nRF5 Series: Developing with SEGGER Embedded Studio

Getting Started Guide

v1.1

1159720_163 v1.1 / 2018-10-16

Contents

Revision history

Date	Version	Description
October 2018	1.1	Added Adding files on page 24
July 2018	1.0	First release

1 Introduction

This guide will help you get started with your nRF5 Series *Development Kit (DK)* and developing your application with *SEGGER Embedded Studio (SES)*.

If you have worked with any of Nordic Semiconductor's products before, you are probably familiar with the required software tools. In this case, this guide will mostly provide reference information.

If this is your first time developing software for a Nordic Semiconductor *System on Chip (SoC)*, this guide will help you set up your development toolchain and guide you through all the steps that are necessary to develop, program, test, and debug your application.

This guide describes how to work with *SES*. *SES* is a cross-platform *Integrated Development Environment (IDE)*, so you can run it on different operating systems. For use with Nordic Semiconductor devices, you can get a free license that has no limitations.

There are two Getting Started Guides that show how to work with different *IDE*s:

- nRF5 Series: Developing with SEGGER Embedded Studio (this document)
- nRF5 Series: Developing on Windows with ARM Keil MDK

Check out the Nordic DevZone for additional setup information and help.

2 Minimum requirements

Ensure that you have all the required hardware and that your computer fulfills the software requirements.

Hardware requirements

- One of the following nRF5 Series development kits:
  - nRF52840 *DK*
  - nRF52 *DK*
  - nRF51 *DK*
- Micro-USB 2.0 cable
- Personal computer (PC) or Mac
- Smartphone or tablet that supports *Bluetooth*® Low Energy
- Optional: nRF52840 Dongle or nRF51 Dongle

Software requirements

One of the following operating systems:

- Windows 7, Windows 8, or Windows 10
- macOS
- Linux

3 Related documentation

This guide is a starting point and gives essential information for getting started with software development on nRF5 Series devices. For more detailed information, check the documentation of the separate components and tools.

Development Kit User Guides

- nRF52840 Development Kit
- nRF52 Development Kit
- nRF51 Development Kit

Dongle User Guides

- nRF52840 Dongle
- nRF51 Dongle

Compatibility Matrices

- nRF52840 Compatibility Matrix
- nRF52832 Compatibility Matrix
- nRF52810 Compatibility Matrix
- nRF51 Series Compatibility Matrix

nRF5 SDK documentation

- nRF5 SDK

Tools User Guides

- nRF5x Command Line Tools
- nRF Connect Bluetooth Low Energy

4 Development kits, boards, and chips

Each nRF5 Series *DK* contains a board, which in turn contains a chip. Some of Nordic Semiconductor's tools target the chip, while others target the development board or the development kit. Therefore, you must be aware of the relation between the different components.

You can find all compatibility information in the compatibility matrix for each chip. The following table summarizes the information that you must be aware of for programming your development board.

Development kit	Development board	Chip
nRF52840 *DK*	PCA10056	nRF52840
nRF52840 Dongle	PCA10059	nRF52840
nRF52 *DK*	PCA10040	nRF52832/nRF52810
nRF51 *DK*	PCA10028	nRF51422
nRF51 Dongle	PCA10032	nRF51422

Table 1: Relation between development kits, boards, and chips

5 SoftDevices

A *SoftDevice* is a wireless protocol stack that complements an nRF5 Series *SoC*. Nordic Semiconductor provides them as qualified, precompiled binary files. While it is possible to build applications without using a *SoftDevice*, all nRF5 SDK example applications that use Bluetooth Low Energy or ANT™ require a *SoftDevice*.

See the compatibility matrices for detailed information about which *SoftDevice* versions are supported for each chip. The following table summarizes the usage scenarios for each *SoftDevice*.

Protocol	**Role**	**Chip**	**SoftDevice**
Bluetooth Low Energy	Peripheral	• nRF51422 • nRF51822	S110
		• nRF52810 • nRF52832	S112
	Central or Peripheral	• nRF51422 • nRF51822	S120
	Central and Peripheral	• nRF51422 • nRF51822	S130
		• nRF52832	S132
		• nRF52840	S140
ANT		• nRF51422	S210
		• nRF52832	S212
Bluetooth Low Energy and ANT	Peripheral	• nRF51422	S310
	Central and Peripheral	• nRF52832	S332

Table 2: SoftDevice overview

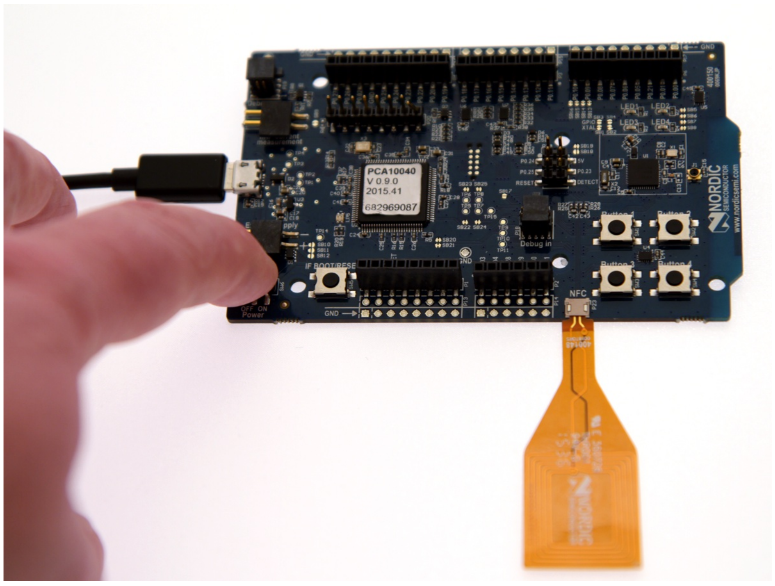
6 Running a first test

Before you start developing, program and run a simple application on your development board to ensure that the board functions as expected and the communication between your computer and your development board works.

Before you begin, download the latest compatible version of the nRF5 SDK. For information about which SDK supports which IC revisions, check the compatibility matrices.

The nRF5 SDK contains precompiled HEX files of the most common examples. Extract the zip file into a folder of your choosing.

1. Power up the nRF5 Series development board:
    a) Connect one end of a micro-USB 2.0 cable to the USB connector on the board and the other end to one of your PC's USB host ports.
    b) Slide the **power switch** to "ON".



    Observe that **LED1** starts blinking.

2. Open a file explorer and confirm that the development board has appeared as a removable drive named "JLINK".
3. In the folder where you extracted the nRF5 SDK, navigate to `examples\peripheral\blinky\hex`.
4. Select the HEX file that corresponds to your development board and copy it to the JLINK drive.
    The development board will now restart and run the application.

If the LEDs on the board start blinking, you have successfully programmed your first application! Next, continue to set up your development toolchain and program a more advanced application.

7 Setting up your toolchain

Before you can start developing, you must install the required software. This software includes tools to connect to your development board, an *IDE* for developing your application, and the nRF5 SDK that provides libraries and example applications.

See Nordic tools and downloads on page 10 for an overview of available tools and the links to download the latest versions for your operating system.

The following tools are required:

- nRF5 SDK
- SEGGER Embedded Studio (SES)
- SEGGER J-Link Software and Documentation Pack

The following tools are optional:

- nRF5x Command Line Tools for Windows, Linux32, Linux64, or macOS (all including nrfjprog)

See the following sections for installation instructions.

7.1 Nordic tools and downloads

This overview lists all available Nordic Semiconductor tools and supported *IDE*s. Not all of these tools are required. To help you pick the *IDE* and tools you want to use, see the following sections for common setup scenarios.

Development IDE

Pick one of the IDEs with a compiler supported by Nordic:

IDE	Windows	Linux	OSX
SEGGER Embedded Studio (SES)	Yes	Yes	Yes
MDK-ARM Keil µVision	Yes	No	No
GNU/GCC	Yes	Yes	Yes
IAR	Yes	No	No

SES is the recommended platform. It is free for use with nRF devices.

Essential tools

You need to download these Nordic tools to develop with our devices.

Tool	Description	Download	Documentation	Protocol
SDK (Software Development Kit)	Application examples, source files, SoftDevices	Windows/Linux	nRF5 SDK nRF5 SDK for Mesh nRF5 SDK for Thread and Zigbee	BLE/ANT Bluetooth Mesh Thread and Zigbee
nRF5x Command Line Tools	Collection of command line tools, like nrfjprog, mergehex	nRF5x Tools Windows32 nRF5x Tools Linux32 nRF5x Tools Linux64 nRF5x Tools OSX	nRF5x Command Line Tools	BLE/ANT

Optional tools

These tools are not essential, but we recommend that you use them.

Tool	Description	Download	Documentation	Protocol
SoftDevice	Wireless protocol stack	Click the Download tab on: nRF52840 Product page nRF52832 Product page nRF52810 Product page nRF51822 Product page nRF51422 Product page	SoftDevice Specifications	BLE/ANT
nRF Connect for Desktop	Expandable desktop tool with several apps, including: • Peer device emulator • Power Profiler • Programmer • Cloud Gateway	nRF Connect for OS-X nRF Connect for Ubuntu Linux nRF Connect for Windows	nRF Connect Bluetooth Low Energy	BLE
nRF Connect for Mobile	Peer device emulator app for smartphones	Android v4.3 or later IOS v8 or later		BLE
Nordic nRF Toolbox app	App that contains all the Nordic apps	Android v4.3 or later IOS v8 or later Windows Phone v8.1 or later		BLE
nRF5x pynrfjprog	Simple Python interface for the nrfjprog DLL	nRF5x pynrfjprog	nRF5x pynrfjprog	BLE/ANT
ANTware II	Peer device emulator for the ANT protocol running on computers	ANTware II		ANT
nRF Sniffer	App for monitoring on-air traffic	nRF Sniffer download	nRF Sniffer	BLE
nRF Thread Topology Monitor	Tool for visualizing Thread mesh network topology in real time	nRF Thread Topology Monitor for Ubuntu Linux 64-bit nRF Thread Topology Monitor for Windows	nRF Thread Topology Monitor	Thread
Thread Border Router	Gateway for connecting Thread	Thread Border Router	Thread Border Router	Thread

Tool	Description	Download	Documentation	Protocol
	network to the Internet			

See also Nordic mobile apps for a list of available Bluetooth Low Energy and Mesh mobile apps for iOS, Android, and Windows Phones.

7.2 Setting up the nRF5 SDK

The nRF5 SDK does not require installation. You only need to download and extract the files.

If you followed the instructions in Running a first test on page 9, you already downloaded and extracted the nRF5 SDK files and are all set up.

To set up your SDK environment:

1. Download the nRF5 SDK zip file.

   If you have an nRF52 device, select the latest version. For nRF51 devices, select the latest version with support for nRF51 (currently, v12.3.0). For information about which SDK supports which IC revisions, check the compatibility matrices.

2. Extract the zip file to the directory that you want to use to work with the SDK.

   This folder will be referred to as *SDK_dir* in the following documentation.

> **Note:** Compilers tend to run into problems with long path names. Therefore, place the folder as close to the root level of your file system as possible (for example, at `C:/Nordic/SDK`). Also, avoid using spaces in the file path and folder name.

7.3 Installing SEGGER tools

Download and install the most recent releases of *SES* and the J-Link Software and Documentation Pack.

1. Download the software packages for your operating system from SEGGER downloads.

   You need the following packages:

   - Embedded Studio for ARM (version 3.30 or later)
   - J-Link Software and Documentation Pack (version 6.10g or later)

2. Install both packages.

3. Obtain and activate your free license for *SES*:

   a) Open *SES*.

      SES will automatically load a test project.

   b) Click **Build** > **Build and Debug**.

      A window asking for a license will pop up. *SES* is free of charge for use with Nordic Semiconductor devices, but you still need to request and activate a license.

   c) Select **Activate Your Free License** and fill in your information to request a license.

   > **Note:** In *SES* versions before 3.34, this option was called **Get a Free License**.

      The license is sent to you in an email.

   d) After you receive your license key, enter it to activate the license.

7.4 Installing the nRF5x Command Line Tools

The nRF5x Command Line Tools are used for developing, programming, and debugging of Nordic Semiconductor's nRF5x SoCs (System on Chip).

When installing on macOS or Linux, the SEGGER software must be installed in its default location, or the shared library must be placed so that `dlopen()` can find it. The default location is:

- On macOS: `/Applications/SEGGER/JLink`
- On Linux: `/opt/SEGGER/JLink`

The SEGGER software can be installed by downloading and running the installer from SEGGER Software.

When installing the nRF5x Command Line Tools on Windows, SEGGER software is automatically installed in addition to the tools.

Complete the following steps to install the nRF5x Command Line Tools:

1. Download the software for your operating system:
   - Windows: nRF5x-Command-Line-Tools for Win32
   - Linux 32-bit: nRF5x-Command-Line-Tools-Linux-i386
   - Linux 64-bit: nRF5x-Command-Line-Tools-Linux-x86_64
   - macOS: nRF5x-Command-Line-Tools for Os X
2. Install the software by running the installer and following the given instructions, or by extracting the `.tar` archive anywhere on your file system and adding the extracted folders to your path.
3. Enter the following command in a command line to make sure that nrfjprog is installed correctly: `nrfjprog --version`

   If you get an error message that the command cannot be found, nrfjprog must be manually added to the PATH. On Windows:

   a) Go to the Windows **Advanced system settings** and click **Environment Variables**.
   b) Select the Path variable and click **Edit**.
   c) Add the following text at the end of the variable value: `;C:\Program Files (x86)\Nordic Semiconductor\nrf5x\bin`

      Make sure that you add a semicolon (;) between entries in the PATH values: `path1;path2`
   d) Click **OK** twice.

   On Linux, assuming that you have extracted the `.tar` archive into `/opt/nrfjprog`:

   a) Add the following command to the configuration file for your command line, for example, to `~/.bashrc`:

      ```
      export PATH=$PATH:/opt/nrfjprog
      ```

   Open a new command prompt and repeat the command. It should now succeed.

8 Programming an application

After setting up the required toolchain, you are ready to compile your application and program (or "flash") it to your development board.

Starting with v14.1.0, the nRF5 SDK supplies SEGGER Embedded Studio projects. If you are using an older version of the nRF5 SDK (for example, nRF5 SDK v12.3.0, which supports nRF51 Series devices), you must import and convert the Keil µVision projects.

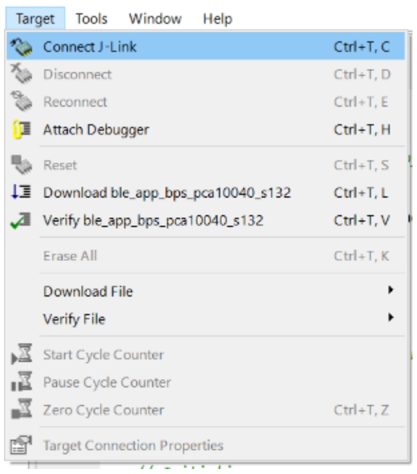
There is a series of video tutorials that show how to get started with SEGGER Embedded Studio in combination with the nRF5 SDK. Check them out here: Getting started with SEGGER Embedded Studio and the nRF5 SDK

8.1 Erasing the board

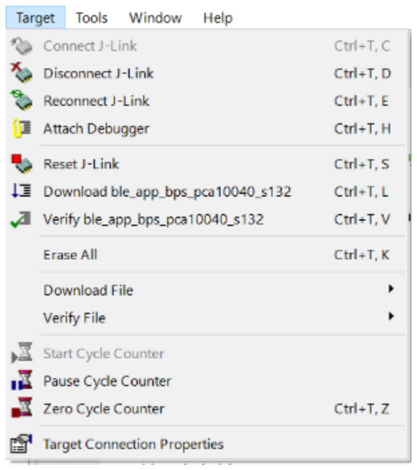
Before you program an example to the development board, you should erase the contents of the board.

There are different ways to erase the board. You can, for example, use *SES* or the command line tool nrfjprog (part of the nRF5x Command Line Tools).

- To erase the contents of the board with *SES*, complete the following steps:
  a) Select **Target** > **Connect J-Link**.



  b) After the connection is established, select **Target** > **Erase All**.



Target
Tools
Window
Help
Connect J-Link Ctrl+T, C
Disconnect J-Link Ctrl+T, D
Reconnect J-Link Ctrl+T, E
Attach Debugger Ctrl+T, H
Reset J-Link Ctrl+T, S
Download ble_app_bps_pca10040_s132 Ctrl+T, L
Verify ble_app_bps_pca10040_s132 Ctrl+T, V
Erase All Ctrl+T, K
Download File
Verify File
Start Cycle Counter
Pause Cycle Counter
Zero Cycle Counter Ctrl+T, Z
Target Connection Properties

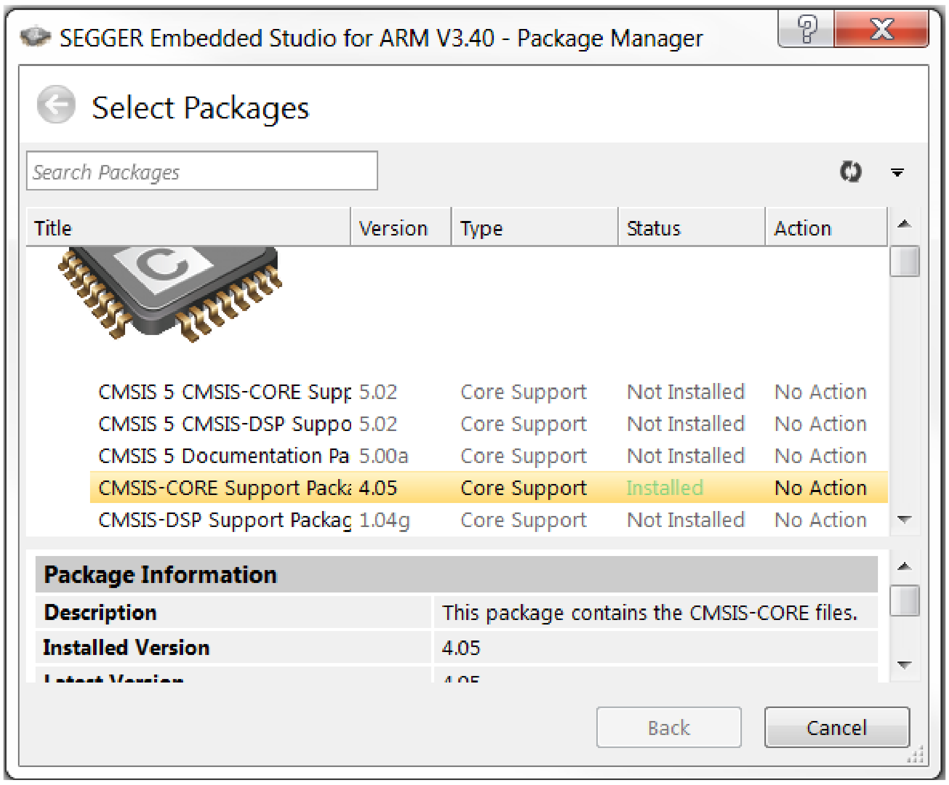
- To erase the contents of the board with nrfjprog, enter the following command:

- For nRF51 devices: `nrfjprog --family nRF51 --eraseall`
- For nRF52 devices: `nrfjprog --family nRF52 --eraseall`

8.2 Importing Keil projects

If you are using a version of the SDK that supports *SES*, thus nRF5 SDK v14.1.0 or later, you should open one of the supplied *SES* projects and skip this step. If you are using an older version of the SDK, for example, nRF5 SDK v12.3.0 for nRF51 Series devices, you must create your own *SES* projects based on the supplied Keil projects.

Before you begin, install the CMSIS-CORE Support Package. To do so, open *SES*, go to **Tools** > **Package Manager**, and select and install **CMSIS-CORE Support Package**.



Title	Version	Type	Status	Action
CMSIS 5 CMSIS-CORE Supp	5.02	Core Support	Not Installed	No Action
CMSIS 5 CMSIS-DSP Suppo	5.02	Core Support	Not Installed	No Action
CMSIS 5 Documentation Pa	5.00a	Core Support	Not Installed	No Action
CMSIS-CORE Support Packa	4.05	Core Support	Installed	No Action
CMSIS-DSP Support Packag	1.04g	Core Support	Not Installed	No Action

Package Information	
Description	This package contains the CMSIS-CORE files.
Installed Version	4.05
Latest Version	4.05

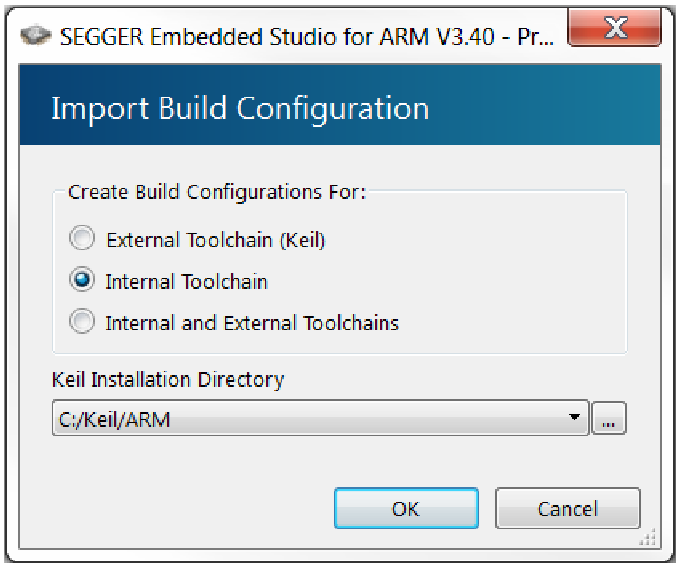
Figure 1: SES Package Manager

Complete the following steps to import a Keil project into *SES*:

1. Select **File** > **Import Project** > **Import Keil MDK Project**.

   In *SES* versions before 3.40, this option was located under **File** > **Import IAR EWARM / Keil MDK Project**.
2. Navigate to the folder that contains the nRF5 SDK and select the example that you want to import.

   Use the `*.uvprojx` file that is located in the `arm5_no_packs` folder.
3. In the **Import Build Configuration** window that appears, select **Internal Toolchain**.


SEGGER Embedded Studio for ARM V3.40 - Pr...

Import Build Configuration

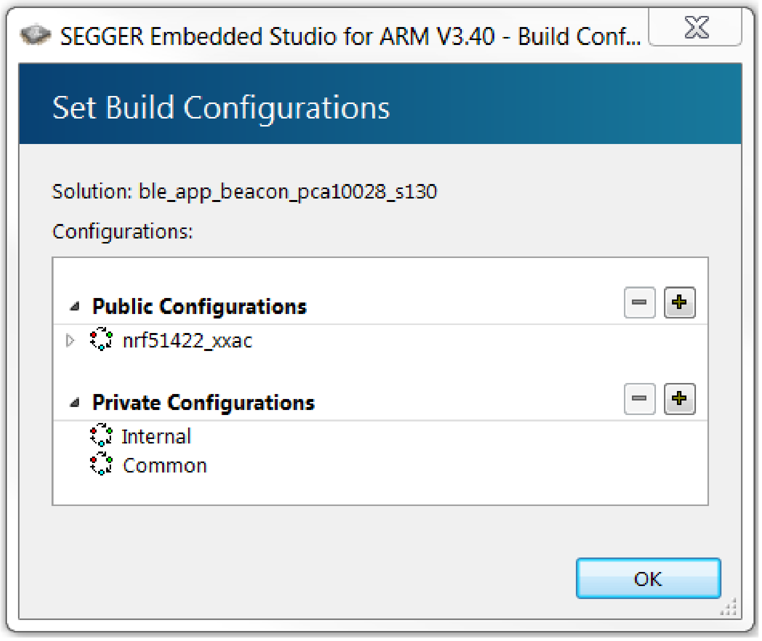
Create Build Configurations For:

- ○ External Toolchain (Keil)
- ● Internal Toolchain
- ○ Internal and External Toolchains

Keil Installation Directory: C:/Keil/ARM

4. If the project that you imported contains a *Target* for flashing the *SoftDevice*, delete this *Target* in *SES*:
   a) Select **Project** > **Build Configurations**.
   b) Under **Public Configurations**, select the *Target* (for example, **flash_s130_nrf51_2.0.1_softdevice**).
   c) Click the **-** symbol to remove the *Target*.

   The build configuration should now look similar to this:


SEGGER Embedded Studio for ARM V3.40 - Build Conf...

Set Build Configurations

Solution: ble_app_beacon_pca10028_s130

Configurations:

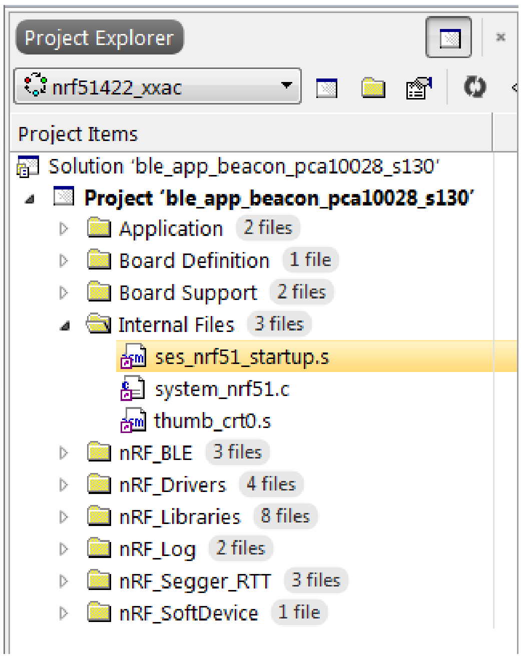
Public Configurations
- nrf51422_xxac

Private Configurations
- Internal
- Common

5. Add the nRF5 MDK files that are required for startup and system setup.

   In Keil µVision, these files are provided by the nRF_DeviceFamilyPack. Since this pack is not available for *SES*, you must add them manually to the *SES* project.
   a) Download ses_nrf51_startup.s and ses_nrf52_startup.s and save them in a new folder *SDK_dir*/components/toolchain/embedded_studio.
   b) In the Project Explorer, navigate to **Internal Files** and remove the `Cortex_M_Startup.s` file.
   c) Right-click **Internal Files** and select **Add Existing File**.
   d) Browse to the *SDK_dir*/components/toolchain/ folder and select the `.c` file that corresponds to your device (for example, `system_nrf52.c`).
   e) Right-click **Internal Files** and select **Add Existing File** again. This time, select the startup file that you downloaded, for example, *SDK_dir*/components/toolchain/embedded_studio/ses_nrf51_startup.s.

The Project Explorer should now look similar to this:



Solution 'ble_app_beacon_pca10028_s130'
Project 'ble_app_beacon_pca10028_s130'
Application 2 files
Board Definition 1 file
Board Support 2 files
Internal Files 3 files
ses_nrf51_startup.s
system_nrf51.c
thumb_crt0.s
nRF_BLE 3 files
nRF_Drivers 4 files
nRF_Libraries 8 files
nRF_Log 2 files
nRF_Segger_RTT 3 files
nRF_SoftDevice 1 file

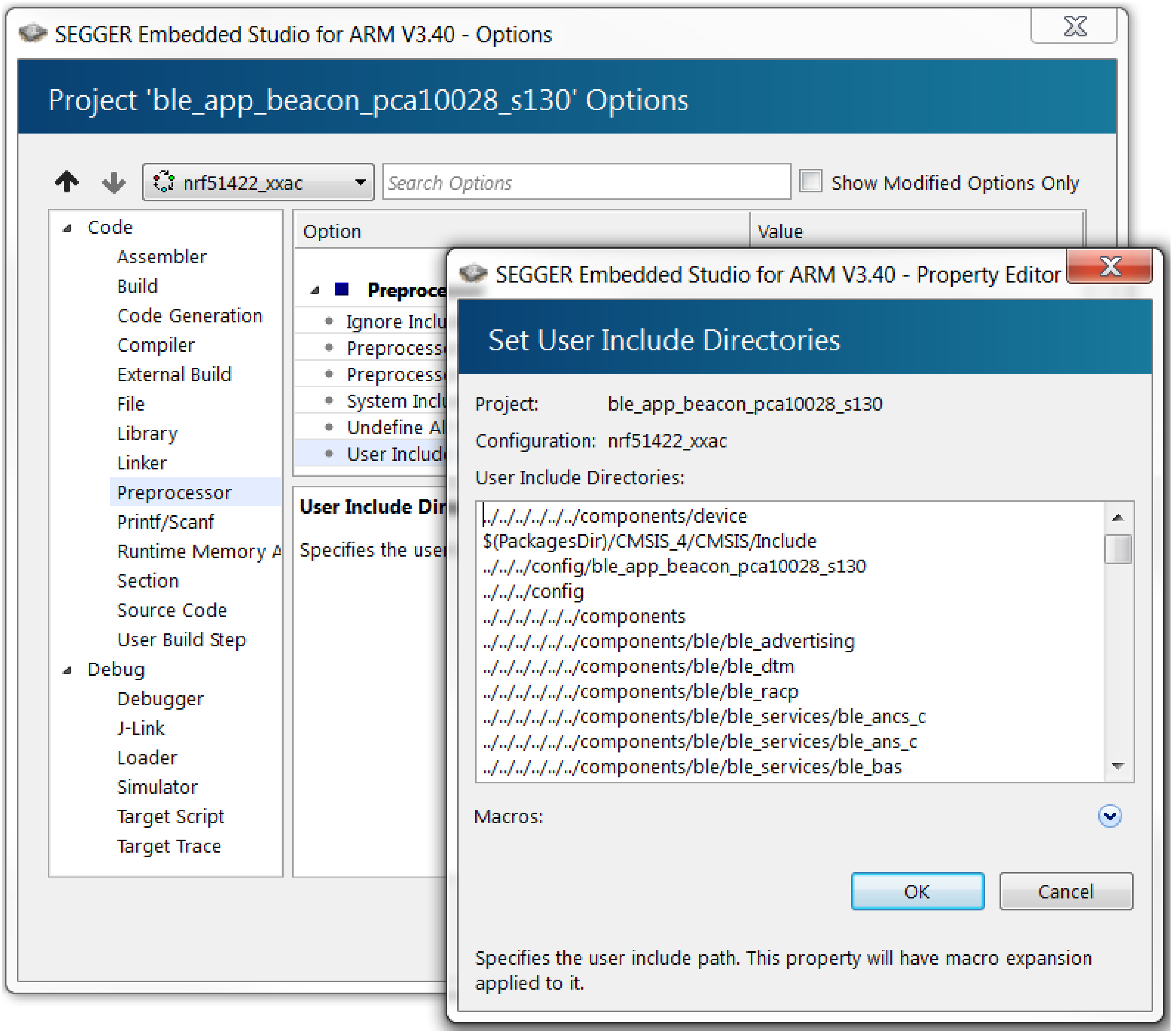
6. Include the device header files in the project:

a) Right-click your project and select **Edit Options**.

b) Select **Preprocessor**.

c) Add the following path to the User Include Directories field: `../../../../../../components/device`

The user include directories should now look similar to this:



Project: ble_app_beacon_pca10028_s130

Configuration: nrf51422_xxac

User Include Directories:

```
../../../../../../components/device
$(PackagesDir)/CMSIS_4/CMSIS/Include
../../../config/ble_app_beacon_pca10028_s130
../../../config
../../../../../../components
../../../../../../components/ble/ble_advertising
../../../../../../components/ble/ble_dtm
../../../../../../components/ble/ble_racp
../../../../../../components/ble/ble_services/ble_ancs_c
../../../../../../components/ble/ble_services/ble_ans_c
../../../../../../components/ble/ble_services/ble_bas
```

Specifies the user include path. This property will have macro expansion applied to it.

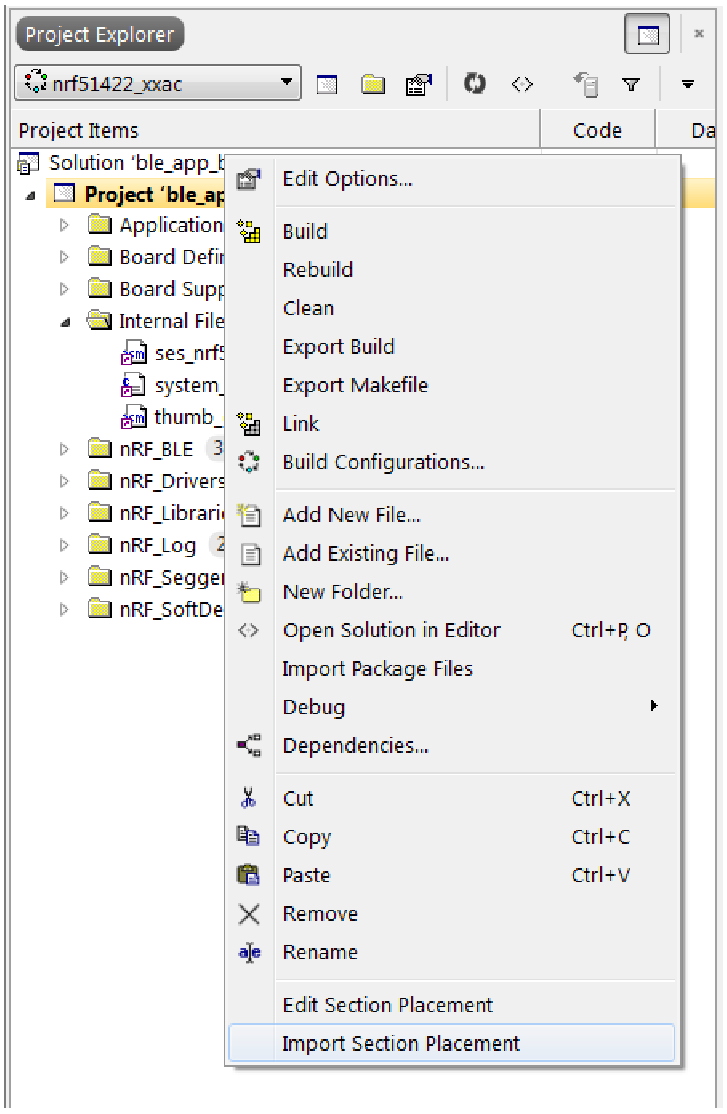
7. If your project uses modules that require section variables (for example, the Peer Manager, Flash Data Storage, or Flash Storage), define where in the flash information from these modules should be stored.

a) Download the `flash_placement.xml` and place it in your project directory.

Files are provided for the following versions of the nRF5 SDK:

- For nRF5 SDK v12.x.x: flash_placement.xml
- For nRF5 SDK v13.x.x and nRF5 SDK v14.0.0: flash_placement.xml

b) Right-click on your project in the Project Explorer and select **Import Section Placement**.



c) Confirm that you want to use the section placement for the current build configuration.

8.3 Compiling the application

You can compile the application from a *SES* project provided by the nRF5 SDK v14.2.0 or later, or from the project that you created based on an older SDK example.

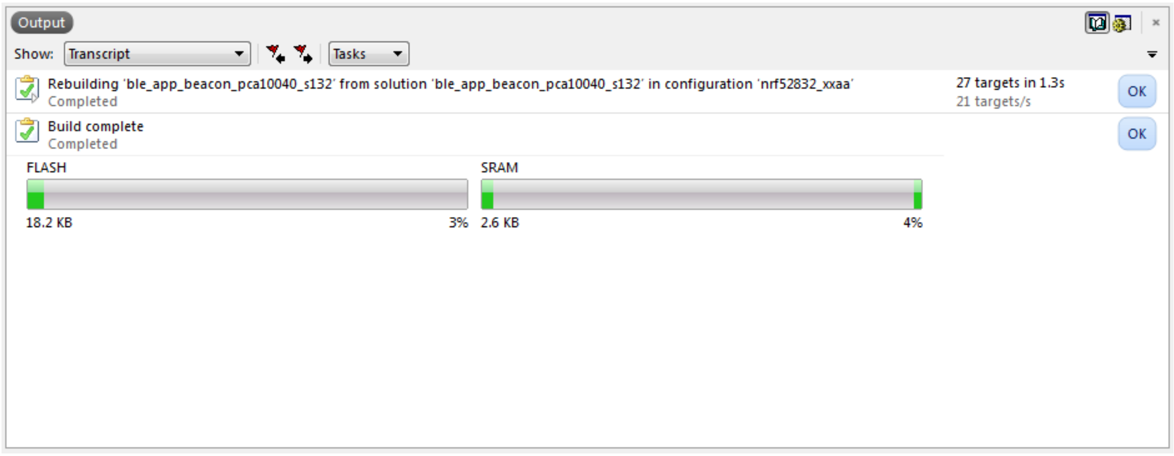
1. Open your project in *SES*.

   In nRF5 SDK v14.2.0 or later, *SES* projects are located in the `ses` subfolder of the example folder, for example, `SDK_dir/examples/ble_peripheral/ble_app_uart/pca10040/s132/ses`.

2. Select **Build** > **Build *project_target***.

   Alternatively, press `F7`. Make sure that there are no build errors.

   The output should look similar to this:



```
Rebuilding 'ble_app_beacon_pca10040_s132' from solution 'ble_app_beacon_pca10040_s132' in configuration 'nrf52832_xxaa'    27 targets in 1.3s    OK
Completed                                                                                                                  21 targets/s
Build complete                                                                                                                           OK
Completed
FLASH                          SRAM
18.2 KB                    3%  2.6 KB                    4%
```

8.4 Configuring placement of the SoftDevice

If your application uses Bluetooth or ANT, you must program a *SoftDevice* in addition to the application.

Note:

If your application does not use a *SoftDevice*, you can skip this step.

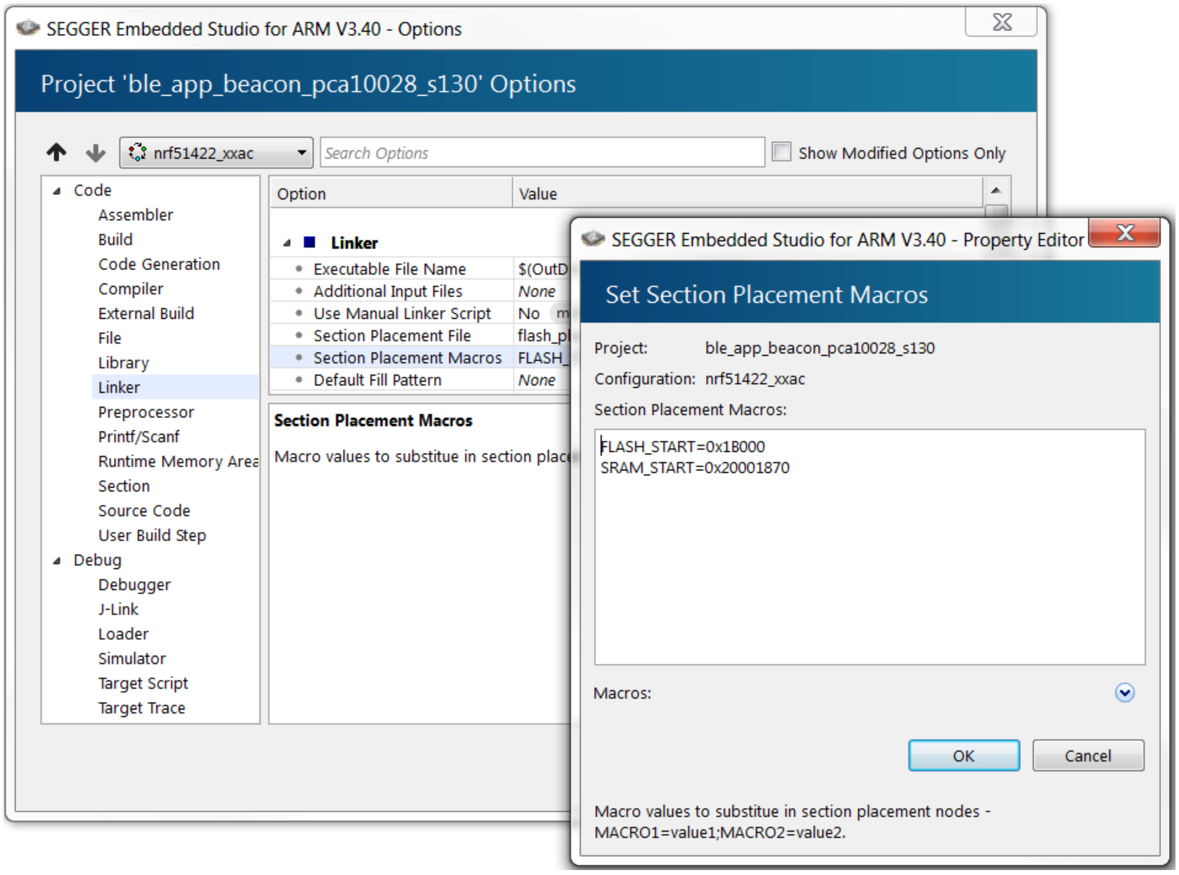
If you are using a *SES* project from nRF5 SDK v14.2.0 or later, the placement of the SoftDevice is already configured correctly and you can skip this step.

If your application requires a *SoftDevice*, the flash and SRAM position where the compiled binary will be placed must be configured as follows:

1. In *SES*, right-click your project and select **Edit Options**.
2. Select **Linker**.
3. In the Section Placement Macros field, add values for FLASH_START and SRAM_START.

   To find the correct values, check the Keil project that you imported (in Keil µVision, select **Options for Target** > **Target**), or program the firmware with approximate values, run it, and check the log output in the debug terminal for the recommended values.

   For example, when running the ble_app_uart example application from nRF5 SDK v12.3.0 on PCA10028 with SoftDevice S130 v2.0.1, specify the section placement macros as shown:



Project: ble_app_beacon_pca10028_s130

Configuration: nrf51422_xxac

Section Placement Macros:

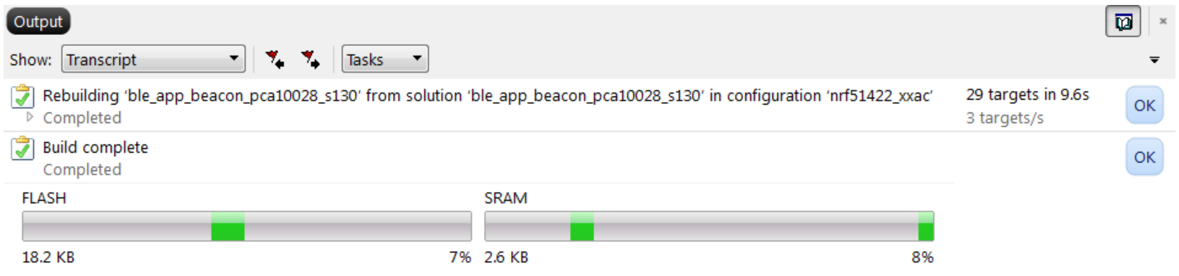
```
FLASH_START=0x1B000
SRAM_START=0x20001870
```

Macro values to substitue in section placement nodes - MACRO1=value1;MACRO2=value2.

4. Select **Build** > **Rebuild *project_target*** to rebuild the project.

   Alternatively, press `ALT + F7`.

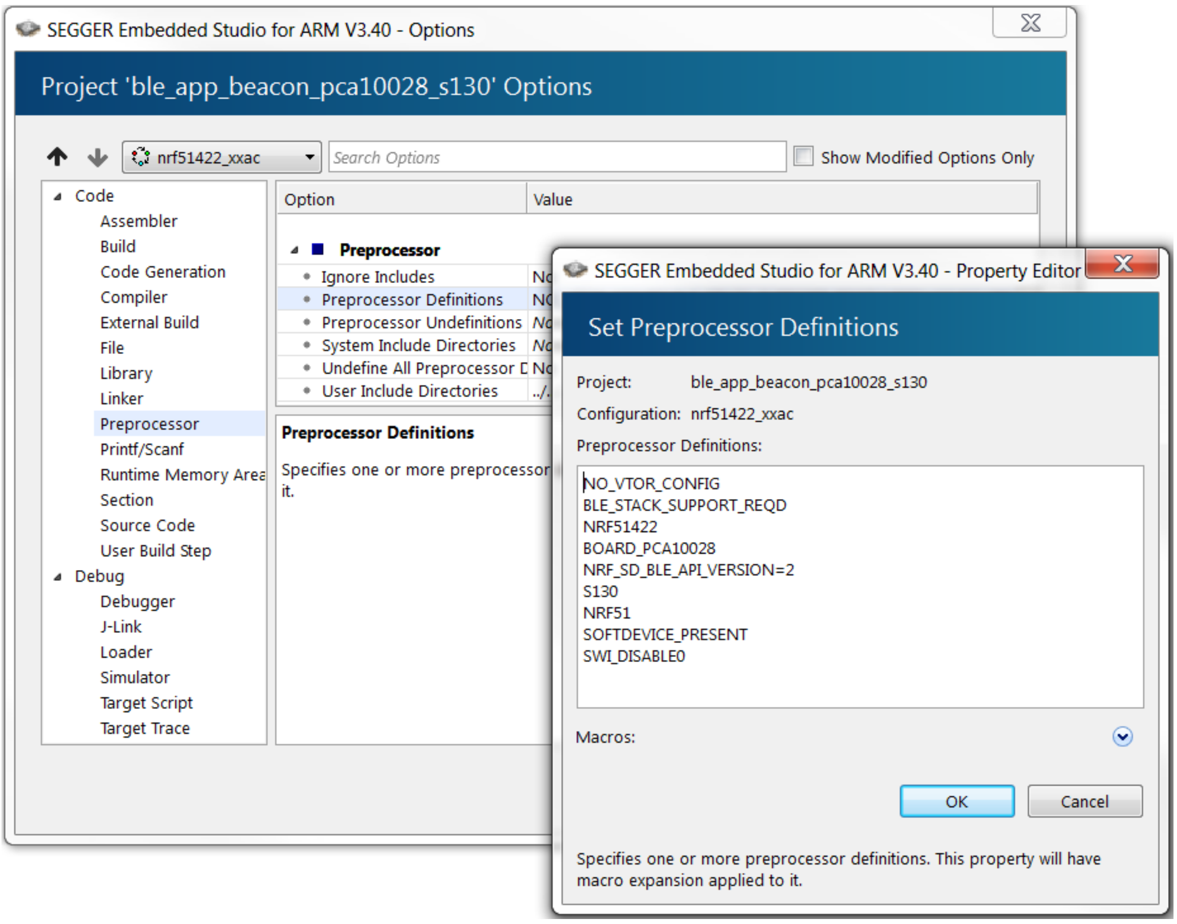
   The output should now look similar to this, with space reserved for the *SoftDevice*:



```
Rebuilding 'ble_app_beacon_pca10028_s130' from solution 'ble_app_beacon_pca10028_s130' in configuration 'nrf51422_xxac'    29 targets in 9.6s
Completed                                                                                                                    3 targets/s
Build complete
Completed
FLASH                                    SRAM
18.2 KB                              7%  2.6 KB                              8%
```

5. Right-click your project and select **Edit Options**.
6. Select **Preprocessor**.
7. Add the definition `NO_VTOR_CONFIG` to the Preprocessor Definitions.

   This definition tells *SES* to expect a *SoftDevice* to be present that will forward exceptions and interrupts to the application.



Preprocessor Definitions

Specifies one or more preprocessor … it.

Project: ble_app_beacon_pca10028_s130

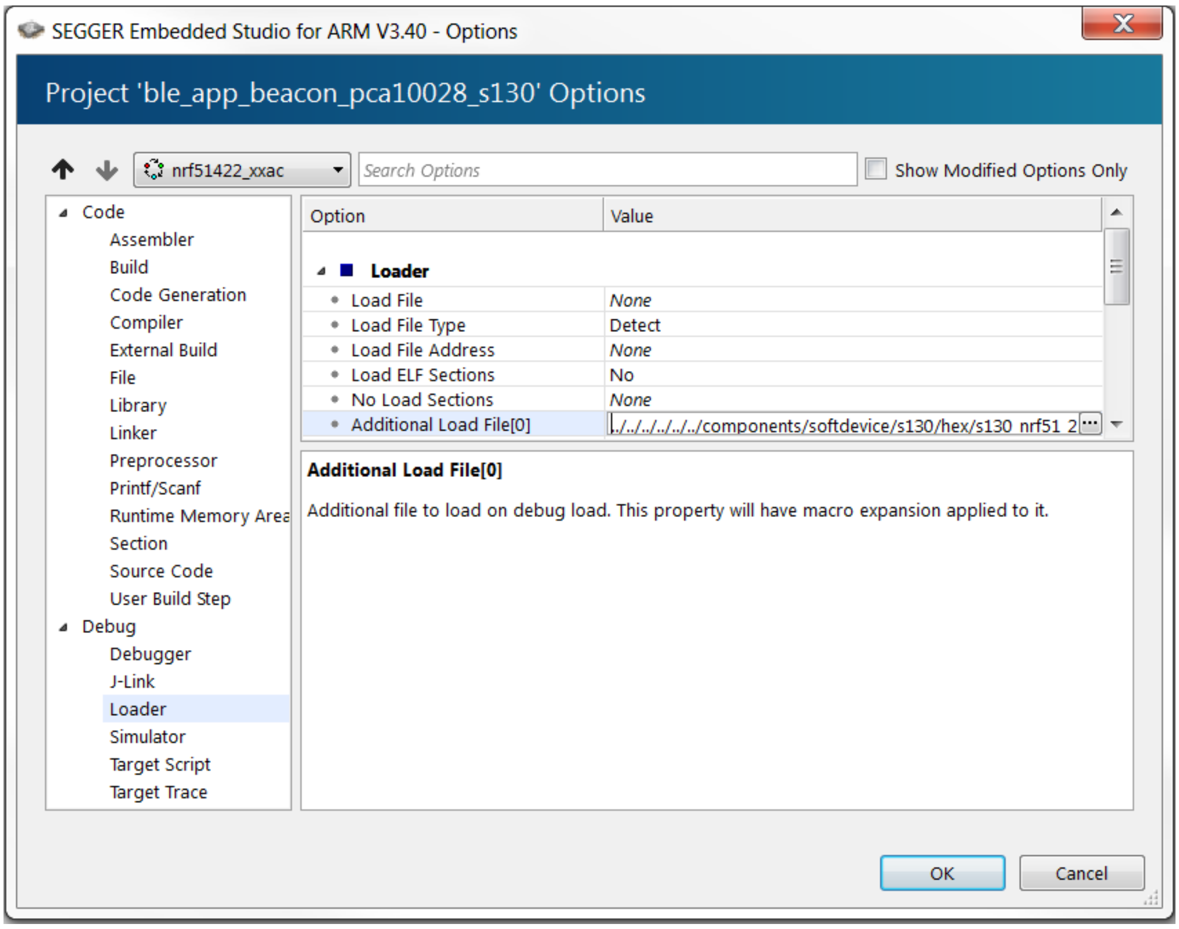
Configuration: nrf51422_xxac

Preprocessor Definitions:

```
NO_VTOR_CONFIG
BLE_STACK_SUPPORT_REQD
NRF51422
BOARD_PCA10028
NRF_SD_BLE_API_VERSION=2
S130
NRF51
SOFTDEVICE_PRESENT
SWI_DISABLE0
```

Specifies one or more preprocessor definitions. This property will have macro expansion applied to it.

8. In the Debug section of the project options, select **Loader**.
9. Add the absolute path to the *SoftDevice* to the Additional Load File[0] field, for example, `../../../../../../components/softdevice/s130/hex/s130_nrf51_2.0.1_softdevice.hex`.


SEGGER Embedded Studio for ARM V3.40 - Options

Project 'ble_app_beacon_pca10028_s130' Options

nrf51422_xxac

Search Options

Show Modified Options Only

- Code
  - Assembler
  - Build
  - Code Generation
  - Compiler
  - External Build
  - File
  - Library
  - Linker
  - Preprocessor
  - Printf/Scanf
  - Runtime Memory Area
  - Section
  - Source Code
  - User Build Step
- Debug
  - Debugger
  - J-Link
  - Loader
  - Simulator
  - Target Script
  - Target Trace

Option	Value
Loader	
Load File	*None*
Load File Type	Detect
Load File Address	*None*
Load ELF Sections	No
No Load Sections	*None*
Additional Load File[0]	../../../../../components/softdevice/s130/hex/s130_nrf51_2

Additional Load File[0]

Additional file to load on debug load. This property will have macro expansion applied to it.

OK Cancel

8.5 Programming the firmware

After compiling the application, you can program it to the development board. If you configured a *SoftDevice* to be used, it is programmed together with the application.

1. Connect the development board to your computer.
2. Select **Debug** > **Build and Run**.

   Alternatively, press `Ctrl + F5`.

8.6 Adding files

After compiling and programming an unmodified example to ensure that your toolchain is set up correctly, modify your project by adding files and libraries.

8.6.1 Adding source files

All source files that are part of the application you are developing must be added to the project.

You can add existing files or create files in the project directory.

- To add an existing file, right-click your project or any subfolder in the **Project Explorer** and select **Add Existing File**. Browse to the file that you want to import and open it.

  The original file is not copied into the project, but it is included from its original location. That means that any modifications that you do will apply to all projects that use this file.

- To create a file, right-click your project or any subfolder in the **Project Explorer** and select **Add New File**.

  By default, the file is created in the project directory.

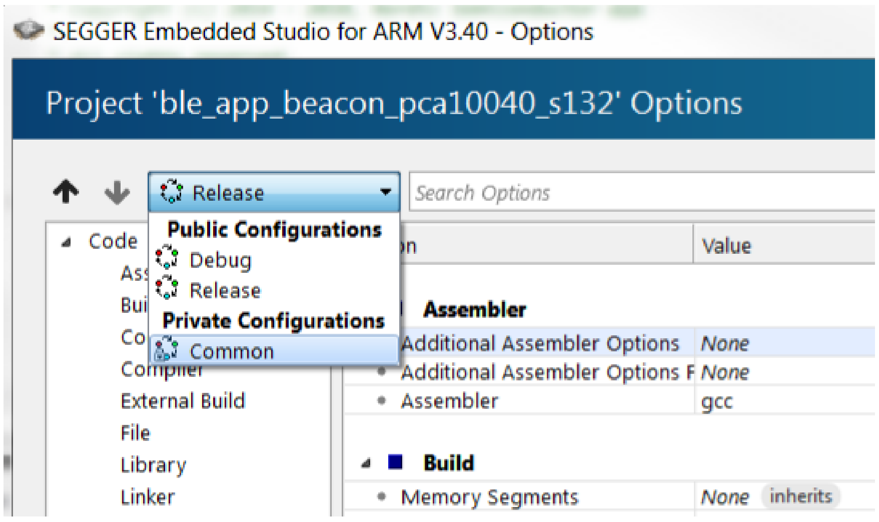
8.6.2 Including header files

Required header files must be linked to your project by adding their path to the user include directories.

Header files contain function declarations and macro definitions. You can request the use of header files by adding a `#include` preprocessing directive in your source files.

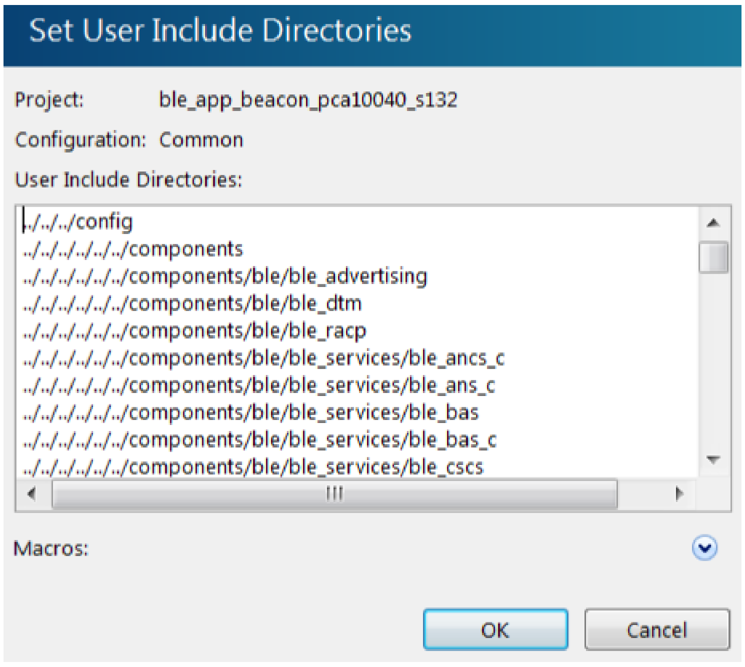
Header files are not linked to the project through the Project Explorer. To include a header file so that *SES* can find it, you must add its path to the list of directories in which *SES* looks for header files:

1. In the **Project Explorer**, right-click your project and select **Edit Options**.
2. In the **Project Options** window, select the **Common** configuration (sorted under **Private Configurations**).



3. Select **Preprocessor**.
4. Double-click **User Include Directories** and add the path to the folder that contains the header file.

   You can specify an absolute path or a path that is relative to the project directory. Using a relative path is preferable if you might want to move or copy your project to, for example, a new SDK version in the future.



Set User Include Directories

Project: ble_app_beacon_pca10040_s132

Configuration: Common

User Include Directories:

```
../../../config
../../../../../../components
../../../../../../components/ble/ble_advertising
../../../../../../components/ble/ble_dtm
../../../../../../components/ble/ble_racp
../../../../../../components/ble/ble_services/ble_ancs_c
../../../../../../components/ble/ble_services/ble_ans_c
../../../../../../components/ble/ble_services/ble_bas
../../../../../../components/ble/ble_services/ble_bas_c
../../../../../../components/ble/ble_services/ble_cscs
```

Macros:

5. Click **OK**.

9 Communicating with the board

Unless you programmed a very simple application, you probably want to connect to the board from your computer to display logging information or send input. You can use *Real Time Transfer (RTT)* or *Universal Asynchronous Receiver/Transmitter (UART)* for communicating with the board.

SEGGER Real Time Transfer (RTT) is a proprietary technology for bidirectional communication that supports J-Link devices and ARM-based microcontrollers. The advantage of using *RTT* is that it is very efficient and does not require any other peripheral than the J-Link debugging interface. *RTT* is directly supported in *SES*.

Connecting via *UART* is quick and power-efficient, but it requires dedicated use of the *UART* peripheral for logging. The nRF5 *DK*s and the nRF51 Dongle include a *UART* to USB CDC ACM bridge, which is needed to connect to the *UART*. Alternatively, you can use an external *UART* to USB bridge. We use the term CDC-UART to refer to *UART* communication through the *UART* to USB CDC ACM bridge, to distinguish it from communication through the Nordic UART Service (NUS) over Bluetooth Low Energy.

9.1 Connecting via RTT

SES directly supports *RTT*. You can see *RTT* output in the Debug Terminal during debugging sessions.

Alternatively, you can use J-Link RTT to view *RTT* output. See the following sections for instructions for Windows and Linux.

9.1.1 Connecting via RTT on Windows

To communicate via *RTT*, connect your development board via USB and run the J-Link RTT Viewer.

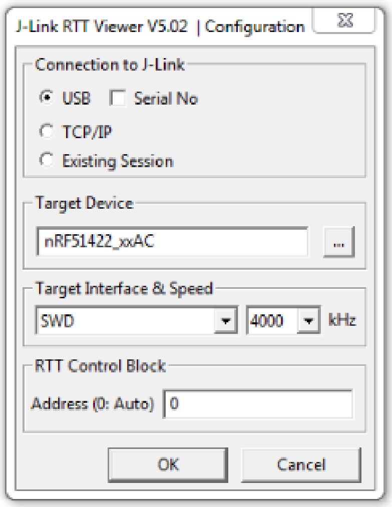
Note: *SES* natively supports *RTT*. If enabled, the monitor shows up when you start debugging. Alternatively, you can use SEGGER's J-Link RTT Viewer as described below.

The J-Link RTT Viewer is installed as part of the nRF5x Command Line Tools.

To run the J-Link RTT Viewer on Windows, complete the following steps:

1. Select the correct target device.

   The target device is represented by the ID of your development board.
2. Select **SWD** as the target interface.


J-Link RTT Viewer V5.02 | Configuration

Connection to J-Link: [✓ USB] Serial No, TCP/IP, Existing Session

Target Device: nRF51422_xxAC

Target Interface & Speed: SWD, 4000 kHz

RTT Control Block: Address (0: Auto) 0

9.1.2 Connecting via RTT on Linux

To communicate via *RTT*, connect your development board via USB and use SEGGER's J-Link RTT to establish a connection.

> **Note:** *SES* natively supports *RTT*. If enabled, the monitor shows up when you start debugging. Alternatively, you can use SEGGER's J-Link RTT Viewer as described below.

SEGGER's J-Link RTT is part of the J-Link Software and Documentation Pack, which is available from SEGGER downloads.

To use J-Link RTT on Linux, complete the following steps:

1. Enter `JLinkExe -if SWD` to set up the connection:

```
you@yourcomputer:~$ JLinkExe -if SWD
SEGGER J-Link Commander V5.10u (Compiled Mar 17 2016 19:06:22)
DLL version V5.10u, compiled Mar 17 2016 19:06:19

Connecting to J-Link via USB...O.K.
Firmware: J-Link OB-SAM3U128-V2-NordicSemi compiled Mar 15 2016 18:03:17
Hardware version: V1.00
VTref = 3.300V


Type "connect" to establish a target connection, '?' for help
J-Link>
```

2. Enter `connect` at the prompt to establish the connection.
   JLinkExe will ask for additional information. You can accept the default values.
3. From another terminal, start **`JLinkRTTClient`**.

RTT output is visible in the terminal that runs **`JLinkRTTClient`**.

9.2 Connecting via CDC-UART

To connect via CDC-UART, start a terminal emulator and connect to the used COM port.

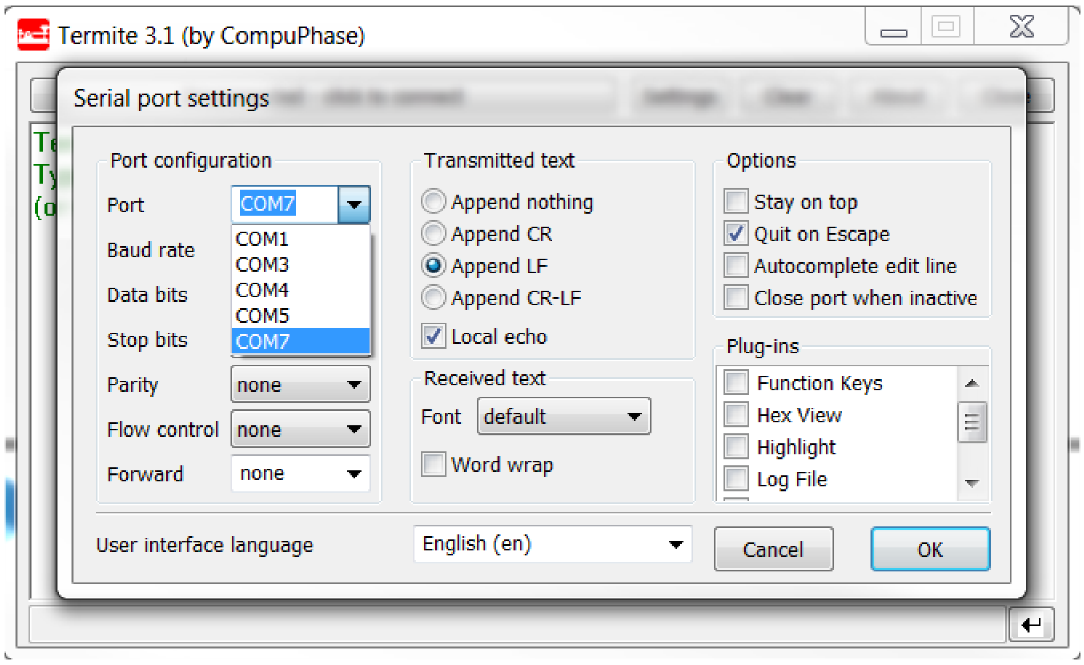
There is a wide variety of terminal emulators that you can use, for example, minicom or screen (both terminal-based, available for Linux), Termite (GUI-based, Windows only), or PuTTY (GUI-based, available for multiple operating systems).

When configuring the connection, use the following *UART* settings:

- Baud rate: 115.200
- 8 data bits
- 1 stop bit
- No parity
- HW flow control: RTS/CTS

The following instructions show how to configure Termite correctly. Other terminal emulators can be set up in a similar way.

1. Download and install the latest version of Termite.
2. Connect the development board to your computer.
3. Open Termite and click **Settings**.
   Depending on what devices you have connected to your computer, you might have several choices, as shown in the following figure:


Termite 3.1 (by CompuPhase)

Serial port settings

Port configuration
- Port: COM7 (COM1, COM3, COM4, COM5, COM7)
- Baud rate
- Data bits
- Stop bits
- Parity: none
- Flow control: none
- Forward: none

Transmitted text
- Append nothing
- Append CR
- ● Append LF
- Append CR-LF
- ✓ Local echo

Received text
- Font: default
- Word wrap

Options
- Stay on top
- ✓ Quit on Escape
- Autocomplete edit line
- Close port when inactive

Plug-ins
- Function Keys
- Hex View
- Highlight
- Log File

User interface language: English (en)

Cancel OK

4. Select the correct COM port to connect to the board.

To find the correct port, follow these steps:

a) Go to the start menu in Windows and type `devmgmt.msc` to open the Device Manager.

b) Scroll down and expand **Ports (COM & LPT)**.

c) Find the port named **JLink CDC UART Port** and note down the number in parentheses.

d) If you have more than one J-Link UART port, unplug the one that you want to use, plug it back in, and observe which one appeared last.

5. Configure the baud rate and the flow control. Use the default values for the rest of the settings (8 data bits, 1 stop bit, no parity).

By default, the SDK uses a baud rate of 115200 and RTS/CTS flow control.

6. Make sure that **Append LF** is selected.

This option appends a newline character to any text that is sent.

7. Configure the terminal to send an RTS (Ready To Send) signal to the development board:

a) Go to **Settings** > **Plug Ins**.

b) Enable **Status LEDs** and click **OK**.

c) Click on the dark green rectangle above RTS to set this signal high.
The text `Start...` is displayed in Termite.

Alternatively, you can turn off hardware flow control in your application.

10 Testing the application

The next step after compiling and programming your application is to test it. Nordic Semiconductor provides its own testing tool, nRF Connect, which is available both for mobile and for desktop.

10.1 Testing with a mobile device

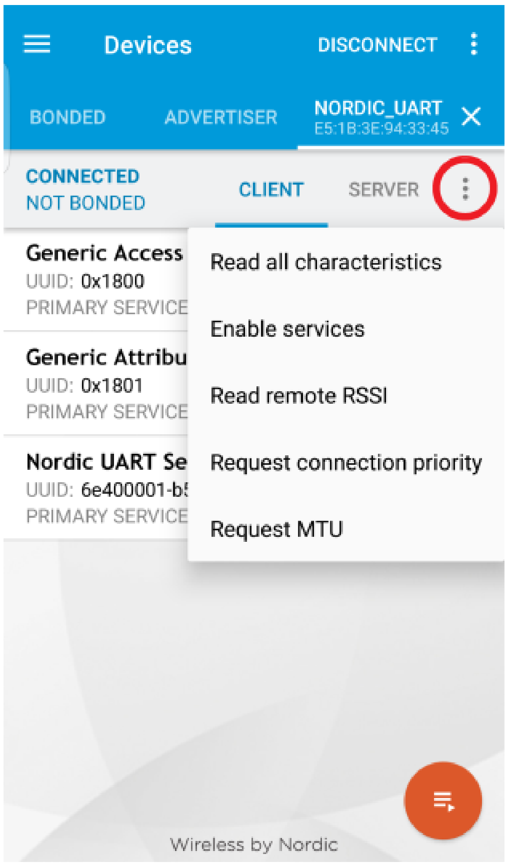
If you have a mobile device that supports Bluetooth Low Energy, download the nRF Connect app from Google Play or App Store to test your application.

The following procedure assumes that you have programmed the ble_app_uart example from the nRF5 SDK. Steps for testing other examples are similar. See the testing instructions for each example in the nRF5 SDK documentation for more information.

1. Download and install nRF Connect from Google Play or App Store.
2. Open nRF Connect.
3. Make sure that the *DK* is running the ble_app_uart example.

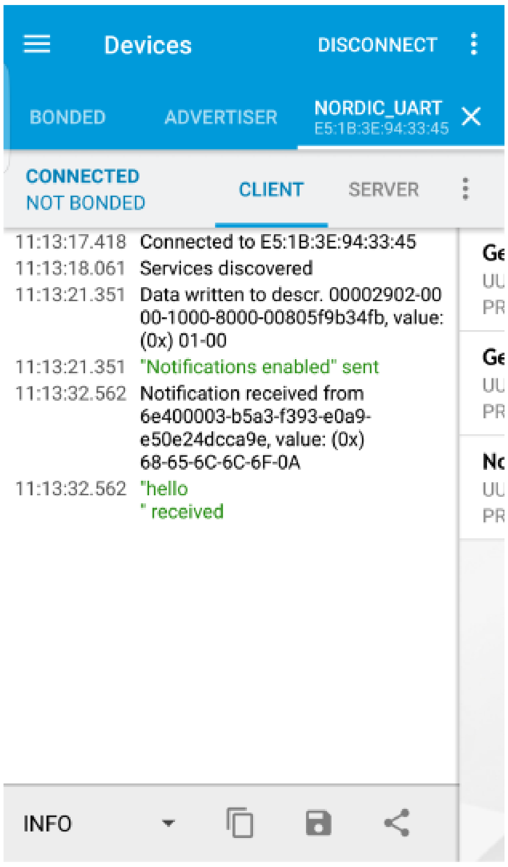
   LED1 should be blinking every 2 seconds, indicating that it is advertising.
4. Tap **Scan**.
5. Find the device and tap **Connect**.

   The default device name for the ble_app_uart example is "Nordic_UART".
6. When connected, tap the options button below the device name and select **Enable services**.



   This example communicates over Bluetooth Low Energy using the Nordic UART Service (NUS).
7. Tap the options button and select **Show log**.

8. In a terminal connected via CDC-UART, enter `hello` and send it to the *DK*.
   The text is sent through the *DK* to your device, which will display it in the nRF Connect log:



```
11:13:17.418  Connected to E5:1B:3E:94:33:45
11:13:18.061  Services discovered
11:13:21.351  Data written to descr. 00002902-00
              00-1000-8000-00805f9b34fb, value:
              (0x) 01-00
11:13:21.351  "Notifications enabled" sent
11:13:32.562  Notification received from
              6e400003-b5a3-f393-e0a9-
              e50e24dcca9e, value: (0x)
              68-65-6C-6C-6F-0A
11:13:32.562  "hello
              " received
```

10.2 Testing with a computer

If you have an nRF51 Dongle or a second *DK*, you can test your application with nRF Connect for Desktop. nRF Connect for Desktop is available for Windows, Linux, and macOS.

Note: This method requires an nRF5 *DK* or dongle to be connected to your computer.

The following procedure assumes that you have programmed the ble_app_uart example from the nRF5 SDK. Steps for testing other examples are similar. See the testing instructions for each example in the nRF5 SDK documentation for more information.

1. Download and install nRF Connect for Desktop.
2. Connect the dongle or the second *DK* to a USB port of your computer.
3. Connect to the board that runs the ble_app_uart example via CDC-UART.
4. Open nRF Connect for Desktop and add the Bluetooth Low Energy app.
5. Launch the Bluetooth Low Energy app.
6. Select the serial port for the dongle or the *DK* that is connected to your computer (not the board that runs the ble_app_uart example).

   If the device has not been used with the nRF Connect Bluetooth Low Energy app before, you may be asked to update the J-Link firmware and connectivity firmware for the device. You need to have the correct connectivity firmware on the nRF SoC to continue. When the nRF SoC has been programmed with the correct firmware, the nRF Connect Bluetooth Low Energy app proceeds to connect to it over USB. When the connection is established, the device appears in the main view.
7. Click **Start scan**.

8. Find the device and click **Connect**.

   The default device name for the ble_app_uart example is "Nordic_UART".
9. Select the *UART* RX characteristic value.
10. Write `30 31 32 33 34 35 36 37 38 39` (the hexadecimal value for the string "0123456789") and click **write**.
    The text "0123456789" is displayed in the terminal that is connected to the board via *UART*.
11. Enter any text, for example, `Hello`, in the terminal.
    In nRF Connect, the *UART* TX characteristic value changes to the corresponding ASCII value. For example, for `Hello`, the value is `48 65 6C 6C 6F`.

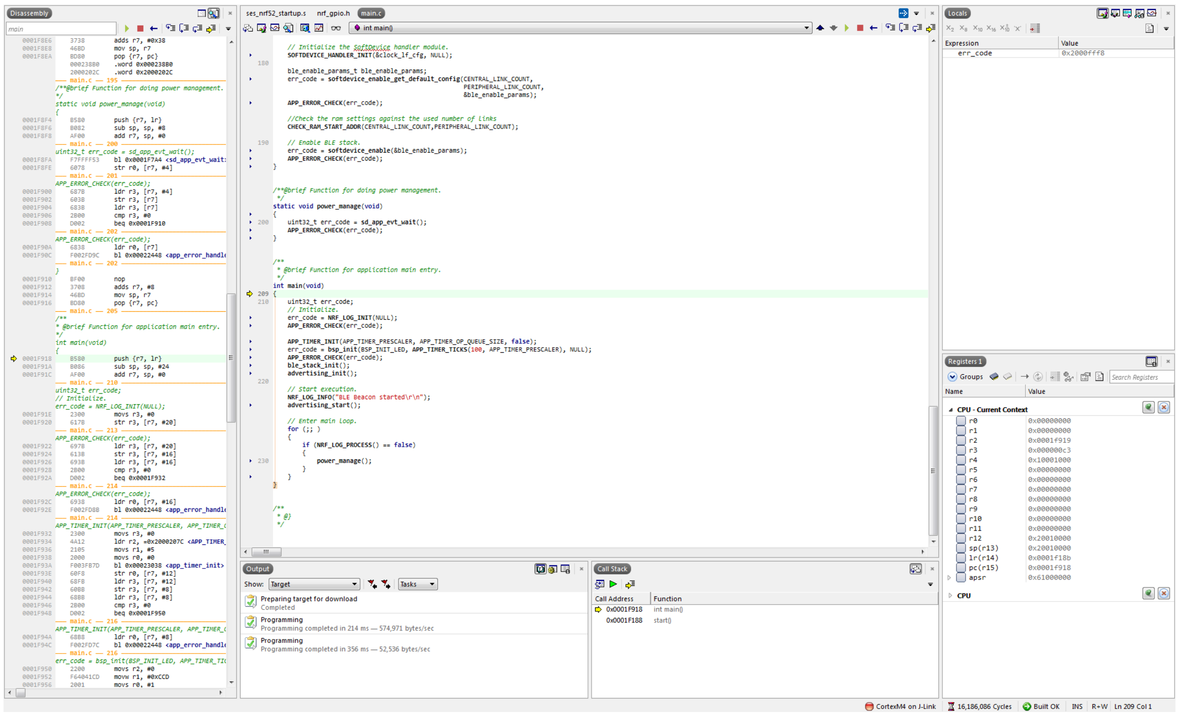
11 Debugging

To actually see what is happening on the development board while the application is running, you must set up a J-Link debugging session. *SES* has an integrated debugger that you can use to step through your application.

1. Open your project in *SES*.
2. Select **Debug** > **Go**.

   Alternatively, press `F5`.

   The debugging interface looks like this:

   Disassembly

   ```
   0001F8E6    3738        adds r7, #0x38
   0001F8E8    46BD        mov sp, r7
   0001F8EA    BD80        pop {r7, pc}
               000238B0    .word 0x000238B0
               2000202C    .word 0x2000202C
   —— main.c —— 195 ——————————————————
   /**@brief Function for doing power management.
   */
   static void power_manage(void)
   {
   0001F8F4    B580        push {r7, lr}
   0001F8F6    B082        sub sp, sp, #8
   0001F8F8    AF00        add r7, sp, #0
   —— main.c —— 200 ——————————————————
   uint32_t err_code = sd_app_evt_wait();
   0001F8FA    F7FFFF53    bl 0x0001F7A4 <sd_app_evt_wait
   0001F8FE    6078        str r0, [r7, #4]
   —— main.c —— 201 ——————————————————
   APP_ERROR_CHECK(err_code);
   0001F900    687B        ldr r3, [r7, #4]
   0001F902    603B        str r3, [r7]
   0001F904    683B        ldr r3, [r7]
   0001F906    2B00        cmp r3, #0
   0001F908    D002        beq 0x0001F910
   —— main.c —— 202 ——————————————————
   APP_ERROR_CHECK(err_code);
   0001F90A    6838        ldr r0, [r7]
   0001F90C    F002FD9C    bl 0x00022448 <app_error_handle
   —— main.c —— 202 ——————————————————
   }
   0001F910    BF00        nop
   0001F912    3708        adds r7, #8
   0001F914    46BD        mov sp, r7
   0001F916    BD80        pop {r7, pc}
   —— main.c —— 205 ——————————————————
   /**
   * @brief Function for application main entry.
   */
   int main(void)
   {
   0001F918    B580        push {r7, lr}
   0001F91A    B086        sub sp, sp, #24
   0001F91C    AF00        add r7, sp, #0
   —— main.c —— 210 ——————————————————
   uint32_t err_code;
   // Initialize.
   err_code = NRF_LOG_INIT(NULL);
   0001F91E    2300        movs r3, #0
   0001F920    617B        str r3, [r7, #20]
   —— main.c —— 213 ——————————————————
   APP_ERROR_CHECK(err_code);
   0001F922    697B        ldr r3, [r7, #20]
   0001F924    613B        str r3, [r7, #16]
   0001F926    693B        ldr r3, [r7, #16]
   0001F928    2B00        cmp r3, #0
   0001F92A    D002        beq 0x0001F932
   —— main.c —— 214 ——————————————————
   APP_ERROR_CHECK(err_code);
   0001F92C    6938        ldr r0, [r7, #16]
   0001F92E    F002FD8B    bl 0x00022448 <app_error_handle
   —— main.c —— 214 ——————————————————
   APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_(
   0001F932    2300        movs r3, #0
   0001F934    4A12        ldr r2, =0x2000207C <APP_TIMER_
   0001F936    2105        movs r1, #5
   0001F938    2000        movs r0, #0
   0001F93A    F003FB7D    bl 0x00023038 <app_timer_init>
   0001F93E    60F8        str r0, [r7, #12]
   0001F940    68FB        ldr r3, [r7, #12]
   0001F942    60BB        str r3, [r7, #8]
   0001F944    68BB        ldr r3, [r7, #8]
   0001F946    2B00        cmp r3, #0
   0001F948    D002        beq 0x0001F950
   —— main.c —— 216 ——————————————————
   APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_(
   0001F94A    68B8        ldr r0, [r7, #8]
   0001F94C    F002FD7C    bl 0x00022448 <app_error_handle
   —— main.c —— 216 ——————————————————
   err_code = bsp_init(BSP_INIT_LED, APP_TIMER_TI(
   0001F950    2200        movs r2, #0
   0001F952    F64041CD    movw r1, #0xCCD
   0001F956    2001        movs r0, #1
   ```

   ses_nrf52_startup.s   nrf_gpio.h   main.c   int main()

   ```c
       // Initialize the SoftDevice handler module.
       SOFTDEVICE_HANDLER_INIT(&clock_lf_cfg, NULL);

       ble_enable_params_t ble_enable_params;
       err_code = softdevice_enable_get_default_config(CENTRAL_LINK_COUNT,
                                                       PERIPHERAL_LINK_COUNT,
                                                       &ble_enable_params);
       APP_ERROR_CHECK(err_code);

       //Check the ram settings against the used number of links
       CHECK_RAM_START_ADDR(CENTRAL_LINK_COUNT,PERIPHERAL_LINK_COUNT);

       // Enable BLE stack.
       err_code = softdevice_enable(&ble_enable_params);
       APP_ERROR_CHECK(err_code);
   }


   /**@brief Function for doing power management.
    */
   static void power_manage(void)
   {
       uint32_t err_code = sd_app_evt_wait();
       APP_ERROR_CHECK(err_code);
   }


   /**
    * @brief Function for application main entry.
    */
   int main(void)
   {
       uint32_t err_code;
       // Initialize.
       err_code = NRF_LOG_INIT(NULL);
       APP_ERROR_CHECK(err_code);

       APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_OP_QUEUE_SIZE, false);
       err_code = bsp_init(BSP_INIT_LED, APP_TIMER_TICKS(100, APP_TIMER_PRESCALER), NULL);
       APP_ERROR_CHECK(err_code);
       ble_stack_init();
       advertising_init();

       // Start execution.
       NRF_LOG_INFO("BLE Beacon started\r\n");
       advertising_start();

       // Enter main loop.
       for (;; )
       {
           if (NRF_LOG_PROCESS() == false)
           {
               power_manage();
           }
       }
   }


   /**
    * @}
    */
   ```

   Output — Show: Target, Tasks

   ```
   Preparing target for download
   Completed
   Programming
   Programming completed in 214 ms — 574,971 bytes/sec
   Programming
   Programming completed in 356 ms — 52,536 bytes/sec
   ```

   Call Stack

Call Address	Function
0x0001F918	int main()
0x0001F188	start()

   Locals

Expression	Value
err_code	0x2000fff8

   Registers 1

Name	Value
CPU - Current Context	
r0	0x00000000
r1	0x00000000
r2	0x0001f919
r3	0x000000c3
r4	0x10001000
r5	0x00000000
r6	0x00000000
r7	0x00000000
r8	0x00000000
r9	0x00000000
r10	0x00000000
r11	0x00000000
r12	0x20010000
sp(r13)	0x20010000
lr(r14)	0x0001f18b
pc(r15)	0x0001f918
apsr	0x61000000
CPU	

   CortexM4 on J-Link   16,186,086 Cycles   Built OK   INS   R+W   Ln 209 Col 1

   By default, the application will break in main. You can set additional break points, single-step through the application, read registers, and so on.

This video tutorial shows you how to use *SES* for debugging:

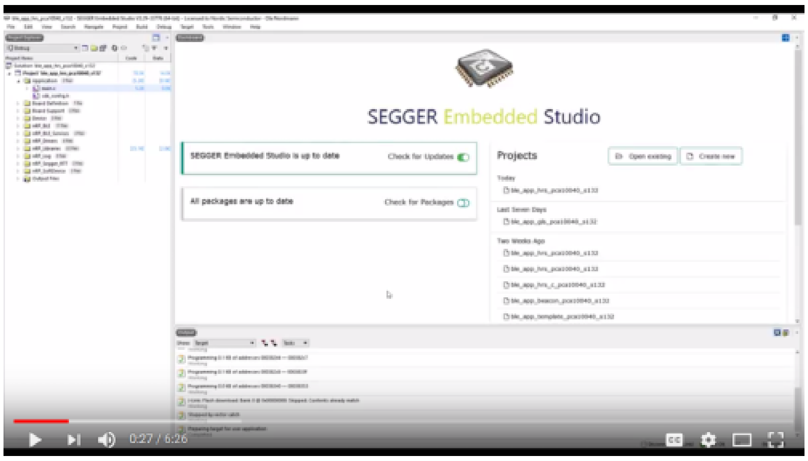


Figure 2: YouTube Tutorial

Glossary

Development Kit (DK)

A development platform used for application development.

GNU Compiler Collection (GCC)

A compiler system that supports various programming languages, maintained by the GNU Project.

Integrated Development Environment (IDE)

A software application that provides facilities for software development.

Real Time Transfer (RTT)

A proprietary technology for bidirectional communication that supports J-Link devices and ARM-based microcontrollers, developed by SEGGER Microcontroller.

SEGGER Embedded Studio (SES)

A cross-platform *IDE* for embedded C/C++ programming with support for Nordic Semiconductor devices, produced by SEGGER Microcontroller.

SoftDevice

A wireless protocol stack that complements the nRF5 Series SoCs. Nordic Semiconductor provides these stacks as qualified, precompiled binary files.

System on Chip (SoC)

A microchip that integrates all the necessary electronic circuits and components of a computer or other electronic systems on a single integrated circuit.

Target

The goal of an operation, for example, programming a specific image on a device, compiling a specific set of files, or removing previously generated files.

Universal Asynchronous Receiver/Transmitter (UART)

A hardware device for asynchronous serial communication between devices.

Acronyms and abbreviations

These acronyms and abbreviations are used in this document.

DK
Development Kit

GCC
GNU Compiler Collection

IDE
Integrated Development Environment

RTT
SEGGER Real Time Transfer

SES
SEGGER Embedded Studio

SoC
System on Chip

UART
Universal Asynchronous Receiver/Transmitter

Legal notices