How to use the Si4x6x Radios with Energy Micro MCU Starter Kits

1. Introduction

This document summarizes how to use Silicon Laboratories' Si4x6x radios with the EFM32LG Leopard Gecko and EFM32TG Tiny Gecko MCU starter kits. The IAR ARM tool chain is used for the FW development. This document provides an overview about the recommended collaterals and describes how the different hardware components need to be setup.

1.1. Prerequisites

1.1.1. Documentation

The following documents will provide information related to setting up hardware components. It is recommended that the designer refer to the following documents prior to starting the firmware development.

1. MCU related collateral (EFM32LG990256 or EFM32TG840F32):
   a. Data Sheets:
      - EFM32LG990 Data Sheet
      - EFM32TG840 Data Sheet

   b. Errata:
      - EFM32LG990 Errata History
      - EFM32LG990 Errata Rev D
      - EFM32TG840 Errata History
      - EFM32TG840 Errata Rev C

   c. Reference Manuals:
      - EFM32LG Reference Manual
      - EFM32TG Reference Manual

   d. User Guides:
      - EFM32 Starter Kit Documentation Package: EFM32TG-STK3300 Tiny Gecko Starter Kit User Guide
      - EFM32LG-STK3600 Leopard Gecko Starter Kit User Guide

   e. API documentation. After installing the Simplicity Studio, the API documentation is located at "C:\Users\[user name]\AppData\Roaming\energymicro\EM_CMSIS_X.XX.X_DOC\index.html" or you can open it in Simplicity Studio: "Simplicity Studio 'Software and Kits' API Documentation".

   f. Application Note:
      - AN0043: Debug and Trace
2. Radio related collaterals
   a. AN633: Programming Guide for EZRadioPro® Si4x6x Devices

   b. Si4x6x API Documentation: EZRadioPRO-API_v1.0.3_12062012.zip

   c. AN632: WDS User's Guide for EZRadioPRO® Devices
3. IAR Toolchain integration

Both EFM32TG-STK3300 and EFM32LG-STK3600 have integrated Segger J-Link USB debugger/emulator with debug out functionality.

Copyright © 2013 by Silicon Laboratories

1.1.2. Software

Install the following software prior to starting the development.

1. Silicon Labs Simplicity Studio:
   http://www.silabs.com/products/mcu/Pages/simplicity-studio.aspx
2. Silicon Labs Wireless Development Suite:
   http://www.silabs.com/products/wireless/EZRadio/Pages/WirelessDevelopmentSuite.aspx
3. IAR ARM toolchain: trial version can be downloaded from: http://www.iar.com
4. J-Link Related documents: http://www.segger.com

1.1.3. Required hardware components

1. 2 pcs. EFM32LG-STK3600 or EFM32TG-STK3300 starter kit
2. 2 pcs. Si446x RFPico cards
3. 2 pcs. EFM-RFPico card adapter
4. 2 pcs. USB cable

2. Setup

2.1. Hardware

2.1.1. General HW Setup

The Energy Micro MCU starter kits (EFM32TG-STK3300 and EFM32LG-STK3600) have integrated Segger J-Link USB debugger/emulator with debug out functionality. The boards need to be connected to the PC through the debug USB connector (DBG connector). The MCU starter kit and the radio RFPico board is powered from USB.

Note: CR2032 battery cannot provide enough current to power both the MCU starter kit and the radio RFPico board, since the maximum peak discharge current of those batteries is 20 mA that can be easily exceeded in transmit mode.

2.1.2. Additional Steps to Connect Si446x Radio

1. Connect the RFPico adapter card (Figure 1) to the EFM32TG-STK3300 or EFM32LG-STK3600 according to Figure 2 and Figure 3.

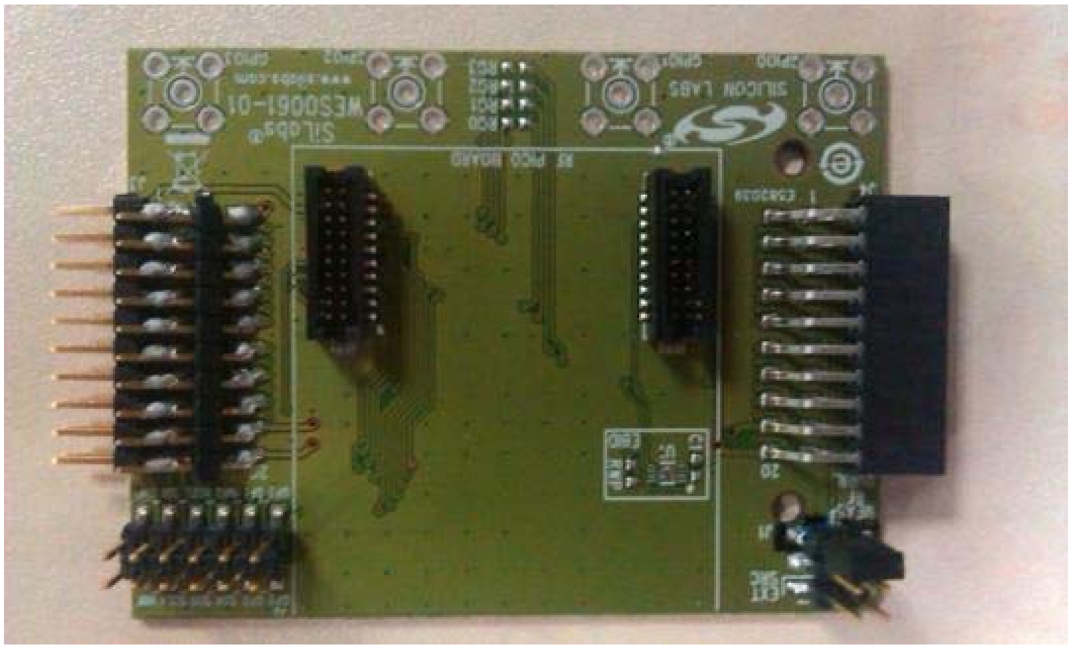


Figure 1. RF Pico Adapter Board

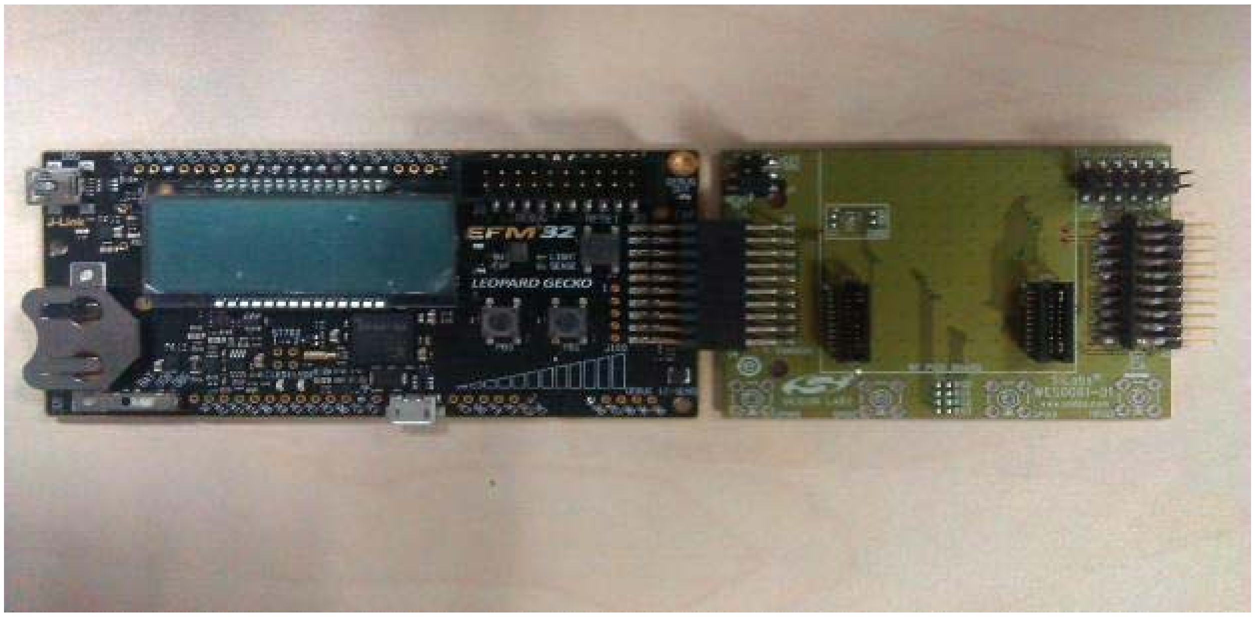


Figure 2. Adapter Boards is Connected to the EFM32LG-STK3600

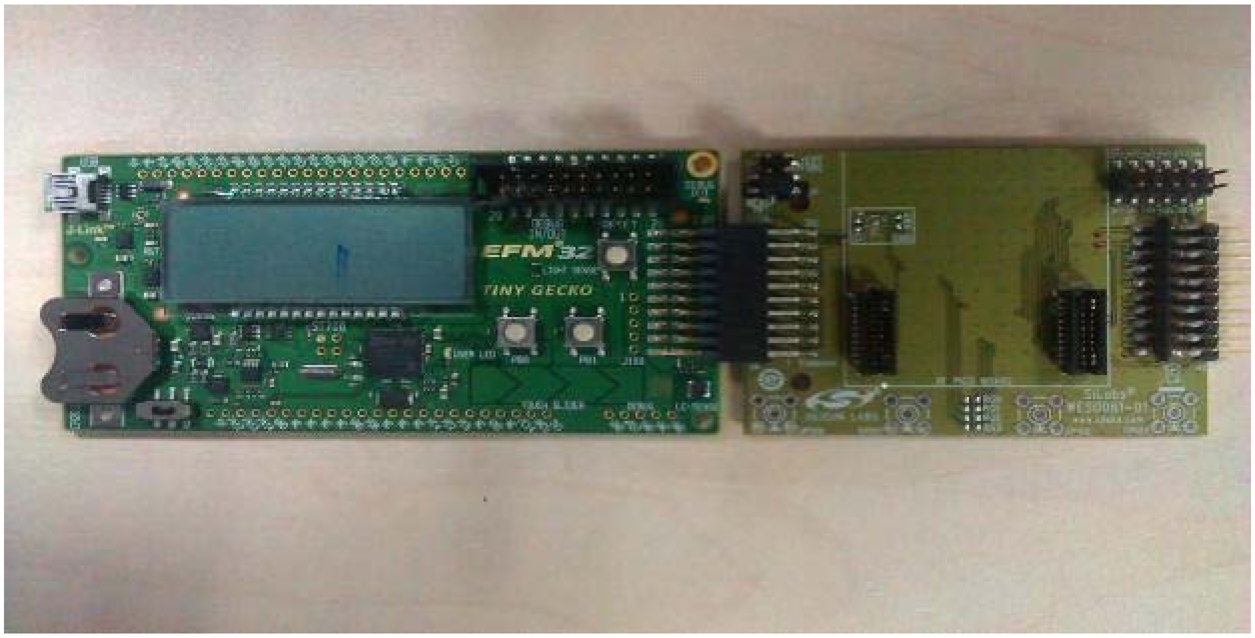


Figure 3. Adapter Boards is Connected to the EFM32TG-STK3300

2. Connect the Si446x RFPico card (Figure 4) to the RFPico adapter board.

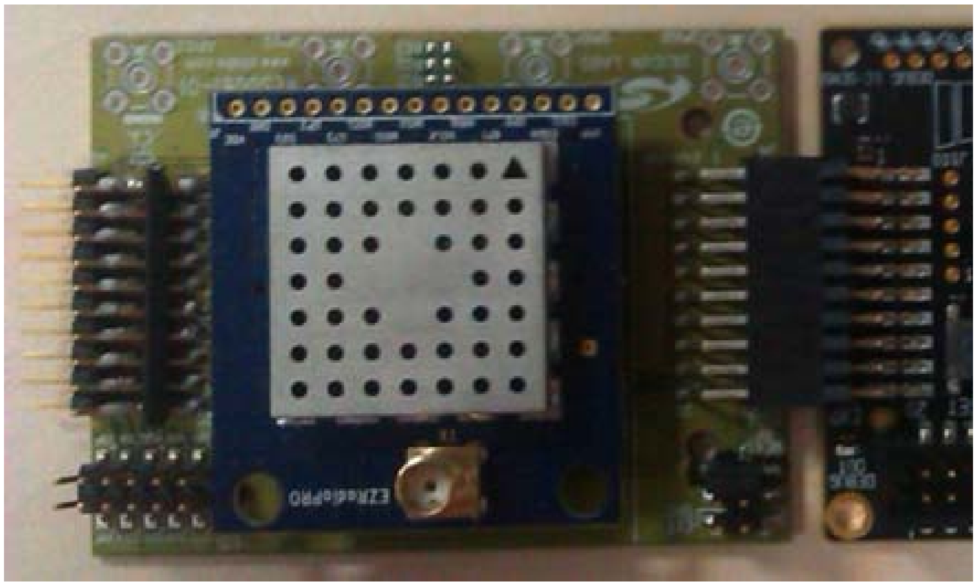


Figure 4. RF Pico Card

2.2. Software

The following PC tools need to be installed:

1. IAR tool chain: Refer to the following application notes for more details about the recommended FW development and debugging process with the Energy Micro MCUs:
   - EFM32TG-STK3300 Tiny Gecko Starter Kit User Guide and EFM32LG-STK3600 Leopard Gecko Starter Kit User Guide
   - AN0043: an0043_efm32_debug_trace_capabilities.pdf
2. Simplicity Studio
3. WDS: refer to the application note, “AN632: WDS User's Guide for EZRadioPRO® Devices” for more details about the installation and usage of WDS.

3. Si4x6x Demo Application

This following section describes the example projects ported to the Energy Micro MCU platform.

3.1. FW Architecture

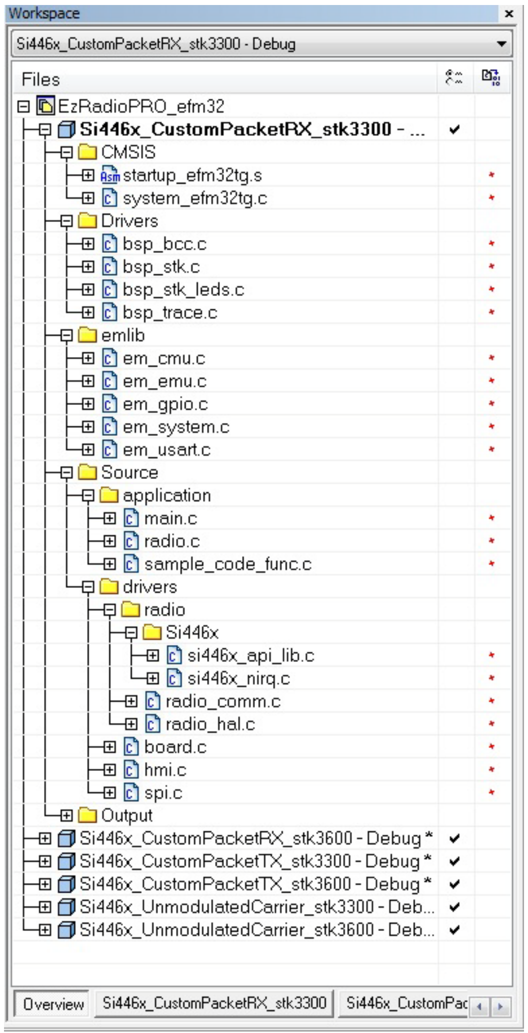
The example projects are based on three Si4x6x example projects:

- Unmodulated Carrier (Continuous Wave)"
- Custom Packet Transmission"
- Customer Packet Reception"

These projects are described in application note, “AN633: Programming Guide for EZRadioPro® Si4x6x Devices”.

The architecture of the EFM32 example project source code is a combination of the Energy Micro MCU driver, the radio driver, and the application layer.

The three example projects are provided as a combined IAR workspace, called EzRadioPRO_efm32.



```
Workspace
Si446x_CustomPacketRX_stk3300 - Debug
Files
⊟ EzRadioPRO_efm32
├⊟ Si446x_CustomPacketRX_stk3300 - ...          ✓
│ ├⊟ CMSIS
│ │ ├⊞ startup_efm32tg.s                            *
│ │ └⊞ system_efm32tg.c                             *
│ ├⊟ Drivers
│ │ ├⊞ bsp_bcc.c                                    *
│ │ ├⊞ bsp_stk.c                                    *
│ │ ├⊞ bsp_stk_leds.c                               *
│ │ └⊞ bsp_trace.c                                  *
│ ├⊟ emlib
│ │ ├⊞ em_cmu.c                                     *
│ │ ├⊞ em_emu.c                                     *
│ │ ├⊞ em_gpio.c                                    *
│ │ ├⊞ em_system.c                                  *
│ │ └⊞ em_usart.c                                   *
│ ├⊟ Source
│ │ ├⊟ application
│ │ │ ├⊞ main.c                                     *
│ │ │ ├⊞ radio.c                                    *
│ │ │ └⊞ sample_code_func.c                         *
│ │ └⊟ drivers
│ │   ├⊟ radio
│ │   │ ├⊟ Si446x
│ │   │ │ ├⊞ si446x_api_lib.c                       *
│ │   │ │ └⊞ si446x_nirq.c                          *
│ │   │ ├⊞ radio_comm.c                             *
│ │   │ └⊞ radio_hal.c                              *
│ │   ├⊞ board.c                                    *
│ │   ├⊞ hmi.c                                      *
│ │   └⊞ spi.c                                      *
│ └⊞ Output
├⊞ Si446x_CustomPacketRX_stk3600 - Debug *      ✓
├⊞ Si446x_CustomPacketTX_stk3300 - Debug *      ✓
├⊞ Si446x_CustomPacketTX_stk3600 - Debug *      ✓
├⊞ Si446x_UnmodulatedCarrier_stk3300 - Deb...   ✓
└⊞ Si446x_UnmodulatedCarrier_stk3600 - Deb...   ✓
Overview | Si446x_CustomPacketRX_stk3300 | Si446x_CustomPac
```

Figure 5. Directory Structure of the Example Projects

After installing the Simplicity Studio, the BSP files will be installed in the user directory, typically in the following location: Win7: C:\Users\[username]\AppData\Roaming\energymicro\kits\common\bsp, or something similar (depending on your OS/Windows version; here we use Win7 for the example). All files in the board support package are prefixed by bsp.

The drivers of the MCU provided by the Simplicity Studio will be located at:

Win7: C:\Users\[username]\AppData\Roaming\energymicro\emlib\.

The two CMSIS files are located at:

Win7: C:\Users\[username]\AppData\Roaming\energymicro\Device\EnergyMicro\[SerialName]\Source\IAR and Win7: C:\Users\[username]\AppData\Roaming\energymicro\Device\EnergyMicro\[SerialName]\Source.

Note: SerialName is the serial name of the MCU. For example, EFM32LG for EFM32LG990F256 and EFM32TG for EFM32TG840F32.

All files are located under the main directory, called EzRadioPRO. The workspace file is under the folder named IAR. Do not change the name of this folder because the workspace file and all the project files need to be located here. If the folder name is changed, the workspace file and project files cannot be integrated into the Simplicity Studio.

All the radio driver code for the example projects can be found under the folder, "radio_drivers". The application code for different projects is under different folders.

'Clearup.bat' can be used to delete the intermediate files generated by the complier.

3.2. Generated Files (energyAware Designer)

The energyAware Designer is used to configure the MCU peripherals and generate the initialization code for the application. It is similar to the 'Configuration Wizard' for the 8-bit MCUs of Silicon Labs.

The following step-by-step guide helps to configure the Leopard Gecko MCUs for the EFM32LG-STK3600 starter kit:

1. Create a new project for EFM32LG990F256.
2. Start the configuration from the default settings.
3. The following peripherals are used for the example projects: GPIO, USART1
4. USART1
    a. Enable.
    b. Module location: 1.
    c. Select mode: SPI
    d. Baudrate: 5000000, maximum 10M.
    e. Databits in frame: 8
    f. Enable 'SPI Master mode' and 'Most Significant Bit first'
    g. Select 'Clock idle low, sample on rising edge'
    h. PRS for USART RX: Disabled
    i. Uncheck ‘Enable AUTOTX mode’
    j. Enable 'TX', 'RX' and 'CLK', do not enable 'CS'
    k. Click ‘Pin# L11' and set this pin to be 'Push-pull'
    l. Click 'Pin# K11' and set this pin to be 'Input enable'
    m. Click 'Pin# J9' and set this pin to be 'Push-pull'
5. GPIO
    a. Check 'Enabled' to enable the whole GPIO module.
    b. Enable these pins: PB9(J7), PB10(J8), PB11(L5), PB12(L6), PC0(H1), PC3(J2), PC4(K2), PC5(L2), PD3(J10), PE2(F10), PE3(E11).
    c. Click the 'Pin#' of these pins and configure them to be 'Input enable': PB9(J7), PB10(J8), PB11(L5), PB12(L6), PC0(H1), PC4(K2), PC5(L2).
    d. Click the 'Pin#' of these pins and configure them to be 'Push-pull': PC3(J2), PD3(J10), PE2(F10), PE3(E11).

Complete the following step-by-step guide to configure the Tiny Gecko MCUs for the EFM32TG-STK3300 starter kit:

1. Create a new project for EFM32TG840F32.
2. Start the configuration from the default settings.
3. The following peripherals are used for the example projects: GPIO, USART1
    a. USART1
        i. Enable
        ii. Module location: 1
        iii. Select mode: SPI
        iv. Baudrate: 5000000, maximum 10M
        v. Databits in frame: 8
        vi. Enable 'SPI Master mode' and 'Most Significant Bit first'
        vii. Select 'Clock idle low, sample on rising edge'
        viii. PRS for USART RX: Disabled
        ix. Uncheck ‘Enable AUTOTX mode’
        x. Enable 'TX', 'RX' and 'CLK', do not enable 'CS'
        xi. Click 'Pin# 28' and set this pin to be 'Push-pull'
        xii. Click 'Pin# 29' and set this pin to be 'Input enable'
        xiii. Click 'Pin# 30' and set this pin to be 'Push-pull'
    b. GPIO
        i. Check 'Enabled' to enable the whole GPIO module.
        ii. Enable these pins: PB11(21), PB12(22), PC4(13), PC5(14), PC12(45), PC13(46), PD7(35), PD8(36).
        iii. Click the 'Pin#' of these pins and configure them to be 'Input enable': PB11(21), PB12(22), PC4(13), PC12(45), PC13(46), PD8(36).
        iv. Click the 'Pin#' of these pins and configure them to be 'Push-pull': PC5(14), PD7(35).
        v. After the configuration is finished, generate the code for the application.

3.3. Demo Files

The radio application uses three example projects as the base, they are "Unmodulated Carrier", "Custom Packet TX", and "Customer Packet RX". The following sections explain the demo.

3.3.1. Drivers

1. Platform definition ('[project name]\bsp_def.h')
    - Platform definitions and application specific common includes are located here.
2. HW ('[project name]\platform_defs.h'):
    - The two new platforms are added under definition EFM32LG990F256 and EFM32TG840F32. Notice that the BSP of the kits and the drivers provided by Simplicity Studio use these two definitions to include the corresponding drivers.
3. SPI ('radio_drivers\spi.c'):
    - SPI driver is added.
4. Radio HAL ('radio_drivers\radio\*.*):
    - All the drivers are modified to work on EFM32TG-STK-3300 and EFM32LG-STK3600.

3.3.2. Application

1. Main ('[project name]\application\main.c'):
   Application specific code parts are to implement the function of this project. For example, transmitting the packet, receiving the packet or implementing the continuous wave transmission.
2. Radio files
   a. '[project name]\application\radio.c' :The radio specific code sections are the same as in the original example. However, few GPIO-related set/clear definitions are modified according to platform requirements.
   b. '[project name]\application\radio_config.h' is the radio configuration file. The property settings listed in this file define the behavior of the radio (modulation format, frequency, packet handler settings, etc.). It can be easily customized for the desired radio parameters using WDS and generating a new header file. Refer to the application note, “AN632: WDS User's Guide for EZRadioPRO® Devices” for more details on how to generate a new radio header file (sections 1.1.1 and 2.1.4.4).

3.4. How to Run the Demo

Follow these steps to run the example projects on the HW platform:

1. In order to develop your application, copy the whole 'EzRadioPRO' folder to the MCU kit example folder.

The EFM32TG-STK-3300 it should be copied to

C:\Users\[username]\AppData\Roaming\energymicro\kits\EFM32TG_STK3300\examples\.

The EFM32LG-STK3600 it should be copied to

C:\Users\[username]\AppData\Roaming\energymicro\kits\EFM32LG_STK3600 \examples\.

2. Open the workspace file EzRadioPRO_efm32.eww with IAR at:

   C:\Users\[username]\AppData\Roaming\energymicro\kits\[kit name]\examples\EzRadioPRO\IAR.

Note: The example has also been integrated into the Simplicity Studio; the user can open the projects from the Simplicity Studio directly:

   a. Open the Simplicity Studio and click Example on the "Software and Kits" area. See Figure 6.
   b. Click EzRadioPRO after you select the correct kit; here it is EFM32LG_STK3600. See Figure 7.

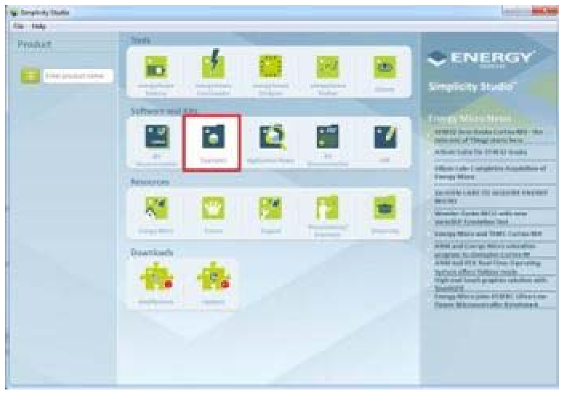
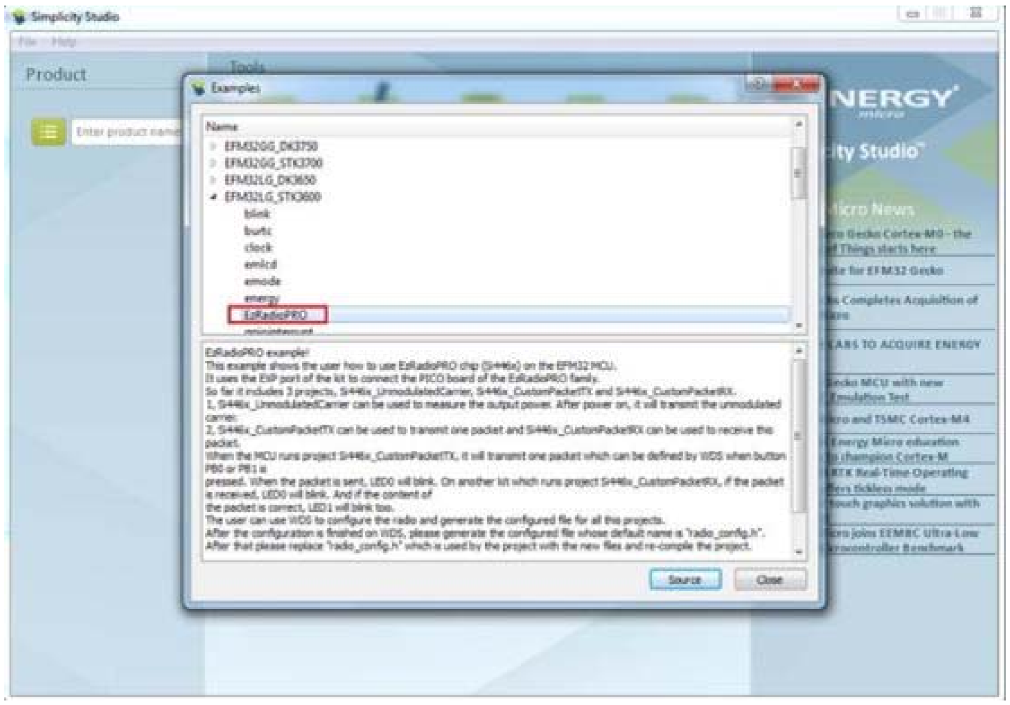


Figure 6. Simplicity Studio



```
EzRadioPRO example!
This example shows the user how to use EzRadioPRO chip (Si446x) on the EFM32 MCU.
It uses the EXP port of the kit to connect the PICO board of the EzRadioPRO family.
So far it includes 3 projects, Si446x_UnmodulatedCarrier, Si446x_CustomPacketTX and Si446x_CustomPacketRX.
1, Si446x_UnmodulatedCarrier can be used to measure the output power. After power on, it will transmit the unmodulated
carrier.
2, Si446x_CustomPacketTX can be used to transmit one packet and Si446x_CustomPacketRX can be used to receive this
packet.
When the MCU runs project Si446x_CustomPacketTX, it will transmit one packet which can be defined by WDS when button
PB0 or PB1 is
pressed. When the packet is sent, LED0 will blink. On another kit which runs project Si446x_CustomPacketRX, if the packet
is received, LED0 will blink. And if the content of
the packet is correct, LED1 will blink too.
The user can use WDS to configure the radio and generate the configured file for all the projects.
After the configuration is finished on WDS, please generate the configured file whose default name is "radio_config.h".
After that please replace "radio_config.h" which is used by the project with the new files and re-compile the project.
```

Figure 7. Example Box

3. Build two development platforms by connecting together the base board, RFPico adapter, and the RFPico card. Make sure that the RFPico boards are designed for the same frequency band.
4. Build the demo project with the desired radio_config.h radio configuration header file. The radio configuration header file can be generated from WDS.
5. Download the project into both development boards.

3.5. Expected operation

After the compiled project is downloaded to the development boards, the example projects should work in the following way:

1. Si446x_UnmodulatedCarrier transmits continuous wave (CW) signal. The output power can be observed with a spectrum analyzer through the SMA connector of the RFPico board. This operation is typically used for lab testing and evaluation.

If the demo is operational, the two user LEDs on the MCU starter board should blink with ~0.5 Hz.

2. Si446x_CustomPacketTX is typically used together with the Si446x_CustomPacketRx to build an RF link between the boards. Upon pressing the push button on the MCU starter board, if programmed with the Si446x_CustomPacketTX demo, the radio will transmit a packet and blink the LED. The following buttons and LEDs are used on the different MCU starter boards:
    a. EFM32LG_STK3600: PB1 or PB0 will initiate the packet transmission; LED0 is turned on while transmitting.
    b. EFM32TG_STK3300: PB0 will initiate the packet transmission; LED is turned on while transmitting.

Note: PB1 is not used on the EFM32TG_STK3300 since the corresponding MCU GPIO is connected to the GPIO2 pin of the radio.

3. Si446x_CustomPacketRX blinks the LEDs upon packet reception:
    a. EFM32LG_STK3600: user LED0 blinks if a packet is received with the selected synch word and LED1 blinks if the content of the packet payload matches the expected content.
    b. EFM32TG_STK3300: user LED will blink only if a packet is received with the expected payload.

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
https://www.siliconlabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Patent Notice
Silicon Labs invests in research and development to help our customers differentiate in the market with innovative low-power, small size, analog-intensive mixed-signal solutions. Silicon Labs' extensive patent portfolio is a testament to our unique approach and world-class engineering team.

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Silicon Laboratories assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Silicon Laboratories assumes no responsibility for the functioning of undescribed features or parameters. Silicon Laboratories reserves the right to make changes without further notice. Silicon Laboratories makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Silicon Laboratories assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Silicon Laboratories products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Silicon Laboratories product could create a situation where personal injury or death may occur. Should Buyer purchase or use Silicon Laboratories products for any such unintended or unauthorized application, Buyer shall indemnify and hold Silicon Laboratories harmless against all claims and damages.

Silicon Laboratories and Silicon Labs are trademarks of Silicon Laboratories Inc.
Other products or brandnames mentioned herein are trademarks or registered trademarks of their respective holders.