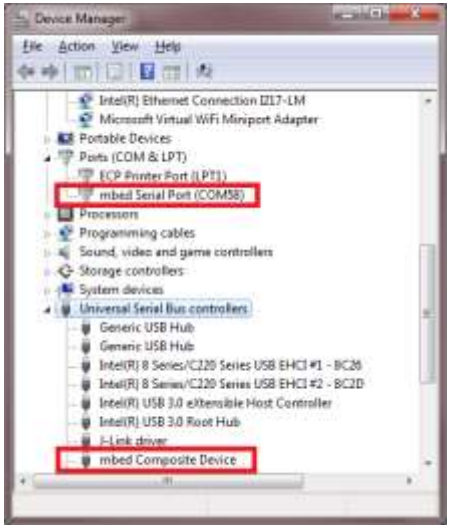
Building a Pre-configured K64F Application (MyWirelessApp)

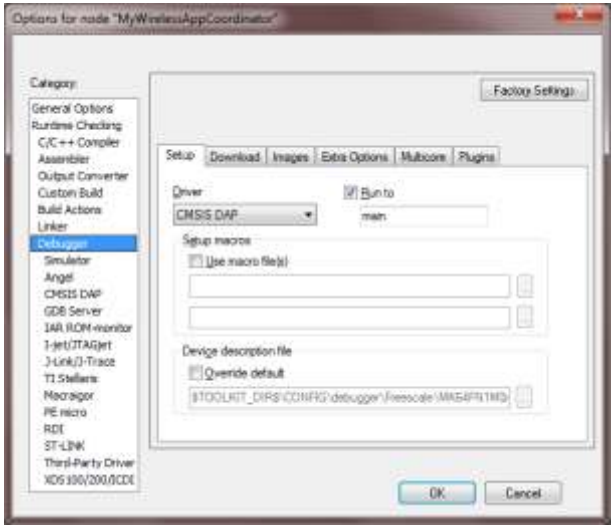
- Make sure to first build (right click->"Make") the two KSDK RTOS libraries required by the application: ksdk_freertos_lib and usb_sdk_frdmk64f_freertos.
- Build the application (right click->"Make").



Flashing a Pre-configured K64F Application (MyWirelessApp)

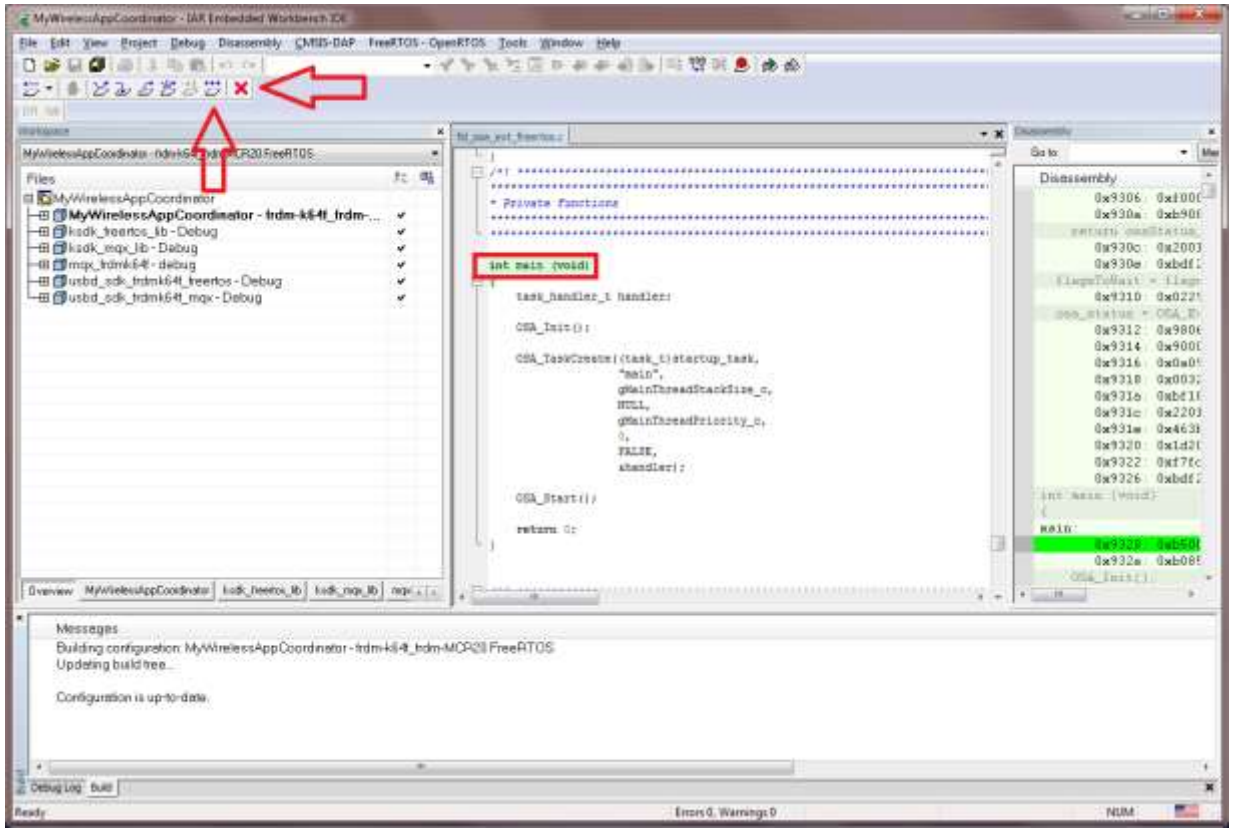
- Make sure you plug in the FRDM-K64F + FRDM-CR20A (referred to as coordinator) setup to your computer via the micro USB port on the K64F motherboard, labelled "**SDAUSB**".
- Make sure that the mBed **CMSIS-DAP** firmware derivers (pre-installed on your computer) have been successfully enabled by Windows by checking Device Manager
- Make sure to select CMSIS-DAP as the debugger driver and **SWD** as the interface in the project options (right-click->"Options").
- After the debugger settings are done and the application is built, click the green arrow labelled "**Download and Debug**" and wait about 10 seconds for the flash download.



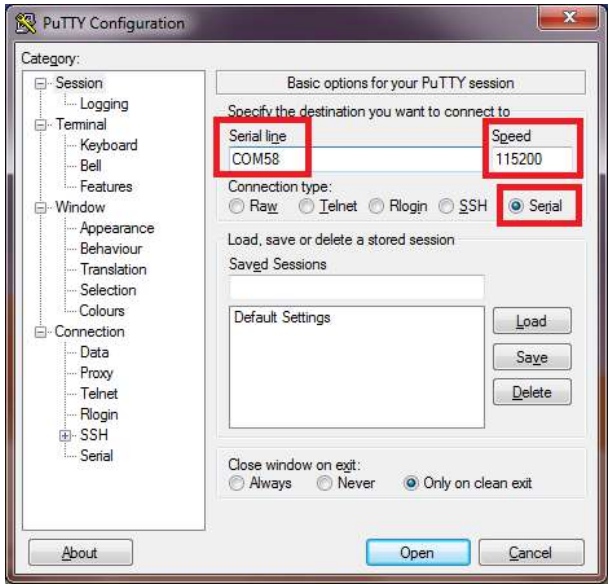


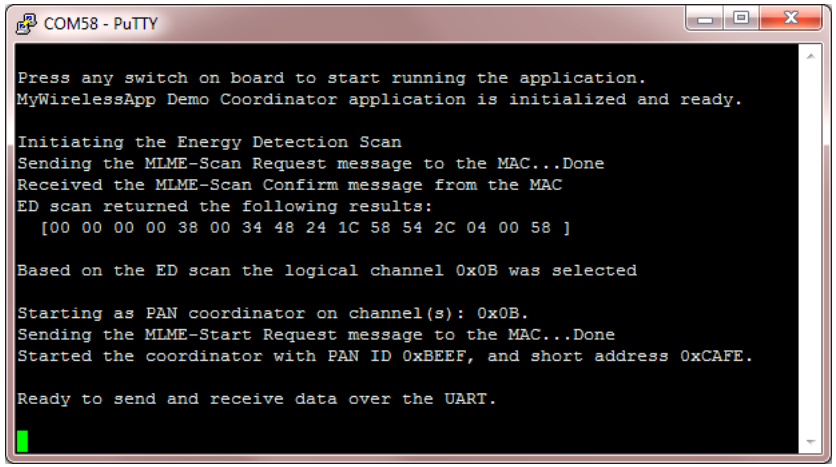
Debugging a Pre-configured K64F Application (MyWirelessApp)

- If everything went well, you should see the debugger window and the application stopped in the **main()** function.
- Click "**Go**" then "**Stop debugging**". The Board LED should be flashing either bright white or RGB, depending on how it was configured in the previous step.



Running the K64F MyWirelessApp Coordinator Demo





- Open a **PuTTY** terminal program (shortcut should be available on the desktop of your machine)
- Configure the same serial port shown in Device Manager as "**mBed Serial Port**" with **115200 baud**.
- Connect or reset the coordinator. The message to press any key should appear.
- Press either **SW1** or **SW2** on the FRDM-CR20A.
- The RGB LED should **stop flashing**.
- The coordinator should start on the **least occupied channel** in the mask and the serial terminal should show the progress.

Porting a MCR20A Application

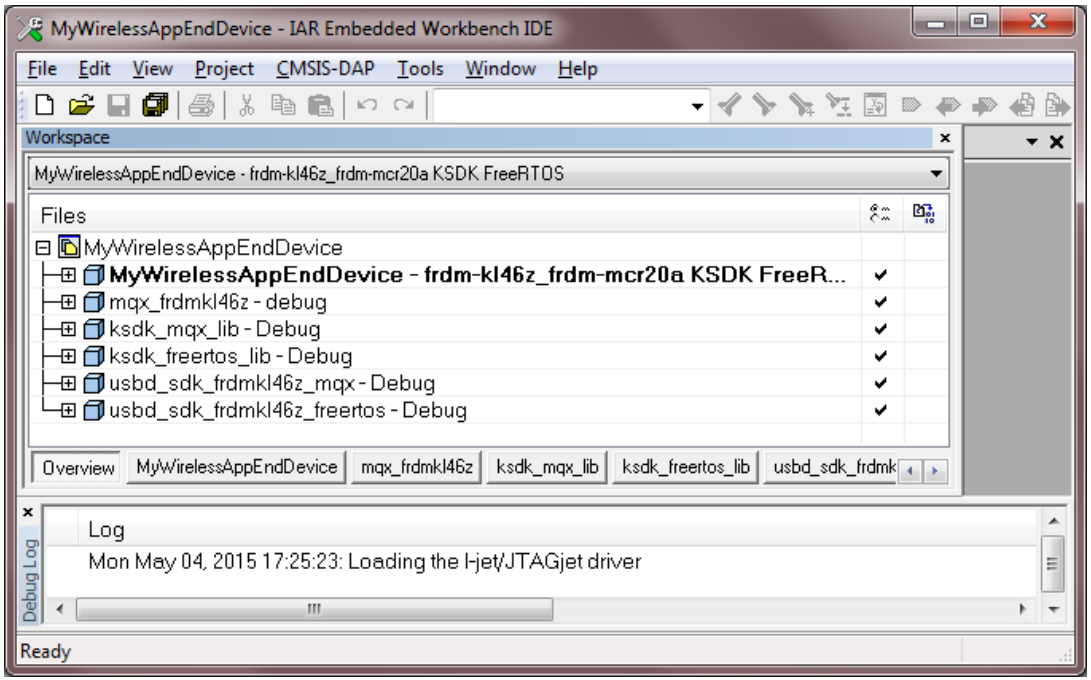
"MyWirelessApp" End Device from KL46Z to KL26Z

Porting a MCR20A Connectivity Application

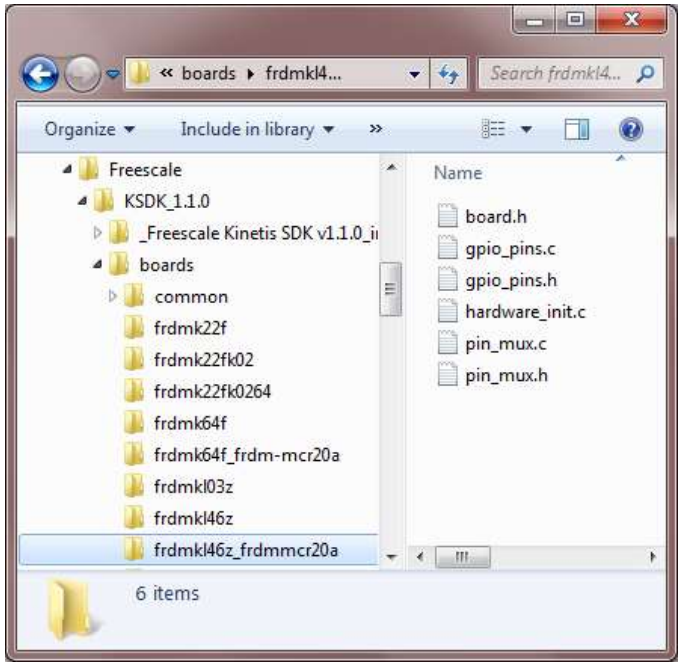
- Steps:
  - Set as baseline the FRDM-KL46Z + FRDM-CR20A MyWirelessApp End Device application
  - Identify the FRDM-KL46Z board and KL46Z processor specific components.
  - Create new application workspace and replace the board/processor specific components with FRDM-KL26 and KL26Z ones.
  - Build and run the FRDM-KL26Z + FRDM-CR20A application.
  - Associate with the FRDM-K64F + FRDM-CR20A previously configured 802.15.4 coordinator.

KL46Z/MCR20A MyWirelessApp End Device Application

- Navigate to C:\Freescale\MCR20A_IEEE_802_15_4\app\ieee_802_15_4\MyWirelessApp\EndDevice\frdm-kl46z_frdm-mcr20a and double click on the .eww file.
- Select the frdm-kl46z_frdm-mcr20a KSDK FreeRTOS build configuration.

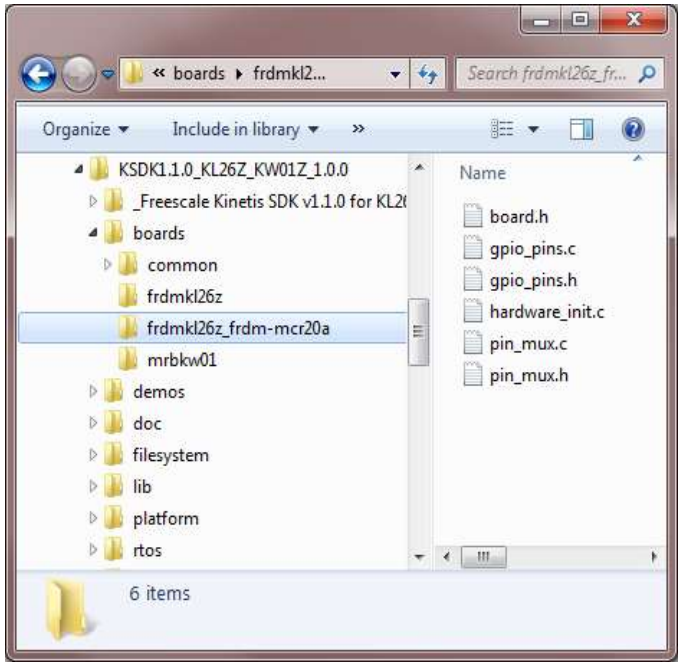


Board Configurations (Existing)



- FRDM-KL46Z and FRDM-K64F board configurations are duplicated.
- This avoids overwriting existing KSDK default configurations.
- Ensures proper configuration for operation with FRDM-CR20A.
- The files in the frdmkl46z_frdmmcr20a and frdmk64f_frdm-mcr20a folders contain the specific board configurations done for FRDM-CR20A: SPI, push-buttons and RGB LED

Platform Specific – Board Configurations (FRDM-KL26Z)



- Need to create FRDM-KL26Z board configuration in the KSDK v1.1 SA for KL26Z folder structure that will allow FRDM-CR20A compatibility and not overwrite the default KSDK settings
- A copy-paste of the existing frdmkl26z configuration is a good starting point
- For simplicity, this has already been done on your machine, in the form of the frdmkl26z_frdm-mcr20a folder.
- The files in the above folder are the most important board configuration elements which need to be modified for FRDM-CR20A proper operation.

Configuring the Board – Compare the Sources

- FRDM-KL46Z Generic Sources:
  - C:\Freescale\KSDK_1.1.0\boards\frdmkl46z\
- FRDM-KL46Z Specific Sources for FRDM-CR20A operation:
  - C:\Freescale\KSDK_1.1.0\boards\frdmkl46z_frdmmcr20a\

- FRDM-K26Z Specific Sources:
  - C:\Freescale\KSDK1.1.0_KL26Z_KW01Z_1.0.0\boards\frdmkl26z\
- FRDM-KL26Z Specific Sources for FRDM-CR20A operation:
  - C:\Freescale\KSDK1.1.0_KL26Z_KW01Z_1.0.0\boards\frdmkl26z_frdm-mcr20a\

- Tool used: **Beyond Compare** from **Scooter Software**
  - Right click on first file and choose 

  - Right click on first file and choose 

GPIO Pins – FRDM-KL46Z Switches Configuration

C:\Freescale\KSDK_1.1.0\boards\frdmkl46z\gpio_pins.c

11/18/2014 4:55:32 AM 4,926 bytes C,C++,C# Source ANSI PC

```
gpio_input_pin_user_config_t switchPins[] = {
    {
        .pinName = kGpioSW1,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntDisabled,
    },
    {
        .pinName = kGpioSW2,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntRisingEdge
    },
```

C:\Freescale\KSDK_1.1.0\boards\frdmkl46z_frdmmcr20a\gpio_pins.c

2/19/2015 4:41:44 PM 6,557 bytes C,C++,C# Source ANSI PC

```
gpio_input_pin_user_config_t switchPins[] = {
    {
        .pinName = kGpioSW1,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = kGpioSW2,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = kGpioSW3,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = kGpioSW4,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
```

165: 1 Default text

GPIO Pins – FRDM-KL26Z Switches Configurations

C:\...\boards\frdmkl26z_frdm-mcr20a\gpio_pins.c

4/28/2015 3:06:42 PM 5,214 bytes C,C++,C# Source ANSI PC

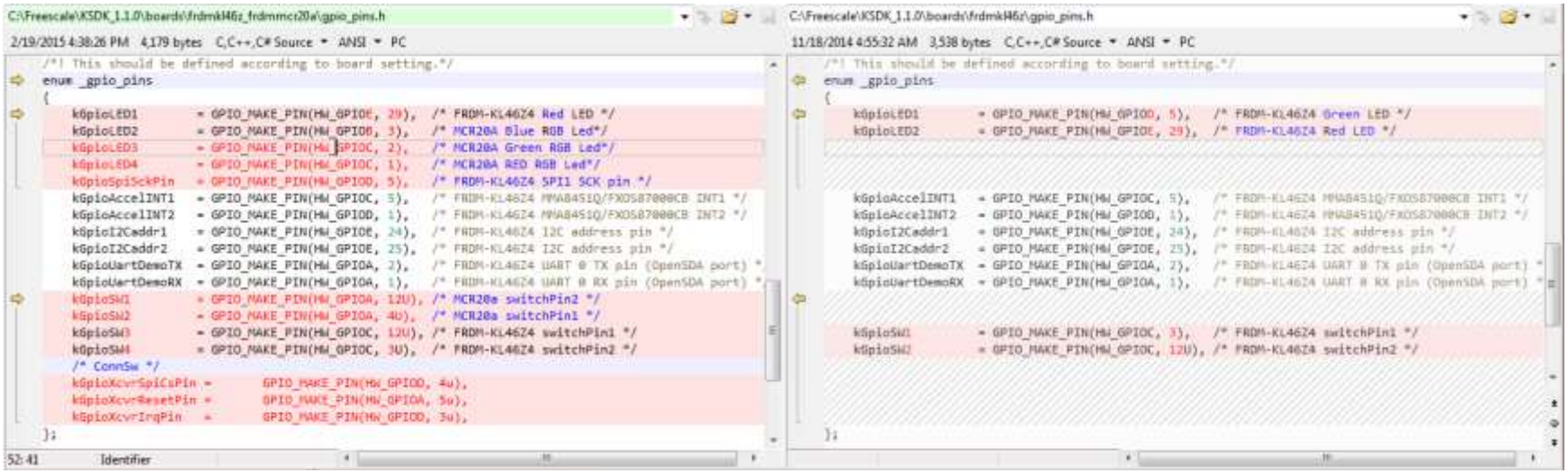
```
gpio_input_pin_user_config_t switchPins[] = {
    {
        .pinName = kGpioSW1,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = kGpioSW2,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = kGpioSW3,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntFallingEdge,
    },
    {
        .pinName = GPIO_PINS_OUT_OF_RANGE,
    }
};
```

C:\...\boards\frdmkl26z\gpio_pins.c

1/16/2015 6:36:50 PM 4,246 bytes C,C++,C# Source ANSI PC

```
gpio_input_pin_user_config_t switchPins[] = {
    {
        .pinName = kGpioSW1,
        .config.isPullEnable = true,
        .config.pullSelect = kPortPullUp,
        .config.isPassiveFilterEnabled = false,
        .config.interrupt = kPortIntDisabled,
    },
    {
        .pinName = GPIO_PINS_OUT_OF_RANGE,
    }
};
```

GPIO Pins – FRDM-KL46Z & FRDM-CR20A LED & Transceiver Pins Declaration



C:\Freescale\KSDK_1.1.0\boards\frdmkl46z_frdmmcr20a\gpio_pins.h

2/19/2015 4:38:26 PM 4,179 bytes C,C++,C# Source ANSI PC

```
/*! This should be defined according to board setting.*/
enum _gpio_pins
{
    kGpioLED1       = GPIO_MAKE_PIN(HW_GPIOE, 29),  /* FRDM-KL46Z4 Red LED */
    kGpioLED2       = GPIO_MAKE_PIN(HW_GPIOB, 3),   /* MCR20A Blue RGB Led*/
    kGpioLED3       = GPIO_MAKE_PIN(HW_GPIOC, 2),   /* MCR20A Green RGB Led*/
    kGpioLED4       = GPIO_MAKE_PIN(HW_GPIOC, 1),   /* MCR20A RED RGB Led*/
    kGpioSpiSckPin  = GPIO_MAKE_PIN(HW_GPIOD, 5),   /* FRDM-KL46Z4 SPI1 SCK pin */
    kGpioAccelINT1  = GPIO_MAKE_PIN(HW_GPIOC, 5),   /* FRDM-KL46Z4 MMA8451Q/FXOS87000CB INT1 */
    kGpioAccelINT2  = GPIO_MAKE_PIN(HW_GPIOD, 1),   /* FRDM-KL46Z4 MMA8451Q/FXOS87000CB INT2 */
    kGpioI2Caddr1   = GPIO_MAKE_PIN(HW_GPIOE, 24),  /* FRDM-KL46Z4 I2C address pin */
    kGpioI2Caddr2   = GPIO_MAKE_PIN(HW_GPIOE, 25),  /* FRDM-KL46Z4 I2C address pin */
    kGpioUartDemoTX = GPIO_MAKE_PIN(HW_GPIOA, 2),   /* FRDM-KL46Z4 UART 0 TX pin (OpenSDA port) */
    kGpioUartDemoRX = GPIO_MAKE_PIN(HW_GPIOA, 1),   /* FRDM-KL46Z4 UART 0 RX pin (OpenSDA port) */
    kGpioSW1        = GPIO_MAKE_PIN(HW_GPIOA, 12U), /* MCR20a switchPin2 */
    kGpioSW2        = GPIO_MAKE_PIN(HW_GPIOA, 4U),  /* MCR20a switchPin1 */
    kGpioSW3        = GPIO_MAKE_PIN(HW_GPIOC, 12U), /* FRDM-KL46Z4 switchPin1 */
    kGpioSW4        = GPIO_MAKE_PIN(HW_GPIOC, 3U),  /* FRDM-KL46Z4 switchPin2 */
    /* ConnSw */
    kGpioXcvrSpiCsPin =     GPIO_MAKE_PIN(HW_GPIOD, 4u),
    kGpioXcvrResetPin =     GPIO_MAKE_PIN(HW_GPIOA, 5u),
    kGpioXcvrIrqPin   =     GPIO_MAKE_PIN(HW_GPIOD, 3u),
};
```

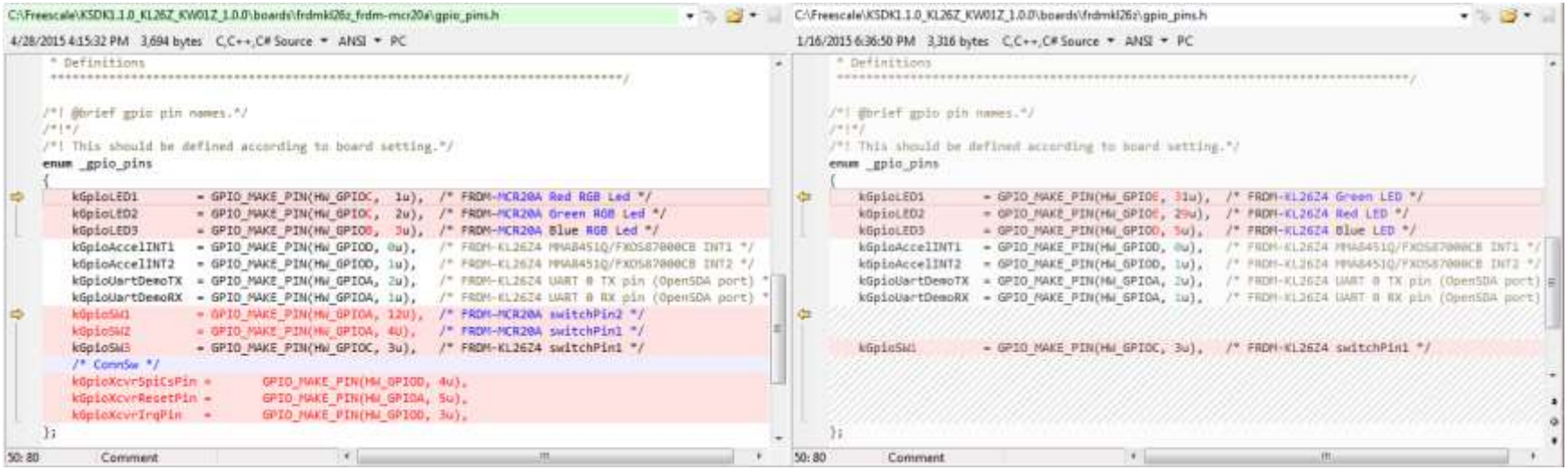
52: 41 Identifier

C:\Freescale\KSDK_1.1.0\boards\frdmkl46z\gpio_pins.h

11/18/2014 4:55:32 AM 3,538 bytes C,C++,C# Source ANSI PC

```
/*! This should be defined according to board setting.*/
enum _gpio_pins
{
    kGpioLED1       = GPIO_MAKE_PIN(HW_GPIOD, 5),   /* FRDM-KL46Z4 Green LED */
    kGpioLED2       = GPIO_MAKE_PIN(HW_GPIOE, 29),  /* FRDM-KL46Z4 Red LED */
    kGpioAccelINT1  = GPIO_MAKE_PIN(HW_GPIOC, 5),   /* FRDM-KL46Z4 MMA8451Q/FXOS87000CB INT1 */
    kGpioAccelINT2  = GPIO_MAKE_PIN(HW_GPIOD, 1),   /* FRDM-KL46Z4 MMA8451Q/FXOS87000CB INT2 */
    kGpioI2Caddr1   = GPIO_MAKE_PIN(HW_GPIOE, 24),  /* FRDM-KL46Z4 I2C address pin */
    kGpioI2Caddr2   = GPIO_MAKE_PIN(HW_GPIOE, 25),  /* FRDM-KL46Z4 I2C address pin */
    kGpioUartDemoTX = GPIO_MAKE_PIN(HW_GPIOA, 2),   /* FRDM-KL46Z4 UART 0 TX pin (OpenSDA port) */
    kGpioUartDemoRX = GPIO_MAKE_PIN(HW_GPIOA, 1),   /* FRDM-KL46Z4 UART 0 RX pin (OpenSDA port) */
    kGpioSW1        = GPIO_MAKE_PIN(HW_GPIOC, 3),   /* FRDM-KL46Z4 switchPin1 */
    kGpioSW2        = GPIO_MAKE_PIN(HW_GPIOC, 12U), /* FRDM-KL46Z4 switchPin2 */
};
```

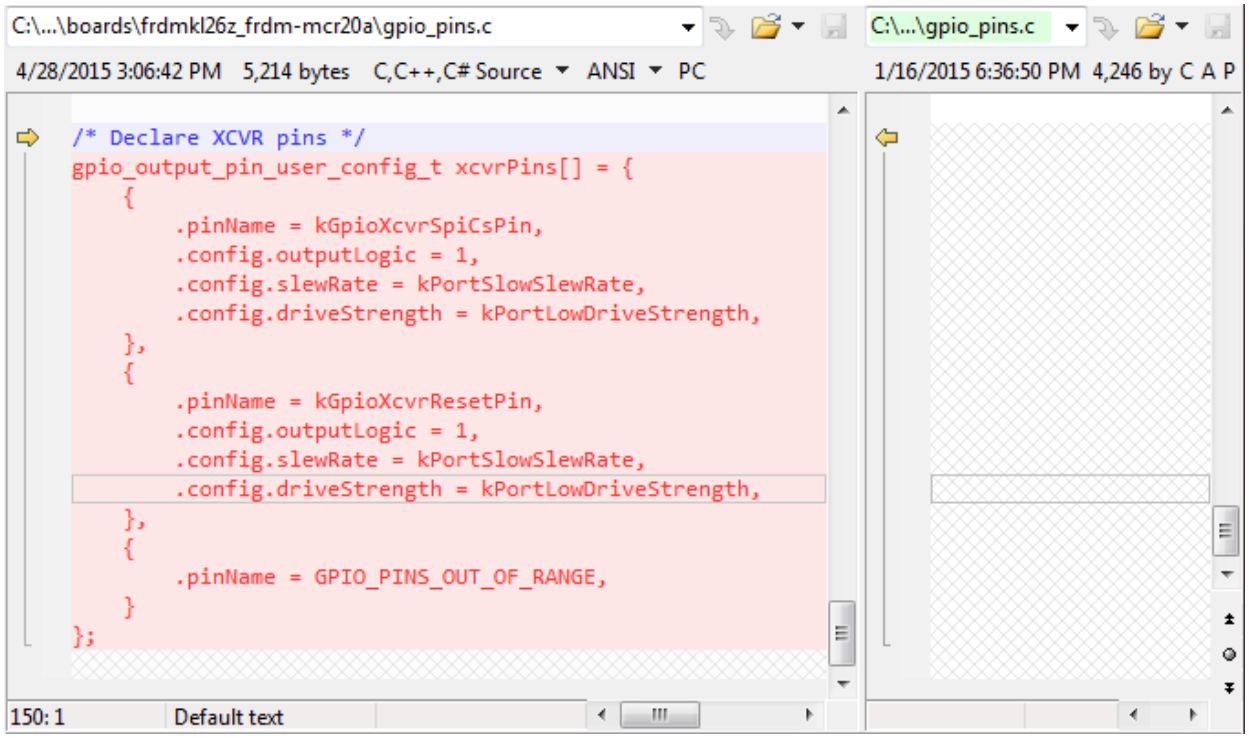
GPIO Pins – FRDM-KL26Z & FRDM-CR20A LED & Transceiver Pins Declaration



```
C:\Freescale\KSDK1.1.0_KL26Z_KW01Z_1.0.0\boards\frdmkl26z_frdm-mcr20a\gpio_pins.h
4/28/2015 4:15:32 PM  3,694 bytes  C,C++,C# Source  ANSI  PC

 * Definitions
 *****************************************************************************/

/*! @brief gpio pin names.*/
/*!*/
/*! This should be defined according to board setting.*/
enum _gpio_pins
{
    kGpioLED1       = GPIO_MAKE_PIN(HW_GPIOC,  1u),  /* FRDM-MCR20A Red RGB Led */
    kGpioLED2       = GPIO_MAKE_PIN(HW_GPIOC,  2u),  /* FRDM-MCR20A Green RGB Led */
    kGpioLED3       = GPIO_MAKE_PIN(HW_GPIOB,  3u),  /* FRDM-MCR20A Blue RGB Led */
    kGpioAccelINT1  = GPIO_MAKE_PIN(HW_GPIOD, 0u),   /* FRDM-KL26Z4 MMA8451Q/FXOS87000CB INT1 */
    kGpioAccelINT2  = GPIO_MAKE_PIN(HW_GPIOD, 1u),   /* FRDM-KL26Z4 MMA8451Q/FXOS87000CB INT2 */
    kGpioUartDemoTX = GPIO_MAKE_PIN(HW_GPIOA, 2u),   /* FRDM-KL26Z4 UART 0 TX pin (OpenSDA port) *
    kGpioUartDemoRX = GPIO_MAKE_PIN(HW_GPIOA, 1u),   /* FRDM-KL26Z4 UART 0 RX pin (OpenSDA port) *
    kGpioSW1        = GPIO_MAKE_PIN(HW_GPIOA, 12U),  /* FRDM-MCR20A switchPin2 */
    kGpioSW2        = GPIO_MAKE_PIN(HW_GPIOA, 4U),   /* FRDM-MCR20A switchPin1 */
    kGpioSW3        = GPIO_MAKE_PIN(HW_GPIOC, 3u),   /* FRDM-KL26Z4 switchPin1 */
    /* ConnSw */
    kGpioXcvrSpiCsPin =      GPIO_MAKE_PIN(HW_GPIOD, 4u),
    kGpioXcvrResetPin =      GPIO_MAKE_PIN(HW_GPIOA, 5u),
    kGpioXcvrIrqPin   =      GPIO_MAKE_PIN(HW_GPIOD, 3u),
};
50: 80  Comment
```

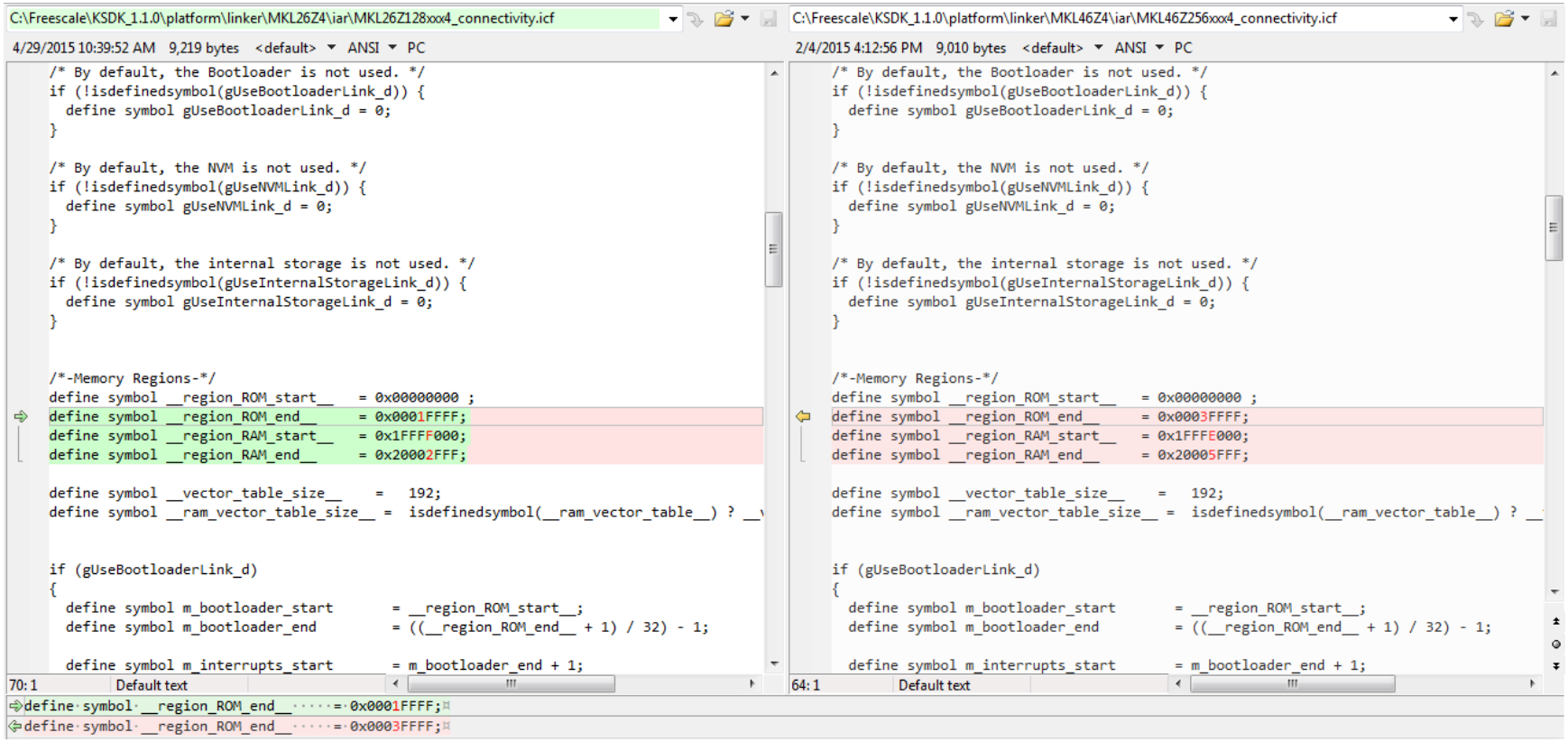
```
C:\Freescale\KSDK1.1.0_KL26Z_KW01Z_1.0.0\boards\frdmkl26z\gpio_pins.h
1/16/2015 6:36:50 PM  3,316 bytes  C,C++,C# Source  ANSI  PC

 * Definitions
 *****************************************************************************/

/*! @brief gpio pin names.*/
/*!*/
/*! This should be defined according to board setting.*/
enum _gpio_pins
{
    kGpioLED1       = GPIO_MAKE_PIN(HW_GPIOE, 31u),  /* FRDM-KL26Z4 Green LED */
    kGpioLED2       = GPIO_MAKE_PIN(HW_GPIOE, 29u),  /* FRDM-KL26Z4 Red LED */
    kGpioLED3       = GPIO_MAKE_PIN(HW_GPIOD, 5u),   /* FRDM-KL26Z4 Blue LED */
    kGpioAccelINT1  = GPIO_MAKE_PIN(HW_GPIOD, 0u),   /* FRDM-KL26Z4 MMA8451Q/FXOS87000CB INT1 */
    kGpioAccelINT2  = GPIO_MAKE_PIN(HW_GPIOD, 1u),   /* FRDM-KL26Z4 MMA8451Q/FXOS87000CB INT2 */
    kGpioUartDemoTX = GPIO_MAKE_PIN(HW_GPIOA, 2u),   /* FRDM-KL26Z4 UART 0 TX pin (OpenSDA port)
    kGpioUartDemoRX = GPIO_MAKE_PIN(HW_GPIOA, 1u),   /* FRDM-KL26Z4 UART 0 RX pin (OpenSDA port)


    kGpioSW1        = GPIO_MAKE_PIN(HW_GPIOC, 3u),   /* FRDM-KL26Z4 switchPin1 */
};
50: 80  Comment
```

Transceiver Pins Configuration (Both Setups)

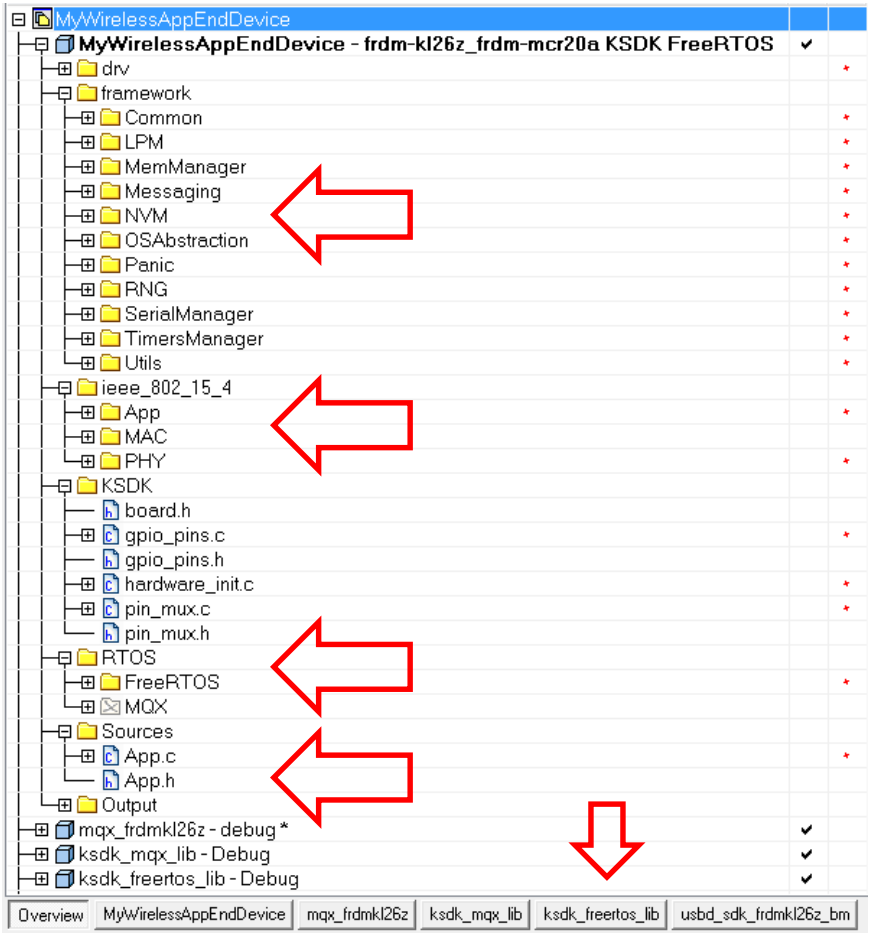


Platform Specific – Linker File

- Connectivity applications use **dedicated linker** files located in the **platform\linker** folder in the KSDK installation.
- For porting a MCR20A application from KL46Z to KL26Z, only the **memory map** needs to be re-defined in the new connectivity KL26Z linker file, starting from the KL46Z one.



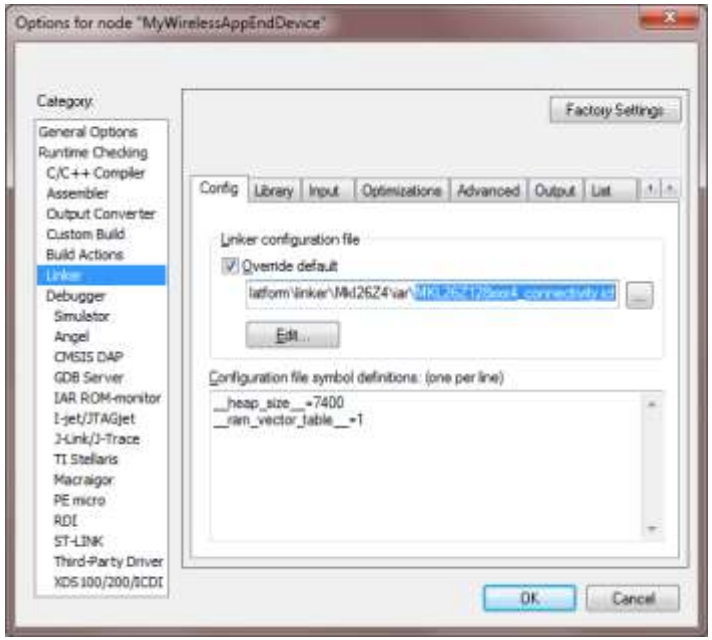
Creating the IAR EWARM Project for the “MyWirelessApp End Device” Application for FRDM-KL26Z

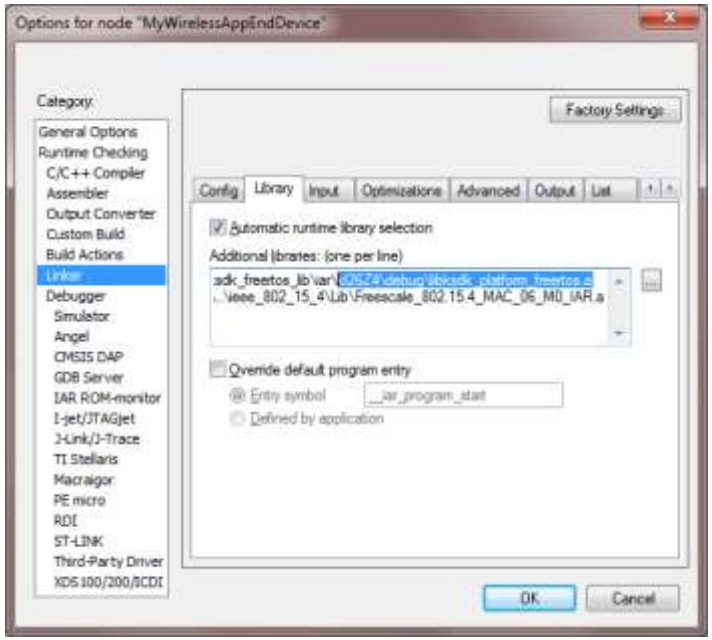


- Make sure to change the KSDK_PATH variable to the KL26Z SA SDK folder before opening any KL26Z workspace.
- A copy-paste of the existing frdmkl26z project and workspace is a good starting point
- For simplicity, this has already been done on your machine, in the form of the frdmkl26z_frdm-mcr20a folder.
- The files in the above folder are the most important board configuration elements which need to be modified for FRDM-CR20A proper operation.

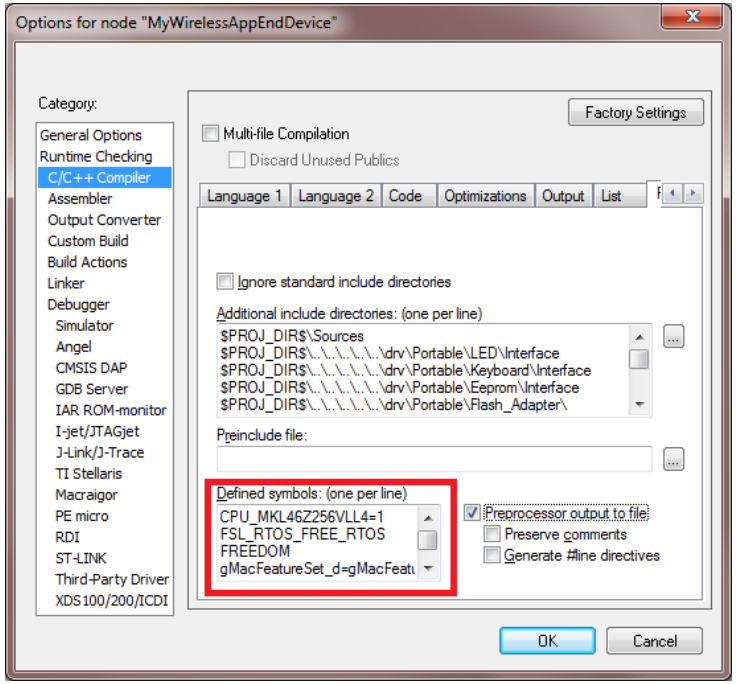
Configuring the Linker Options

- In the Linker->Config panel, select the connectivity linker file
- The linker file is already created on your machine having as a starting point the connectivity KL46Z one and NOT the original KSDK KL26Z file.
- In the Linker-> Library Panel, make sure that the KSDK KL26Z platform library for FreeRTOS is selected





Configuring the Preprocessor Definintions



- IAR Preprocessor Definitions:

```
CPU_MKL46Z256VLL4=1
FSL_RTOS_FREE_RTOS
gMacFeatureSet_d=gMacFeatureSet_06M0_d
gFsciIncluded_c=0
gRNG_HWSupport_d=gRNG_NoHWSupport_d
gRNG_UsePhyRngForInitialSeed_d=1
gSecLib_HWSupport_d=gSecLib_NoHWSupport_d
gKeyBoardSupported_d=1
gKBD_KeysCount_c=4
gLEDsOnTargetBoardCnt_c=4
gMCR20_ClkOutFreq_d=gCLK_OUT_FREQ_DISABLE
```

Preprocessor Definitions - Continued

Definition on FRDM-KL46Z	Description	Definition on FRDM-KL26Z
CPU_MKL46Z256VLL4=1	Processor definition	CPU_ MKL26Z128VLH4=1
FSL_RTOS_FREE_RTOS	RTOS definition used in the abstraction layer	FSL_RTOS_FREE_RTOS
gMacFeatureSet_d= gMacFeatureSet_06M0_d	IEEE 802.15.4 MAC feature set (2006, Cortex M0+)	gMacFeatureSet_d= gMacFeatureSet_06M0_d
gFsciIncluded_c=0	Freescale Serial Connectivity Interface included	gFsciIncluded_c=0
gRNG_HWSupport_d= gRNG_NoHWSupport_d	Hardware Random Number generator support	gRNG_HWSupport_d= gRNG_NoHWSupport_d
gRNG_UsePhyRngForInitialSeed_d= 1	Get initial RNG seed from over-the-air energy measurements	gRNG_UsePhyRngForInitialSeed_d= 1
gSecLib_HWSupport_d= gSecLib_NoHWSupport_d	Hardware acceleration for security (AES-128)	gSecLib_HWSupport_d= gSecLib_NoHWSupport_d
gKeyBoardSupported_d=1	Keyboard on the FRDM-CR20A supported	gKeyBoardSupported_d=1
gKBD_KeysCount_c=4	Number of keys avalable	gKBD_KeysCount_c=4
gLEDsOnTargetBoardCnt_c=4	Number of LEDs available	gLEDsOnTargetBoardCnt_c=4
gMCR20_ClkOutFreq_d= gCLK_OUT_FREQ_DISABLE	Have the MCR20A output a clock signal for the MCU	gMCR20_ClkOutFreq_d= gCLK_OUT_FREQ_DISABLE

Preprocessor Definitions in Sources - SPI

```
#include "EmbeddedTypes.h"
#include "SPI.h"
#include "fsl_clock_manager.h"
#include "pin_mux.h"

#if BOARD_USE_DSPI
    #include "fsl_dspi_hal.h"
    #include "fsl_dspi_master_driver.h"
#else
    #include "fsl_spi_hal.h"
    #include "fsl_spi_master_driver.h"
#endif
```

- Transceiver Driver SPI transfers support
  - C:\Freescale\MCR20A_IEEE_802_15_4\ieee_802_15_4\Source\Phy\Source\XcvrSpi\SPI.c
  - Contains wrapper code over KSDK drivers that enable different types of SPI modules: SPI (KL46Z, KL26Z) or DSPI (K64F).
  - Ensures unified SPI transfer API for the MCR20A transceiver driver.
  - The **BOARD_USE_DSPI** macro selects DSPI or SPI

Putting it all Together

Connect the FRDM-K64F Coordinator with the FRDM-KL26Z End Device

Build and Download the FRDM-KL26Z Application

- Configure the RF channel mask to be the same as on the K64F Coordinator.
- Build the KSDK KL26Z FreeRTOS platform library from the workspace. The USB library is not necessary.
- Build the MyWirelessApp End Device KL26Z configuration for FreeRTOS
- Plug the FRDM-KL26Z/FRDM-CR20A node to the USB port of the FRDM-KL26Z labelled “SDA” and wait for the CMSIS-DAP drivers to be installed.

Create a Simple Two Node Network

- Open **two PuTTY terminals** corresponding to the virtual COM Ports of exposed by the two nodes: FRDM-KL26Z/FRDM-CR20A and FRDM-K64F/FRDM-CR20A.
- Configure the serial ports with **115200 baud**.
- **Reset the boards**. Each RGB LED should **start flashing** and each terminal should display the following text:

```
Please press any key on board to start running the application
```

- Press **SW1** or **SW2** on the FRDM-CR20A of the **coordinator** (K64F).
- The RGB LED should **stop flashing** and the corresponding terminal should display the same text as shown before in this presentation.
- Now press **SW1** or **SW2** on the FRDM-CR20A of the end device (KL26Z). The RGB led on this board should also **stop flashing**.

Create a Simple Two Node Network - Continued

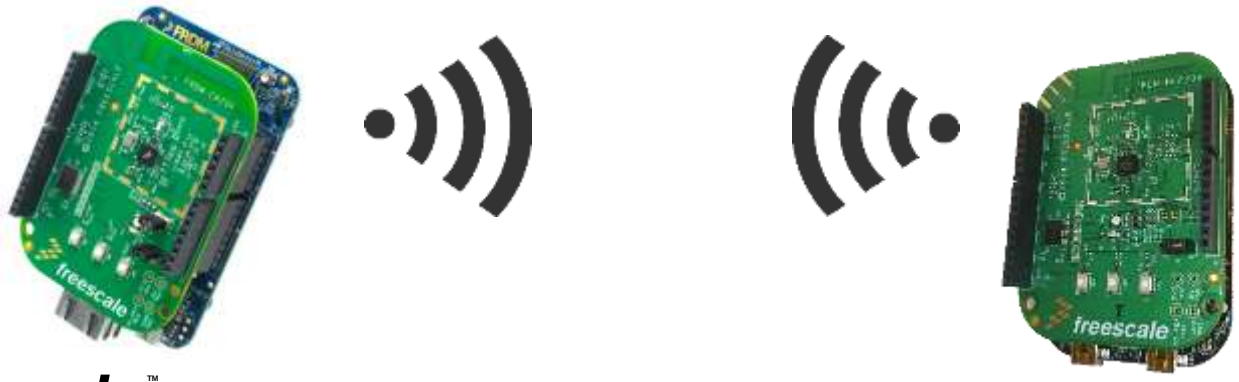
The terminals for the two nodes should show the following messages:

COM58 - PuTTY

```
Press any switch on board to start running the application.
MyWirelessApp Demo Coordinator application is initialized and ready.

Initiating the Energy Detection Scan
Sending the MLME-Scan Request message to the MAC...Done
Received the MLME-Scan Confirm message from the MAC
ED scan returned the following results:
  [00 0C 24 3C 00 40 68 60 30 00 34 2C 30 00 00 00 ]

Based on the ED scan the logical channel 0x0B was selected

Starting as PAN coordinator on channel(s): 0x0B.
Sending the MLME-Start Request message to the MAC...Done
Started the coordinator with PAN ID 0xBEEF, and short address 0xCAFE.

Ready to send and receive data over the UART.

Received an MLME-Associate Indication from the MAC
Sending the MLME-Associate Response message to the MAC...Done
Received an MLME-Comm-Status Indication from the MAC
```

COM68 - PuTTY

```
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Received an MLME-Beacon Notify Indication
Found a coordinator with the following properties:
-----------------------------------------------------
Address..........0xCAFE
PAN ID...........0xBEEF
Logical Channel...0x0B
Beacon Spec.......0xCFFF
Link Quality......0x9F

Associating to PAN coordinator on channel 0x0B
Sending the MLME-Associate Request message to the MAC...Done
Successfully associated with the coordinator.
We were assigned the short address 0x0001

Ready to send and receive data over the UART.
```



Create a Simple Two Node Network - Conclusion

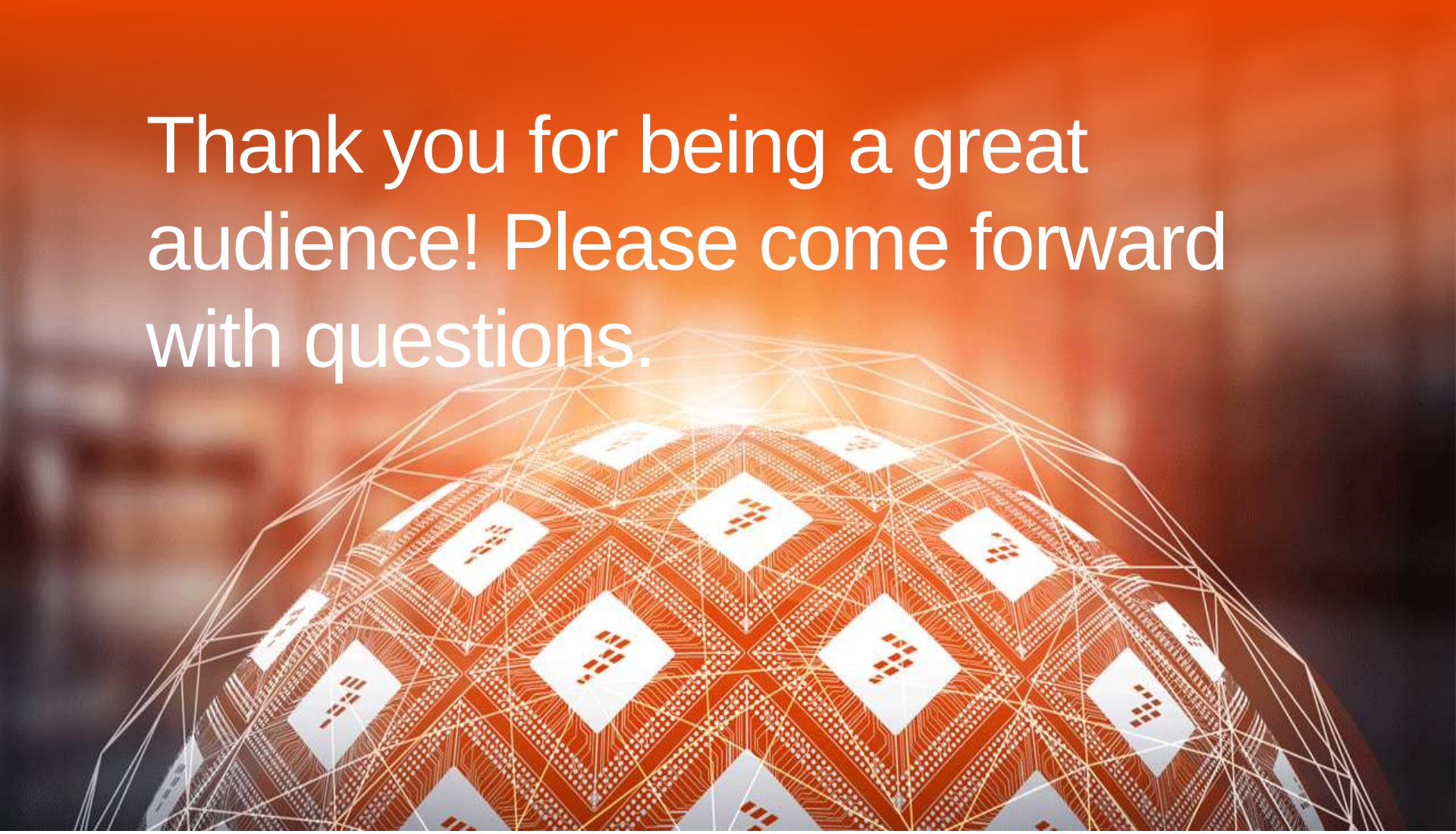
- What actually happens: **Coordinator**:
  - Starts Energy Detection Scan on the channel mask selected (only one in our case)
  - If the channel is free, the coordinator starts a network with PAN ID 0xBEEF and assigns itself a short 16-bit address of 0xCAFE
  - It then waits with the receiver open to receive the a message from an end device
  - Upon receiving the Active Scan from the device it sends a beacon
  - Upon receiving the association request from the end device it starts the association procedure
- What actually happens: **End Device**:
  - Starts an Active Scan procedure on the selected channel mask
  - Upon receiving the beacon from the coordinator it sends an association request
- After the association, characters typed in a terminal console will be sent over the air and printed in the other. **Try it!**

Conclusions, Q&A and Session Closing

Conclusions

- By now, you should be able to:
  - Appreciate the **flexibility** and differentiating features that MCR20A offers in adding connectivity to microcontroller systems.
  - Effectively describe, at a high level, the Freescale **MCR20A development setup** for IEEE® 802.15.4 applications.
  - Understand the way the IEEE® 802.15.4 MAC and PHY **integrate with the Kinetis SDK**.
  - Describe the **key configuration elements** for switching a microcontroller with another for MCR20A usage.
  - Apply the knowledge gained in this presentation to begin or refine your design efforts for **IoT applications** using MCR20A.

Thank you for being a great audience! Please come forward with questions.



www.Freescale.com