Creating Applications with the Keil™ C51 C Compiler

nAN-15

Application Note v1.1

Key words

- Development using nRFgo SDK
- Setup of Keil compiler
- Getting started with your first application
- Debugging your application with Keil compiler
- Developing with nRF devices

2011-06-29

Contents

1 Introduction

This application note will help you start developing with an nRF device using the nRFgo development platform. This application note contains a step-by-step guide on how to set up your first project in a Keil compiler and explains how to debug using nRFprobe.

The target device for this tutorial is the nRF24LE1, which is a single chip solution for wireless applications. The nRF24LE1 features an ultra low power nRF24L01+ 2.4 GHz transceiver core with an 8051 flash microcontroller, ADC and rich set of digital interfaces.

This tutorial can also be used as reference when developing with other nRF devices with an 8051 microcontroller. In such cases you will need to switch to different nRF devices as shown in Figure 4. on page 7. The setup of the compiler, programming the device and debugger are all carried out in the same way as for the nRF24LE1.

2 Theory of operation

The nRFgo Software Development Kit (SDK) with the Keil C51 C compiler provides a flexible and simple way to develop using an nRF device. By following this guide you will acquire basic knowledge about the Keil compiler and nRFgo SDK so you can create, debug and program your first application.

2.1 Prerequisites to the tutorial

We recommend that you consult the documentation for the nRFgo SDK, nRFprobe and user documentation for the nRFgo Starter Kit while you carry out related development work.

Necessary software and hardware for this application note are listed below in the order you should install them:

Software

- Keil C51 (Version 9.00 or newer)
- nRFgo Studio (Version 1.4 or newer)
- nRFgo SDK (Version 2.2 or newer)
- nRFprobe (Version 1.2.0.5585 or newer)

Hardware

- nRFgo Starter Kit (nRF6700)
- nRFgo Development Kit for nRF24LE1 (nRF24LE1-F16Qxx-DK)

The following steps show how you install the software:

1. All nRFgo software is included in the nRFgo kits, but you must download the Keil compiler separately from www.keil.com. For the purposes of this application note the evaluation version of the Keil compiler is sufficient.
2. Make sure to download and install the latest software before you follow the instructions in this application note.
3. After installing all the software listed above, you must connect the nRFgo Motherboards to the computer and wait for the driver to install.
4. Thereafter, start up nRFgo Studio and update the firmware if prompted. Exit the nRFgo Studio application before you continue.
5. You can use nRFgo Studio to program nRFgo modules or your own prototypes through the ISP header. For development, use the Keil compiler along with the nRFprobe debugger.

2.2 Organizing the nRFgo SDK

The nRFgo SDK is built up in a tree structure. Here is a short description of the most important folders in the tree structure:

\Docs

This is the documentation for the SDK. It is mandatory reading in order to understand how a project is built up. It contains a description of the functionality for the software modules included in the SDK and of how they interact.

\Precompiled files

This folder includes the precompiled HEX files for the example projects. If you are using the evaluation version of the Keil compiler you will not be able to compile all the projects due to a code size limit, but you can still run the examples by programming the HEX files with nRFgo Studio.

\Source code\gazell

Gazell is a link and pairing library for wireless applications.

\Source code\hal

The Hardware Abstraction Layer (HAL) contains general interface functions to quickly start developing with the hardware modules embedded in an nRF chip. You will use some or all of these files for any project you develop. The path to these files needs to be included in the project.

\Source code\lib

The libraries interface with the HAL to provide more specific functionality. An example of this is the RF test library that can be used to set up different RF test modes.

\Source code\projects

The project folders contain the example projects included in the SDK. This is where you normally will put your own project.

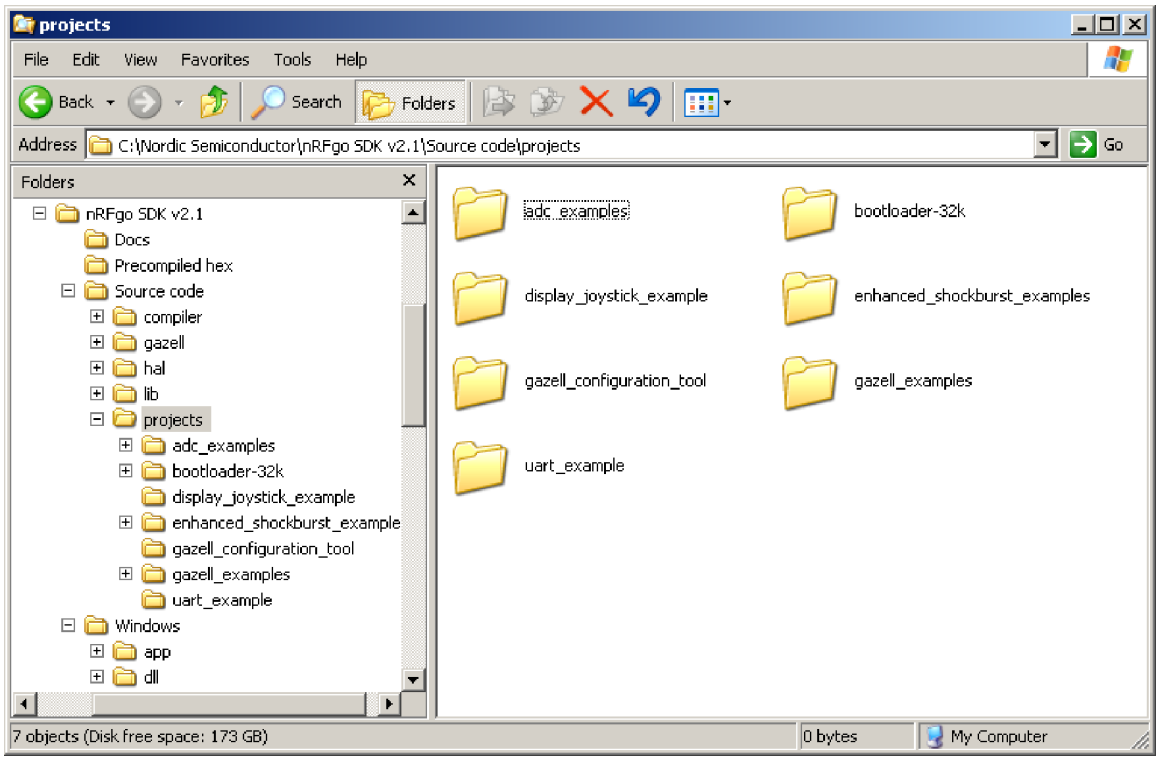


Figure 1. Folder structure for nRFgo SDK

3 Developing with nRFgo SDK

You can break down the development process for nRFgo SDK into four steps. These steps, listed below, range from the setup of the Keil compiler to the debugging of your first project:

1. Set up your first Keil project
2. Your first application
3. Including files
4. Debug your project

3.1 Step 1: Set up your first Keil project

Before writing any lines of code you need to create a new project in which you configure the Keil compiler to use the specific nRF device and include the paths that you need in the application.

1. Startup Keil uVision from the **Start** menu in Windows.
2. Thereafter, select **Project, New uVision Project** from Keil's menu**.**
3. You are prompted to specify where you want to save your new project. In this case you want to make a new project in *Source code\projects* and you should call the project "my_first_project". (see project folder encircled in Figure 2.)

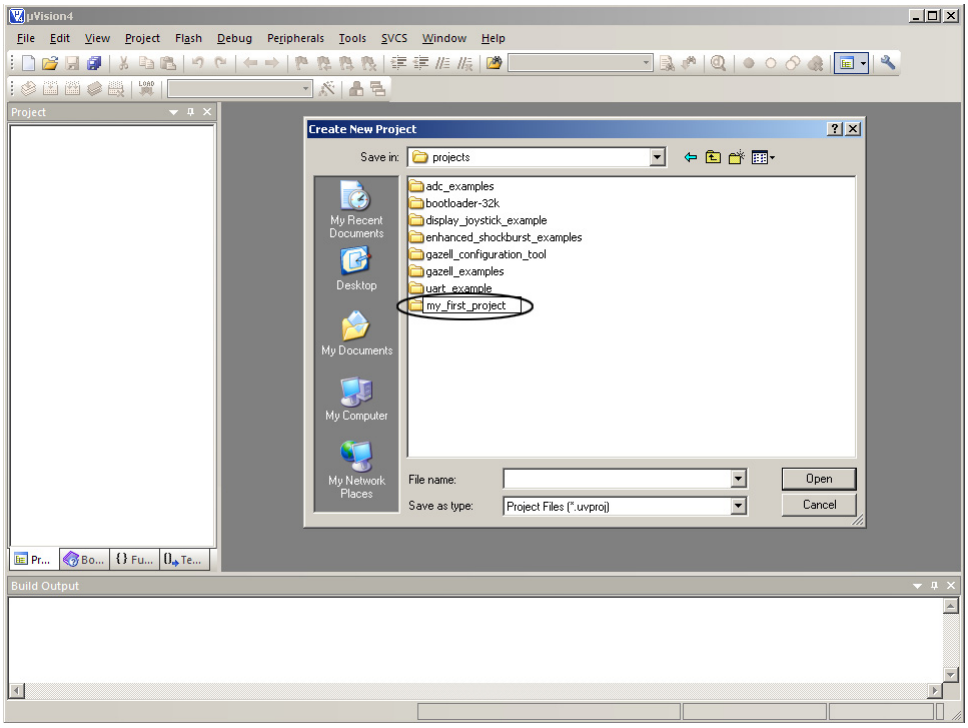


Figure 2. Create a new project

4. Create a new folder in the *Source code\projects* called *"my_first_project"* and then select **Save**. See Figure 3.

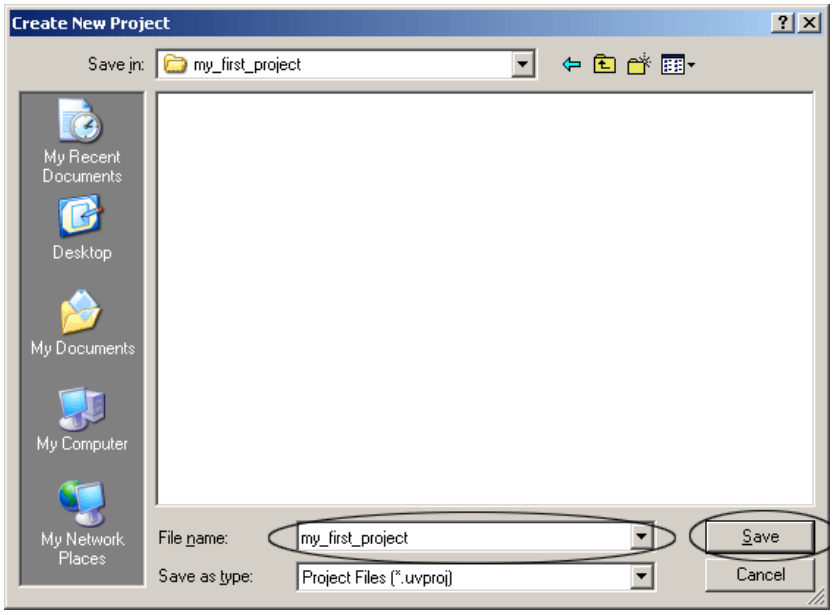


Figure 3. Naming the project

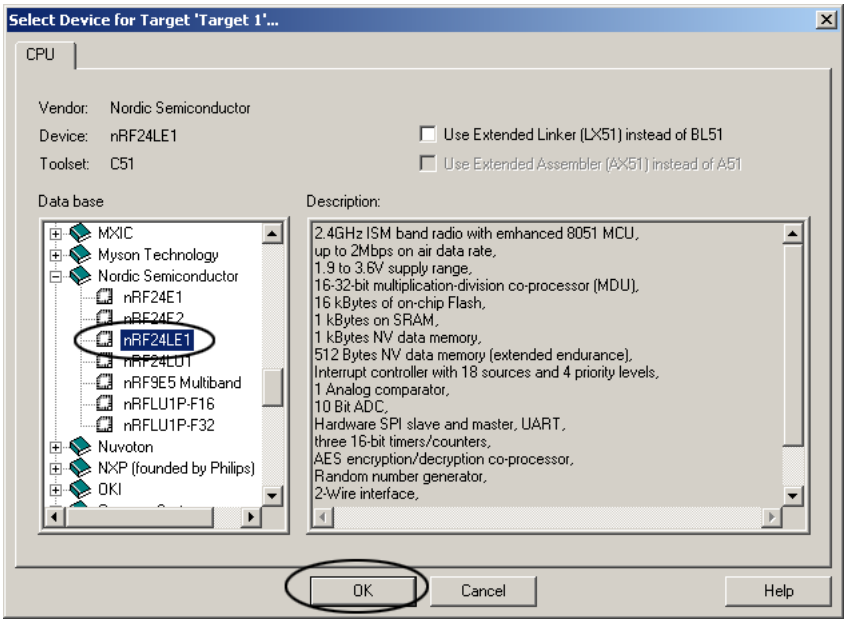


Figure 4. Selecting the correct device

5. Choose the device you want to start developing with (see device encircled in Figure 4.). For your project you will use the nRF24LE1. If you are using the full version of Keil uVision PK51 you can also choose to use the extended linker/assembler. However for this project you will use the evaluation version of the Keil compiler.
6. Select **OK**.

7. Select **No** when prompted to copy the standard startup code to the project folder. (The startup code can be added later if required.)

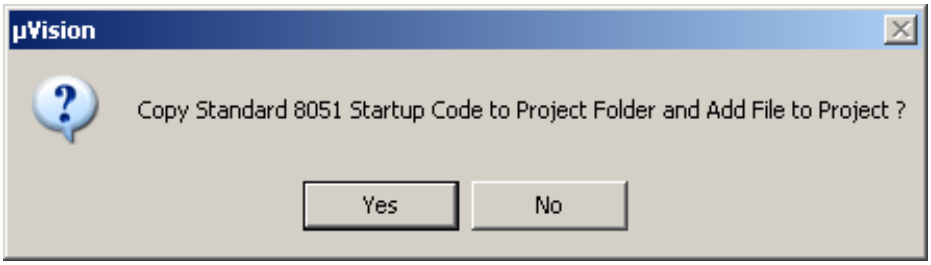


Figure 5. Dialog box

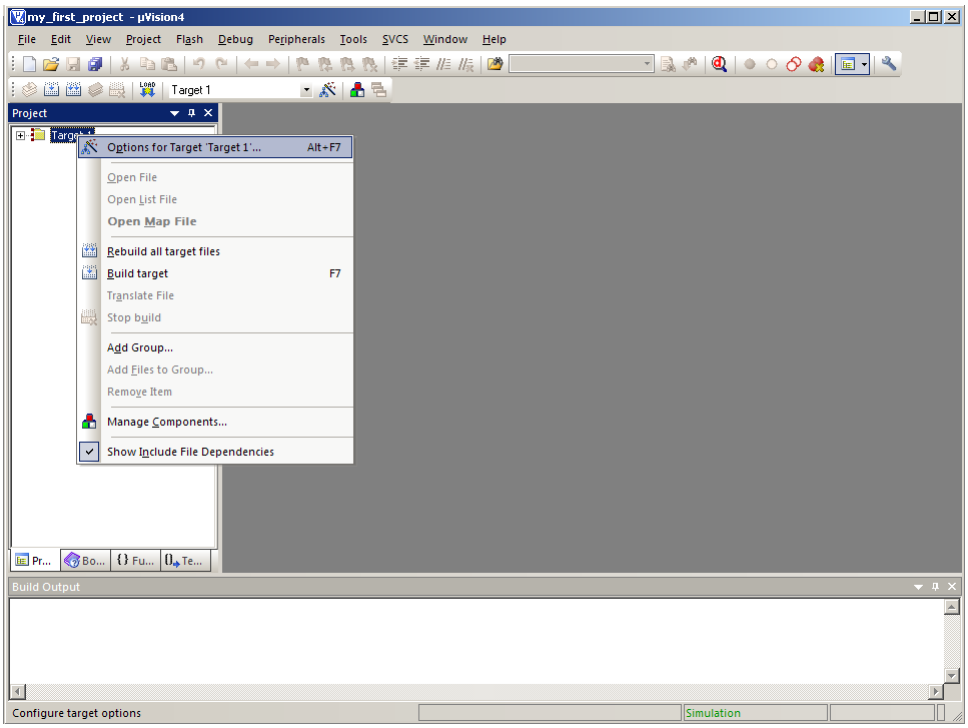


Figure 6. Main window

8. You need to configure the project before you can include files and start to write code. Right-click on “Target 1” in the Project tree view and choose **Options for Target** from the context menu.

9. Figure 7. shows a screenshot of the “Output” tab, where you set up Keil for your project.

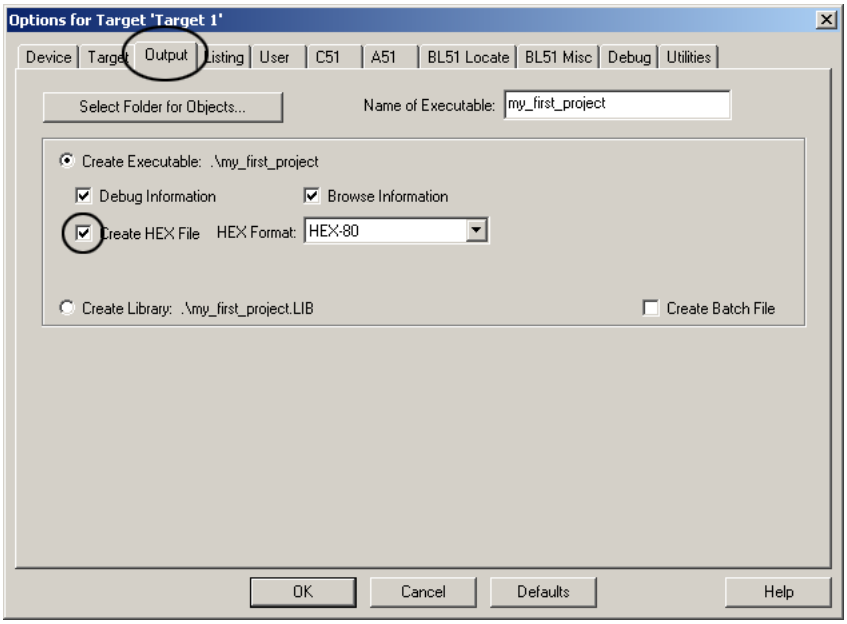


Figure 7. Check “Create HEX file”

10. Create a HEX file in the “Output” tab (see Figure 7.) to later program the nRF24LE1.
11. Next activate the “C51” tab and include some folders where you can later find the files you need for your application.

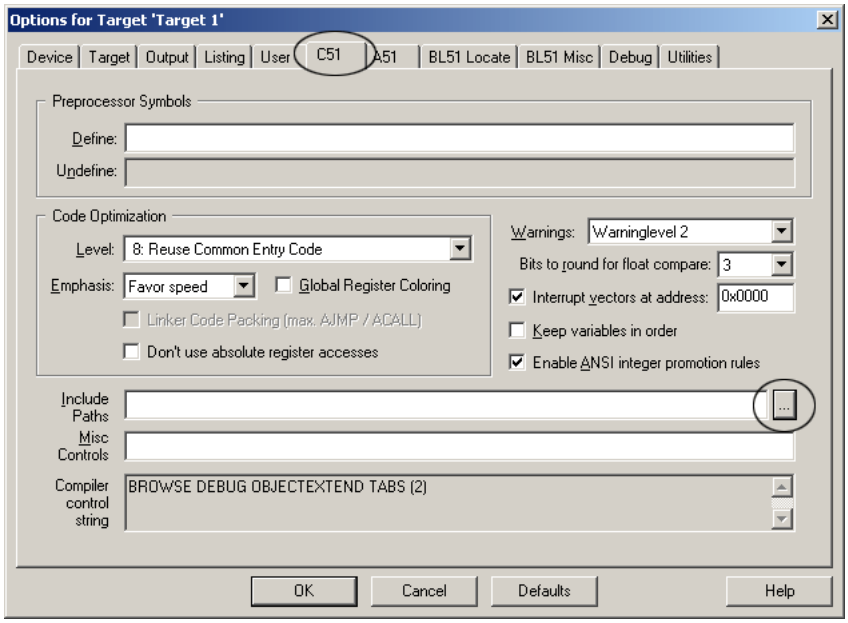


Figure 8. Selecting to include paths

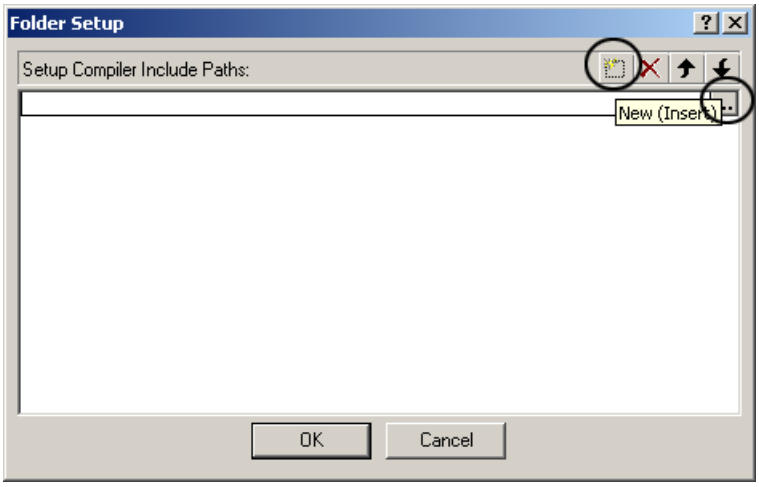


Figure 9. Selecting folders to include

12. By default no paths are included, so you need to include four paths for this project, from the "Folder setup" dialog. See Figure 9.
13. Use the function buttons to include all the paths as displayed in the screenshots in Figure 10. on page 11.

14. Figure 10. show the paths that contain the files needed for this project.

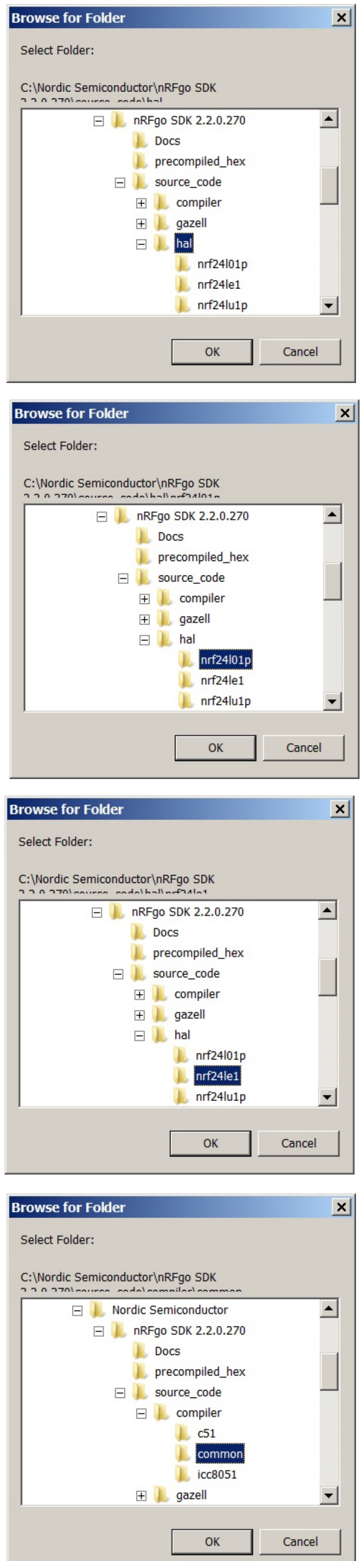


Figure 10. Browse to the folders to include, and repeat for four folders

15. After choosing the folders you want to include, confirm your choice by selecting **OK** in the "Browse for folder" dialog. See Figure 10. on page 11.

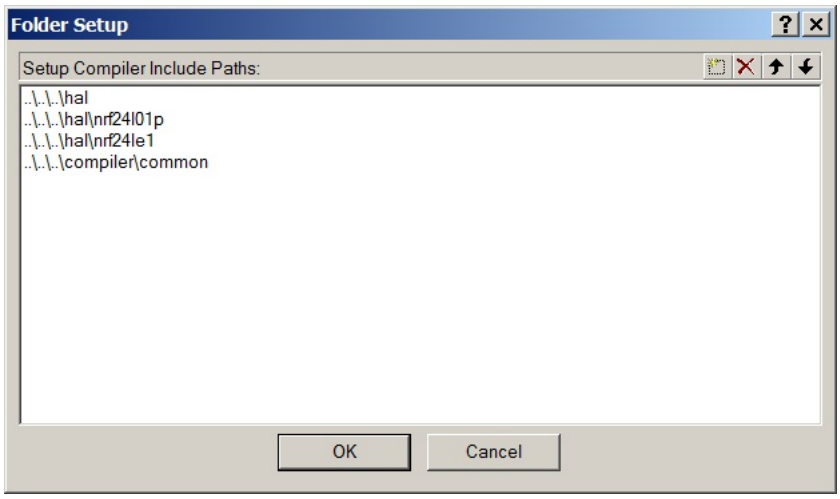


Figure 11. Confirm by pressing OK

16. The folders you now have included contain all the HAL files that you will use later to configure the various hardware modules inside the nRF24LE1. If you are developing with another device, you will need to include the hardware specific files for that device. All devices supported by the HAL have their own folders. Select **OK** in the "Folder setup" dialog as shown in Figure 11. when done.
17. Required paths are now included in your project. See Figure 12.

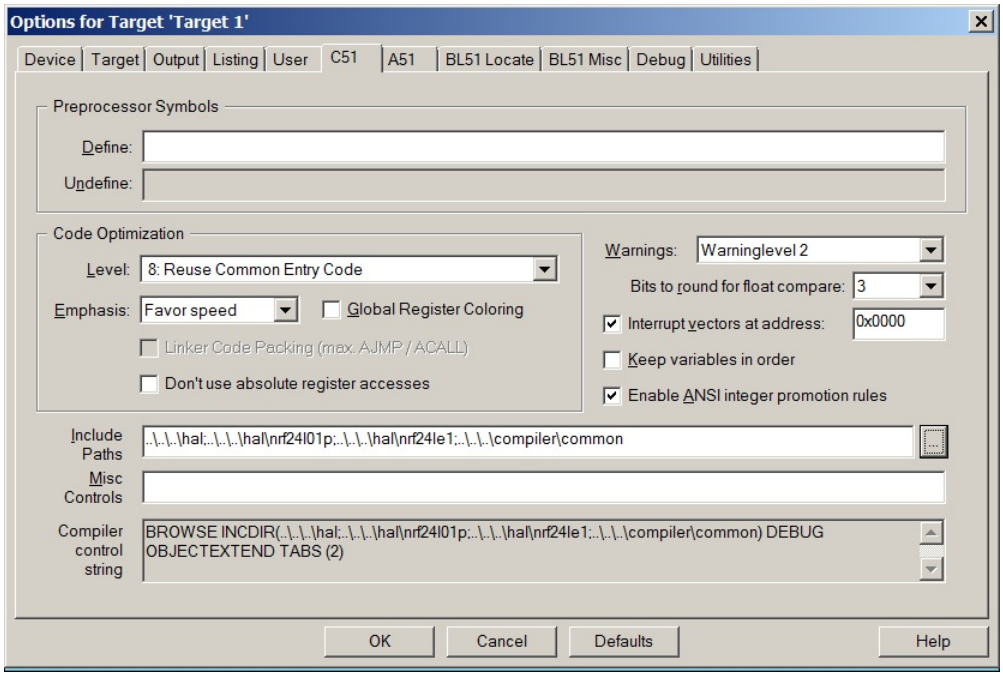


Figure 12. Required paths now included in the project

18. If you are using the extended linker of Keil you have the option to write 'REMOVEUNUSED' in the "Misc controls" field in This will reduce code space by removing unused functions during compilation.
19. When you compile a project with functions that are not used, a related warning message may appear. To disable this warning you can write '15, 16' in the "Disable warning numbers" field. See encircled area in Figure 13.
20. Now set up the debugger to use with the Keil compiler. Take care to select the nRFprobe Keil driver as shown in Figure 14. on page 14 and Figure 15. on page 15.
21. If you cannot find nRFprobe Keil Driver from the list, then make sure you have installed nRFprobe before you continue.

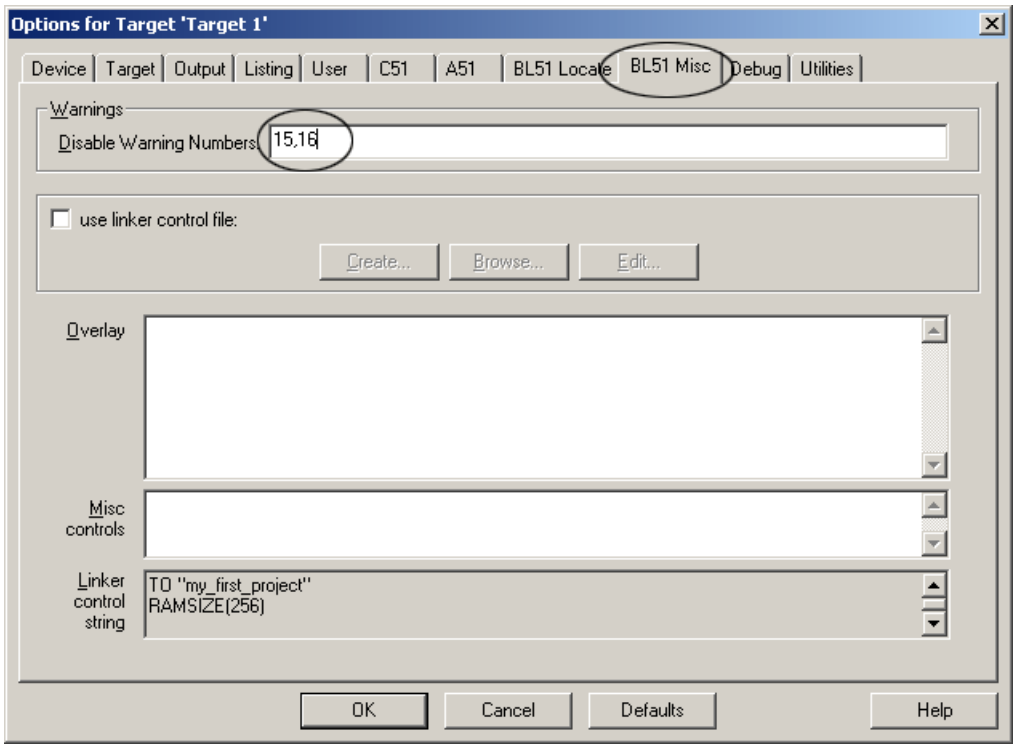
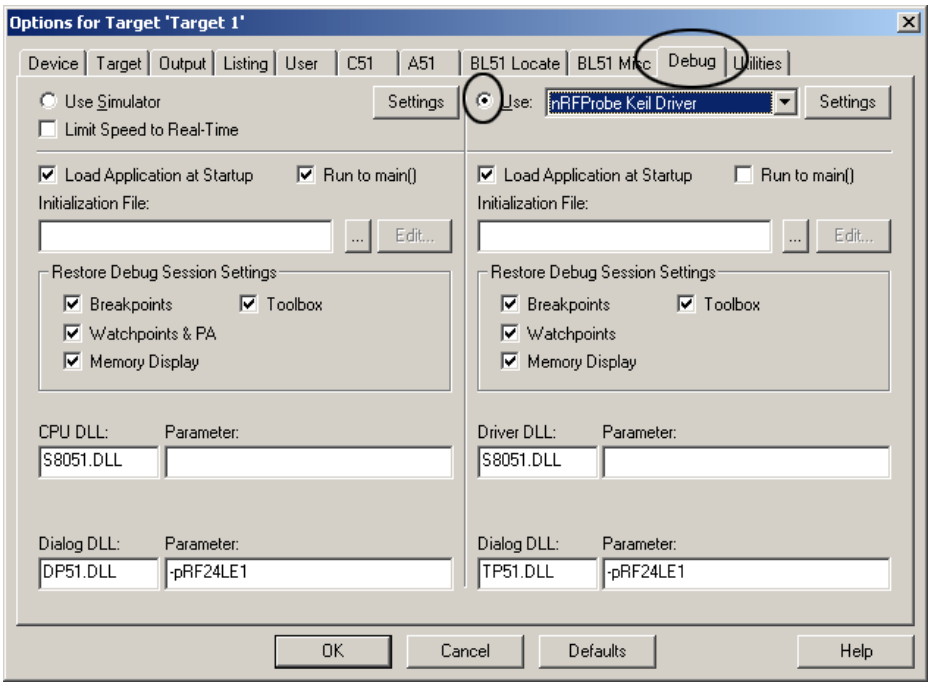
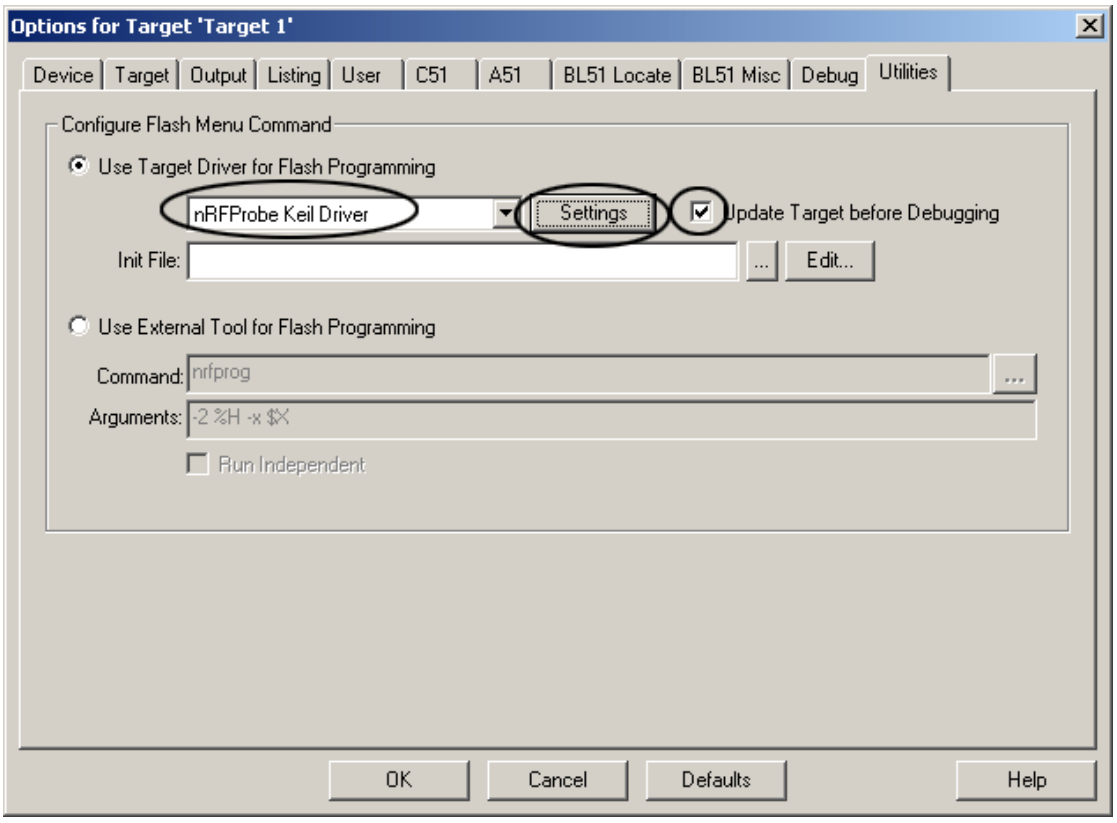


Figure 13. Disabling two common warning messages (optional)



Note: Make sure to check “Use”. See the encircled area in the “Debug” tab.

Figure 14. Selecting the nRFprobe debug driver



Note: Make sure to check “Update target before debugging”.

Figure 15. Selecting the nRFprobe debug driver

22. By selecting the **Settings** button in Figure 15. you can configure the nRFprobe debugger so you, for instance, can program or debug your own hardware. Refer to the documentation for the nRFgo Starter Kit on how to connect your own hardware.

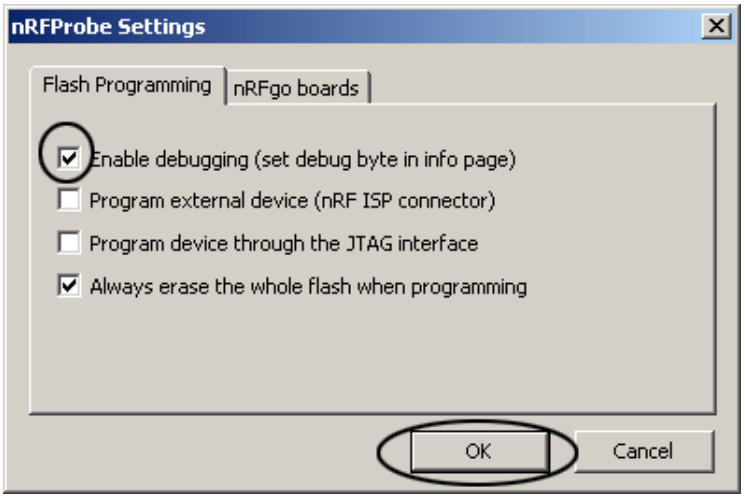


Figure 16. Settings for the nRFprobe driver (optional)

Note: If you want to run the application stand alone, then you need to uncheck “Enable debugging” (see example of this box checked in Figure 16. on page 15) in the “nRFProbe Settings” dialog before programming the device.

23. Click the **OK** button in the “nRFProbe settings” dialog to use the changes you have made to the project, and save the project from the **File** menu by selecting **Save All.**
24. You can right-click in the Project tree view to add and also rename some of the folders to give them more appropriate names. See encircled area in Figure 17. for illustration.

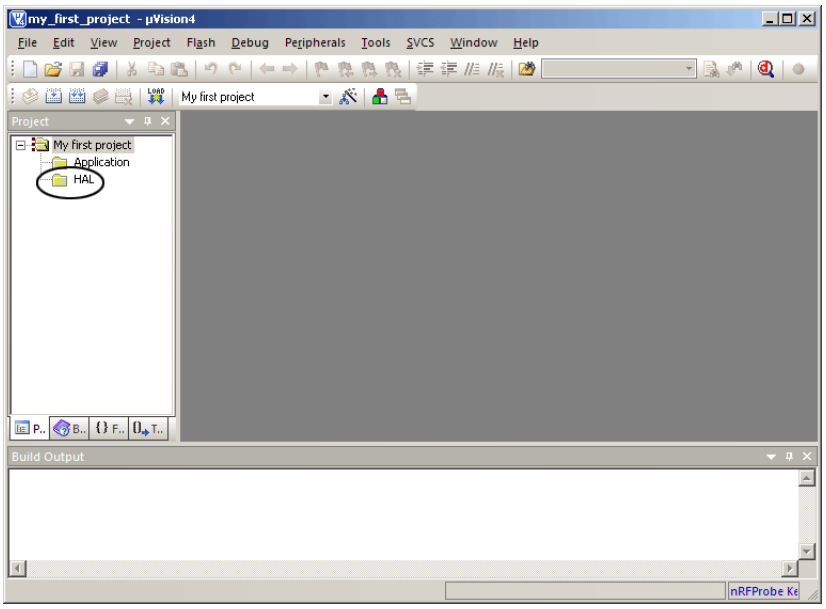


Figure 17. Project set up and ready for source code

You have now completed Step 1 of the tutorial, and the project is ready for you to start writing your first application. This step is outlined in section 3.2 on page 16.

3.2 Step 2: Your first application

The project is now configured, and you can start writing the first few lines of application code in order to compile and program the nRF24LE1. This section explains how you do so.

1. Before you can start writing code you must add an empty source file called **main.c** in the project folder.
2. From the Project tree view’s context menu, select **Add files to group “application”...** as shown in Figure 18. on page 17 The “Add files to group ‘application’” dialog appears.

3.

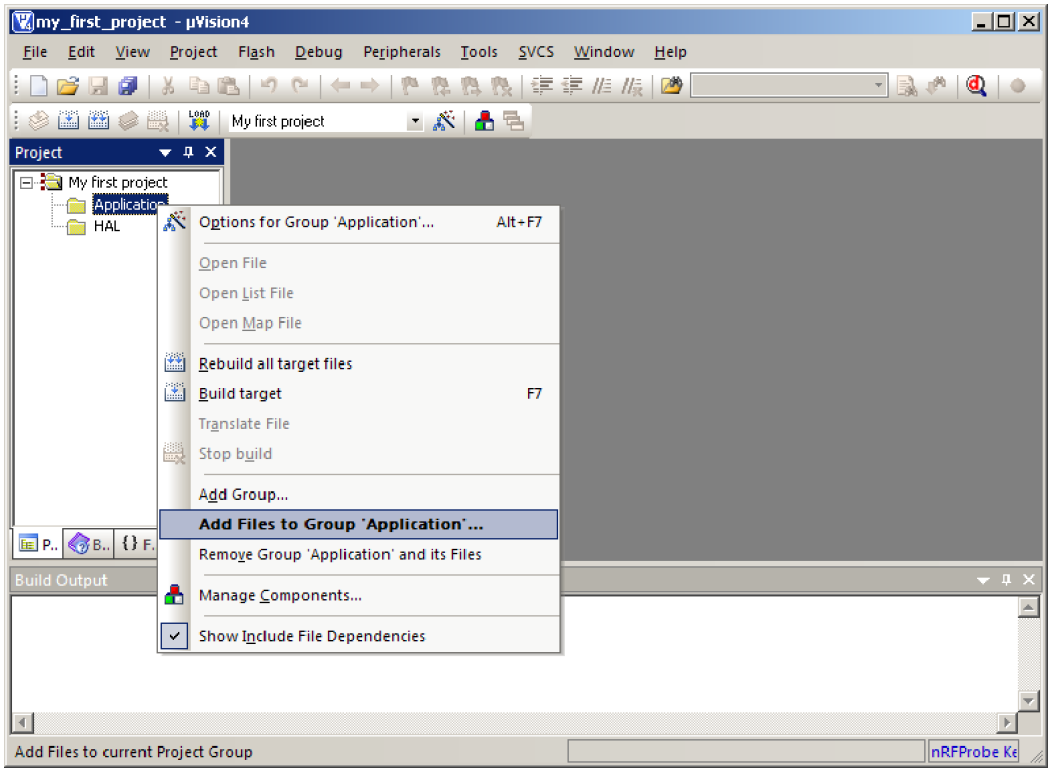


Figure 18. Adding the first file to the project

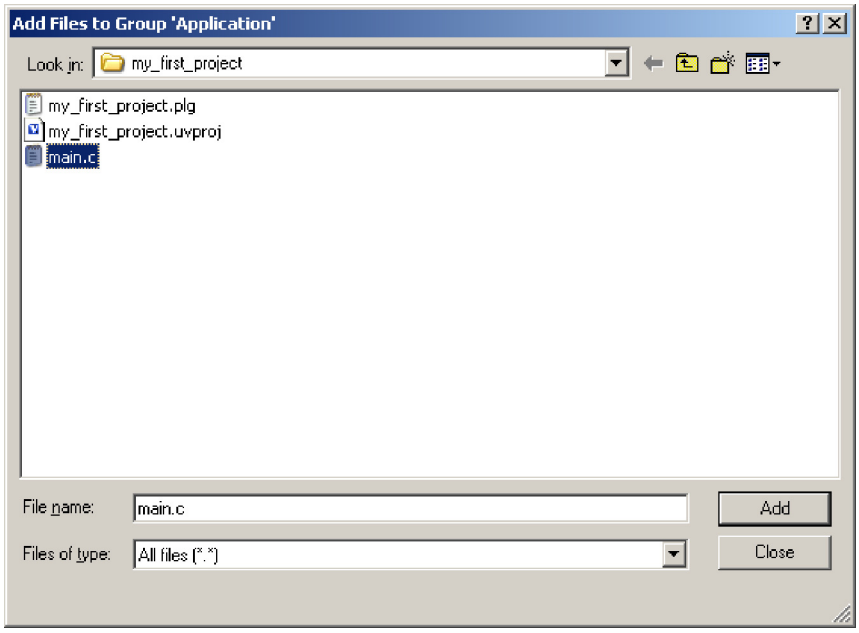


Figure 19. Creating a new file called main.c

4. Create a text file by right-clicking in the folder view and choosing **New text document** and renaming the created file to main.c. See Figure 19. This file will work as an entry point for your application.

5. Double-click on the main.c in the Project tree view to edit the file.

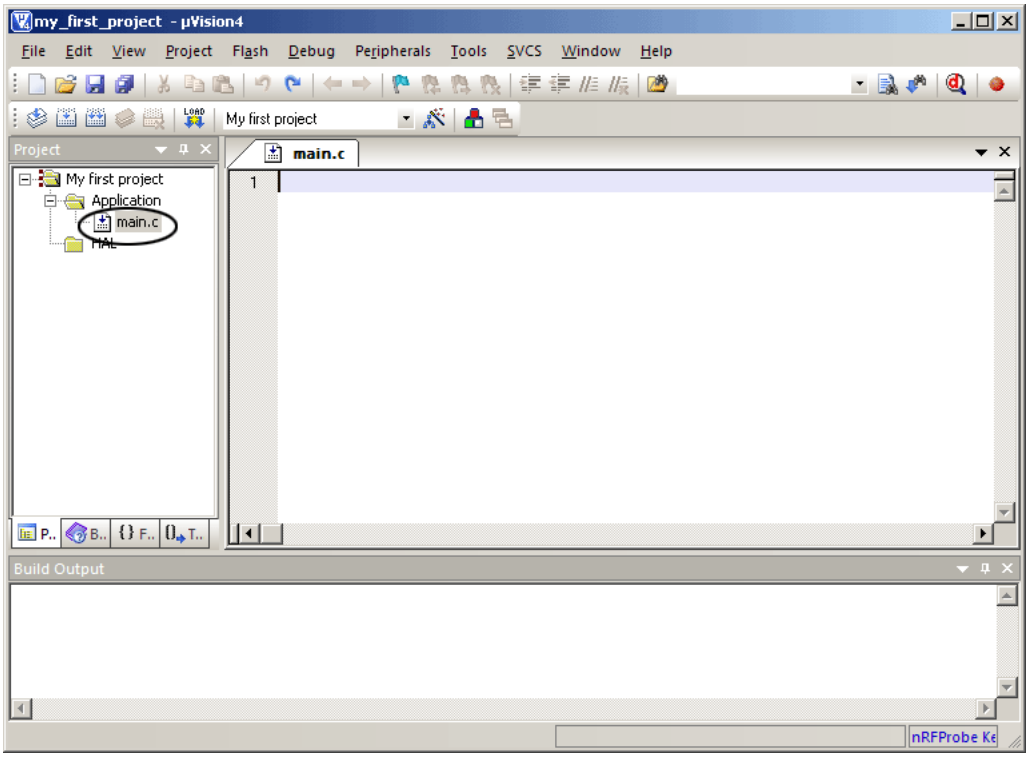


Figure 20. Main window with empty main.c file

6. Add the empty main.c to the project, write some lines of code and compile it.
7. You can copy Code example 1 from Figure 21. into the space for writing code below the main.c tab for your first application. See Figure 22. on page 19.
8. The example code will continuously toggle Port 0 pin 0. If Port 0 pin 0 doesn't toggle on your development kit module, then you should check the header on the module is not connected to the 32 kHz XO.

```
/* My first application */

#include <Nordic\reg24le1.h>

// Main routine
void main()
{
  // Set P0 as output
  P0DIR = 0x00;
  while(1)
  {
    // Toggle a GPIO
    P00 = !P00;
  }
}
```

Figure 21. Code example 1

9. After writing the code example into the main.c you can compile the program by selecting **Rebuild**. See Figure 22.

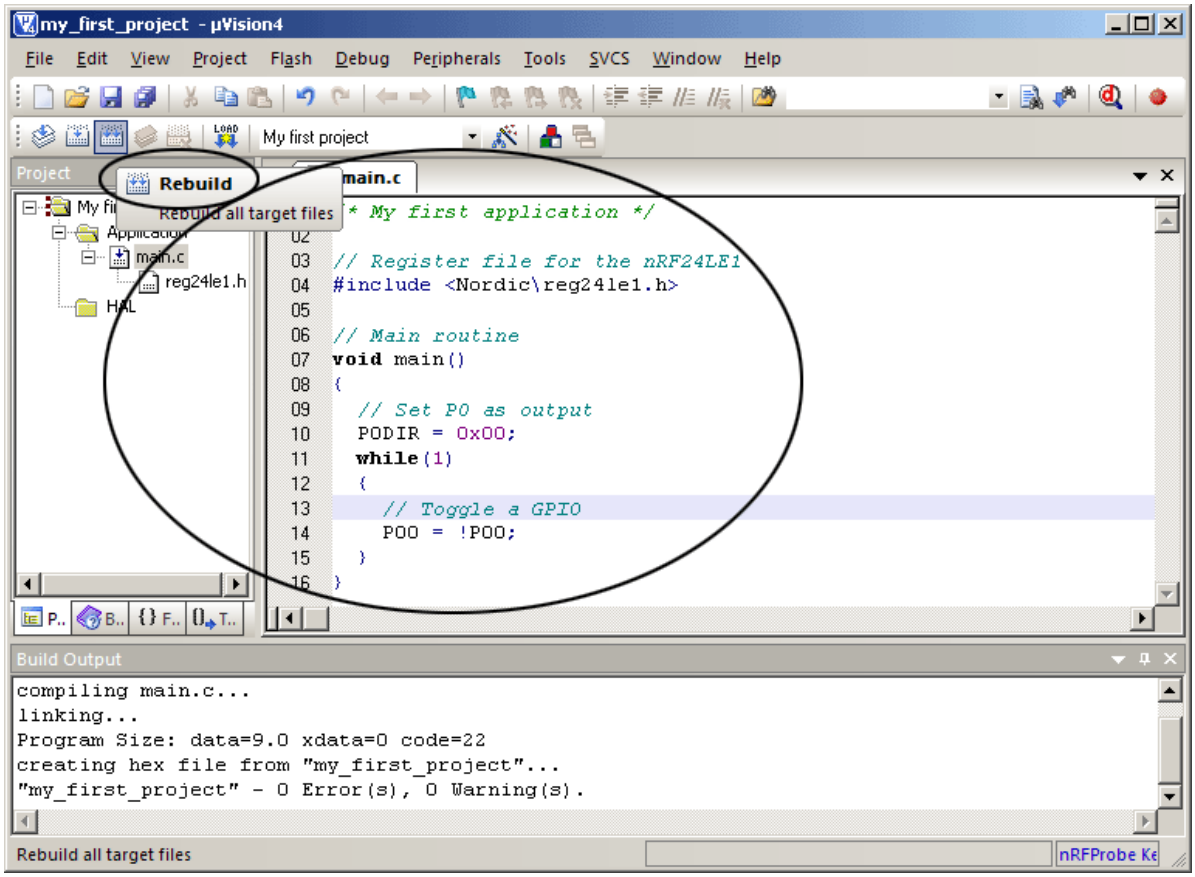


Figure 22. Example code with build of HEX file (shortcut F7)

10. After compiling the code you can program the nRF24LE1 module plugged into the nRFgo Motherboard by selecting **Download**. See encircled area in Figure 23. on page 20.
11. If you want to run the application without the debugger you need to deselect “Enable Debugger” (see Figure 16. on page 15).
12. The first time you attempt to program an nRFgo module you will be prompted about which nRFgo Motherboard you want to use. See Figure 24. on page 20. Make sure that an nRFgo Motherboard and an nRF24LE1 module are plugged into the computer.

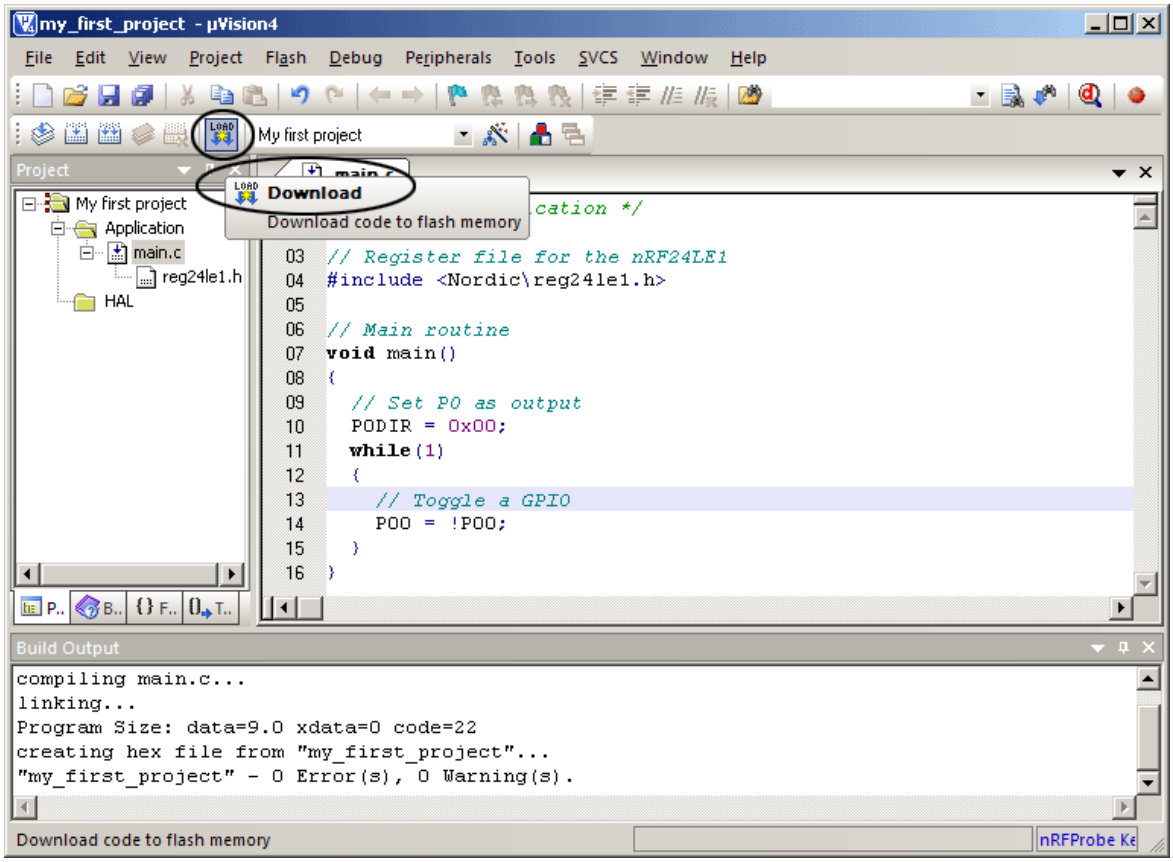


Figure 23. Programming the nRFLE1 module from Keil

13. The board you use depends on the number of boards connected to the computer. You can have several projects open at the same time and program boards individually. Boards can be programmed from their own project. Each nRFgo Motherboard has a small status LED that indicates the board number.

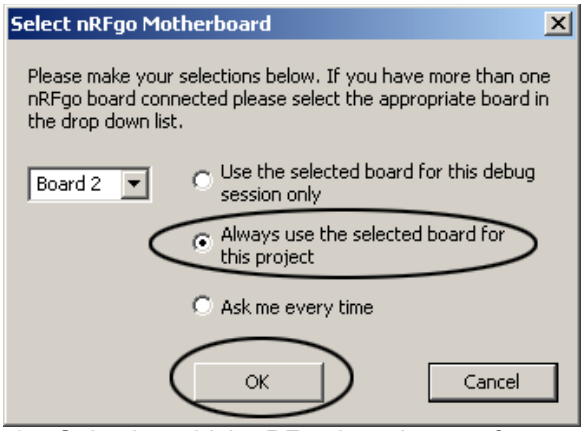


Figure 24. Selecting which nRFgo board to use for your project

You have now successfully compiled and programmed your first application, and the next step outlined in section 3.3 will help you include files to make more complex applications.

3.3 Step 3: Including files

Writing source code from scratch can be time consuming, since you will have to study in detail all the registers in the datasheet to set up the device. To shorten development time there are ready-made software modules that can be included and re-used for several projects. Each module contains the functionality for a specific hardware module.

For this project you want to use the ADC within the nRF24LE1, so you will include the HAL for the ADC.

1. Select the HAL folder in the Project tree view and activate the context menu by right-clicking. Select **Add Files to Group 'HAL'...** from the context menu. See Figure 25.

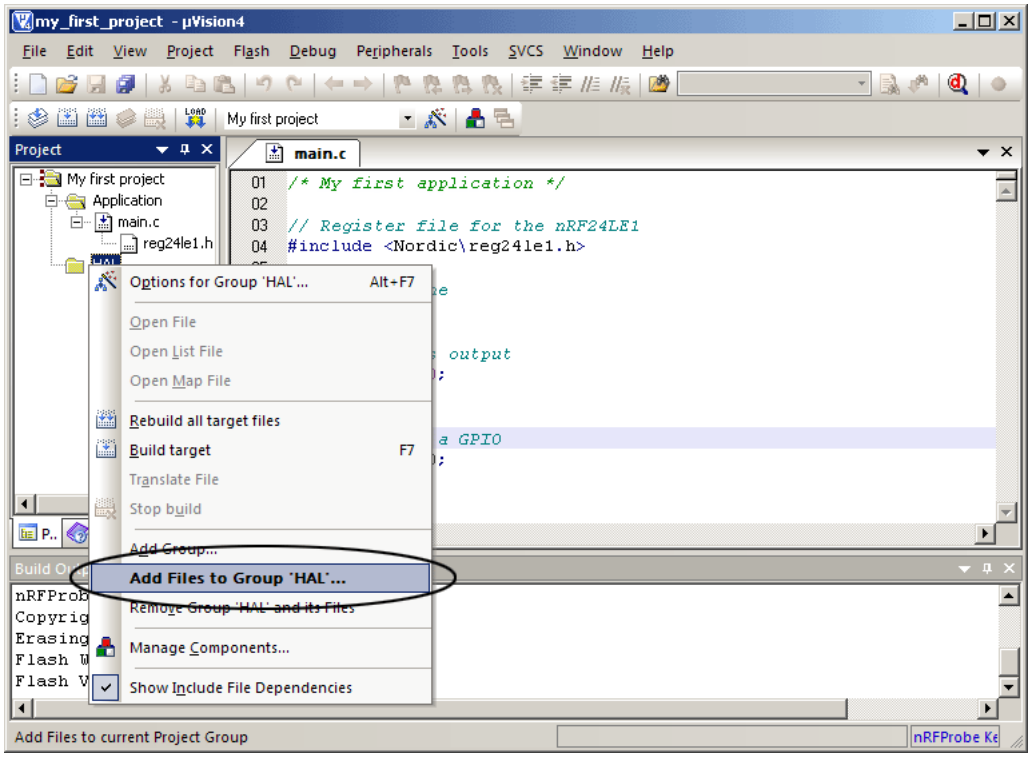


Figure 25. Including the HAL for the ADC

2. The HAL can easily be included by right-clicking on the HAL folder in the Project tree view and then selecting “Add the hal_adc.c to the project” as shown in Figure 25. on page 21. See Figure 26.

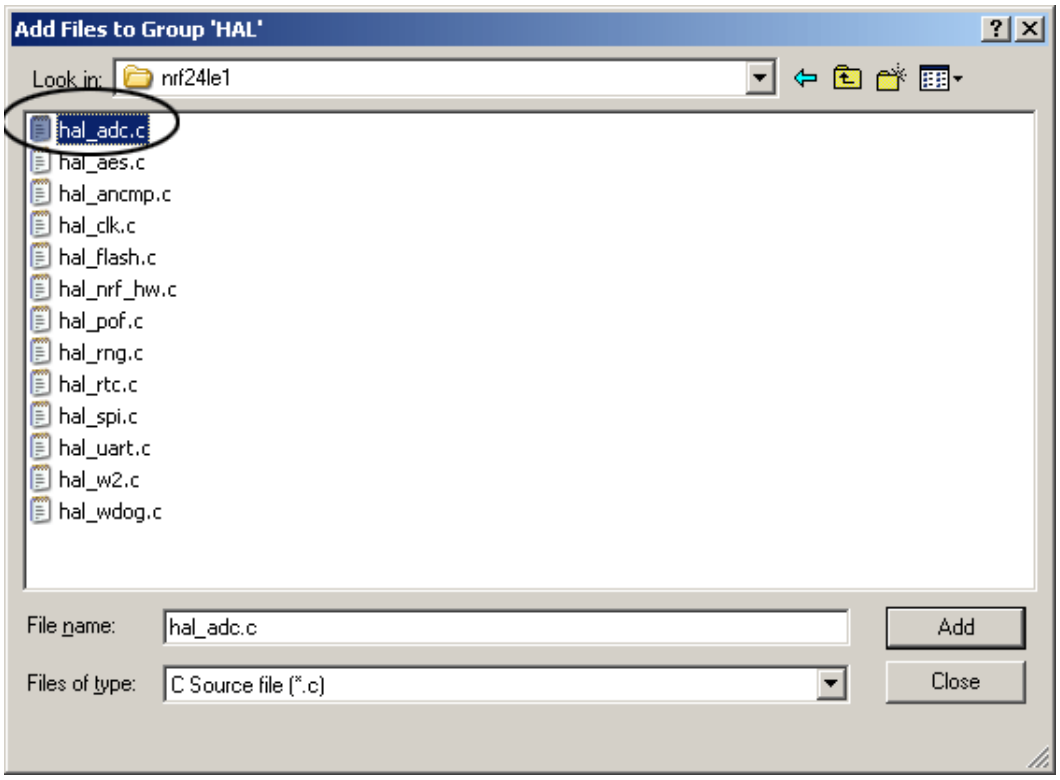


Figure 26. Browse to \Source code\hal\nrf24le1 and include hal_adc.c

3. The HAL for the ADC is now included in the project, and to use the functionality of the HAL you can include it in the main.c by writing #include <hal_adc.h>. See encircled area in Figure 26.

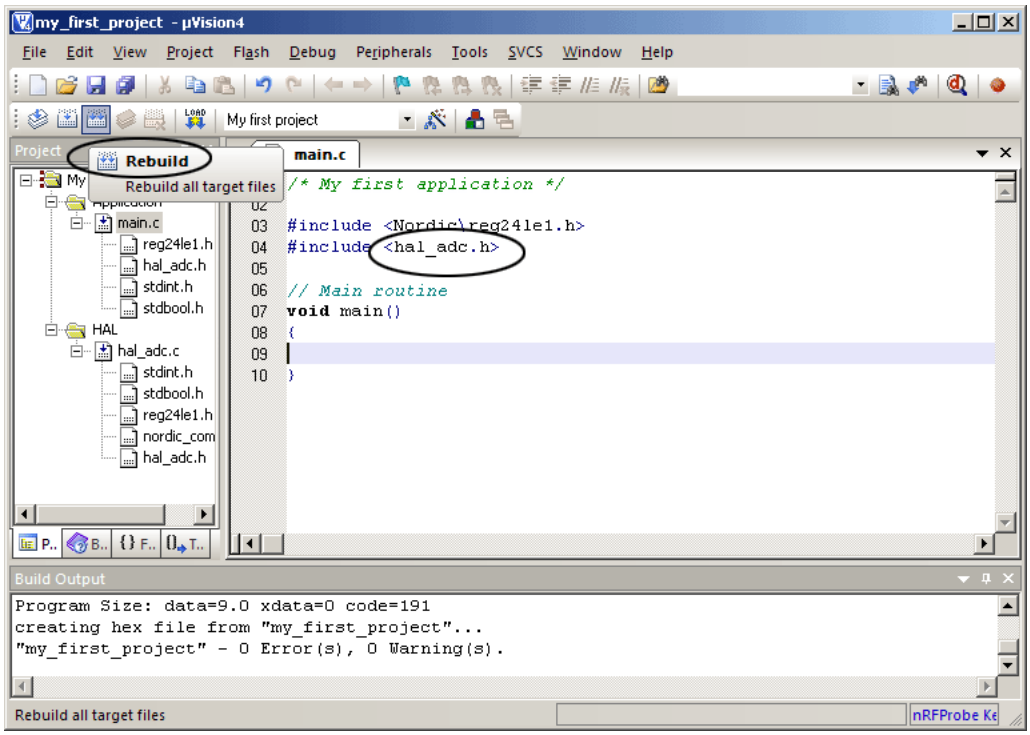


Figure 27. Rebuilding target

4. After including the file in the project you can **Rebuild** the application. See Figure 27. You can use a project with an empty main routine. The compiled code space is now 191 bytes, the reason for this is that you have included all the functionality of the ADC to your project, but not used it yet. If you are using the extended linker, then unused functions are not compiled and there is no increase in code size.

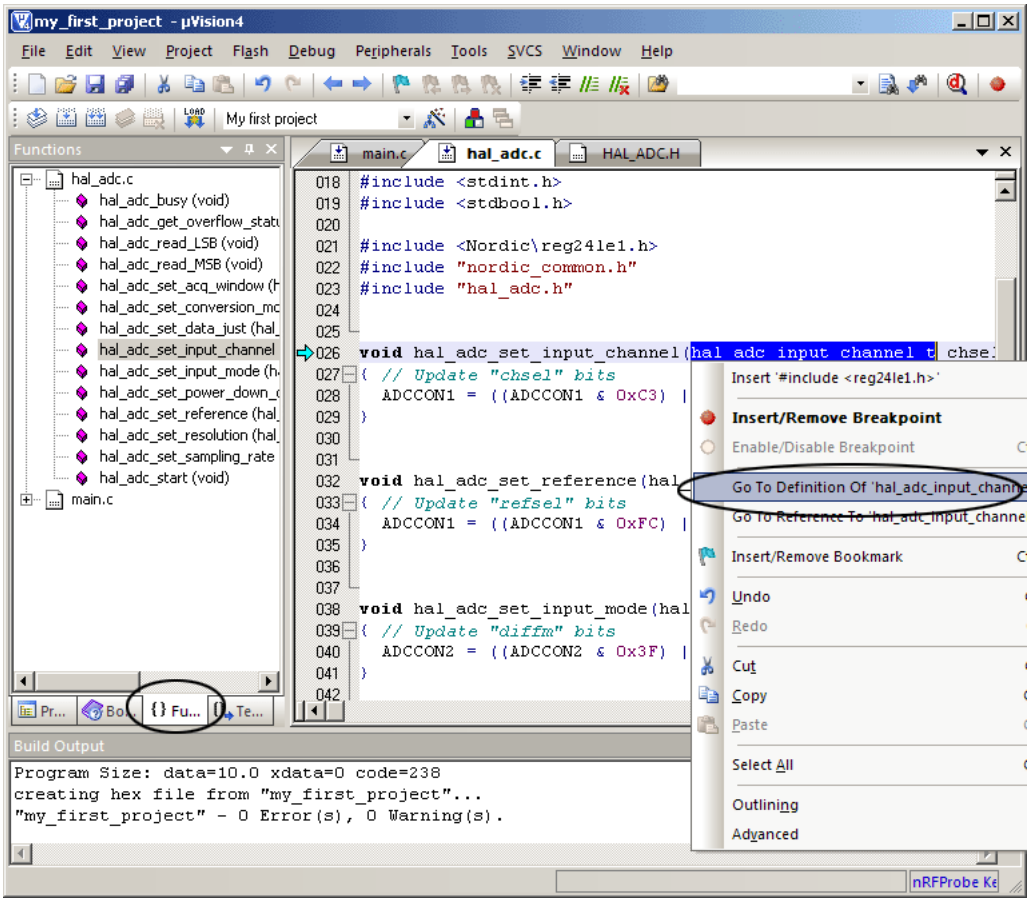


Figure 28. Available functionality of the HAL ADC with source code

5. Optionally you can browse the functions in the hal_adc.c by selecting the “Function tab”. See the encircled area to the left in Figure 28. Click on one of the functions to see the argument for the functions and the source code for it. All parameters for the functions are enumerators that can be used in the function call.
6. You can also list up all available enumerators by using the **Go To Definition of** menu option (see Figure 28.) for each function call (see Figure 29. on page 25).

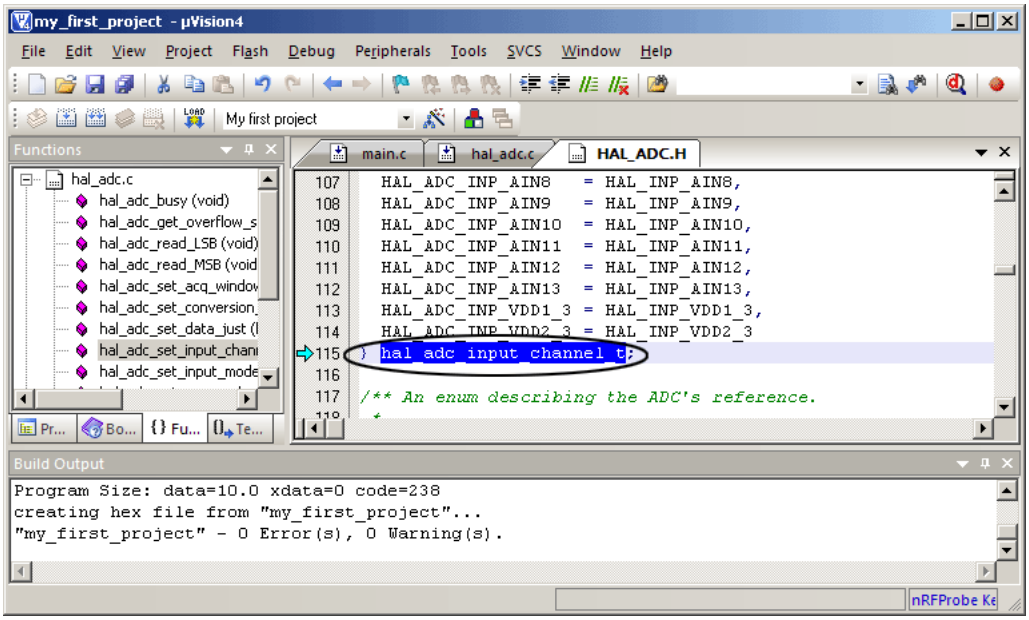


Figure 29. Available parameters used as input for specific function

7. By using the functions listed in the hal_adc.c you can easily build an application that can initialize and read an analog input, and there is no need to study the datasheet in detail. Similar HAL are available for all modules inside the nRF24LE1.

8. For now you can copy the content from Figure 30. (Code example 2) below into your the main.c.

```
/* My second application */

#include <Nordic\reg24le1.h>
#include <hal_adc.h>

void main()
{
  // Init a variable for the ADC measurement
  uint8_t adc_measurement = 0;

  // Configure ADC
  hal_adc_set_input_channel(HAL_ADC_INP_AIN0);
  hal_adc_set_reference(HAL_ADC_REF_VDD);
  hal_adc_set_input_mode(HAL_ADC_SINGLE);
  hal_adc_set_conversion_mode(HAL_ADC_SINGLE_STEP);
  hal_adc_set_resolution(HAL_ADC_RES_8BIT);
  hal_adc_set_data_just(HAL_ADC_JUST_RIGHT);

  while(1)
  {
    hal_adc_start(); // Start the ADC
    while( hal_adc_busy() ) // Wait for the ADC to finish a
conversion
      ;
    adc_measurement = hal_adc_read_LSB(); // Read the ADC result
  }
}
```

Figure 30. Code example 2

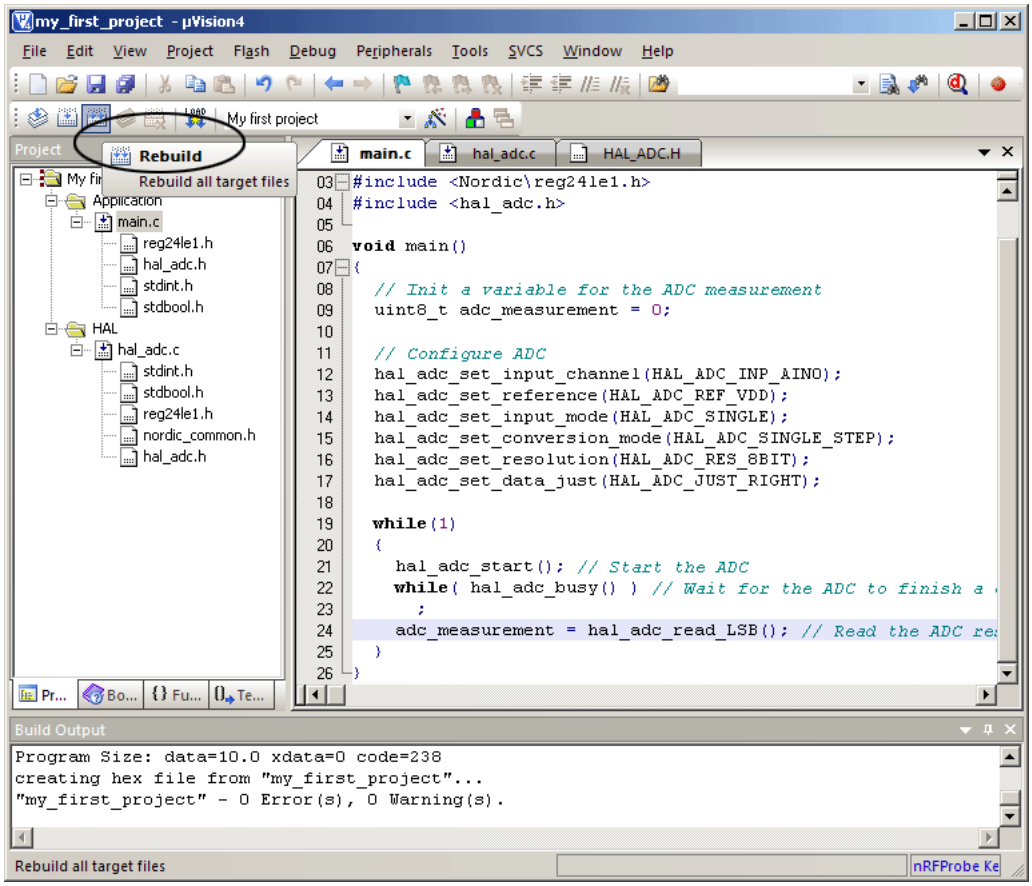


Figure 31. Creating an application that measures an analog input by using the HAL

9. After pasting the code into main.c, as shown in Figure 31.,use the **Rebuild** function to rebuild the target and verify that it builds successfully.
10. You have now successfully included and used the first HAL module. The HAL provides an easy way to start using a specific functionality such as the ADC. Your next step will be to start debugging the project.

3.4 Step 4: Debug your project

An important feature of the Keil compiler is the possibility it gives to debug the application by using the nRFprobe debugger.

1. To start the debug session select the **Start Debug Session** button. See encircled area in Figure 32. on page 28.

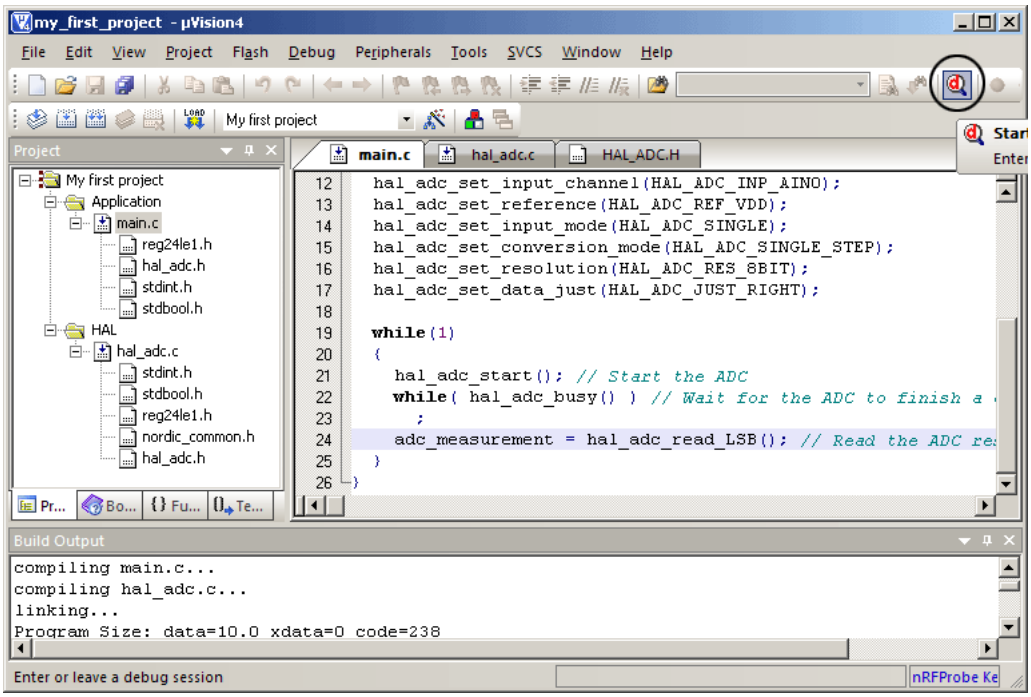


Figure 32. Start debug session (shortcut Ctrl + F5)

2. In debug mode you have access to the disassembly window, SFR registers and all variables in the project. You can run code in real time, set breakpoints, single step or run to cursor.
3. To simplify the project window, you can exit the dissassembly window for the purposes of this application note. (See the encircled area in Figure 33. on page 29 for how to exit the window. You can open it again later from the **View** menu at the top.) Also select the “Project” tab which is shown on Figure 33. on page 29 to see all project files.

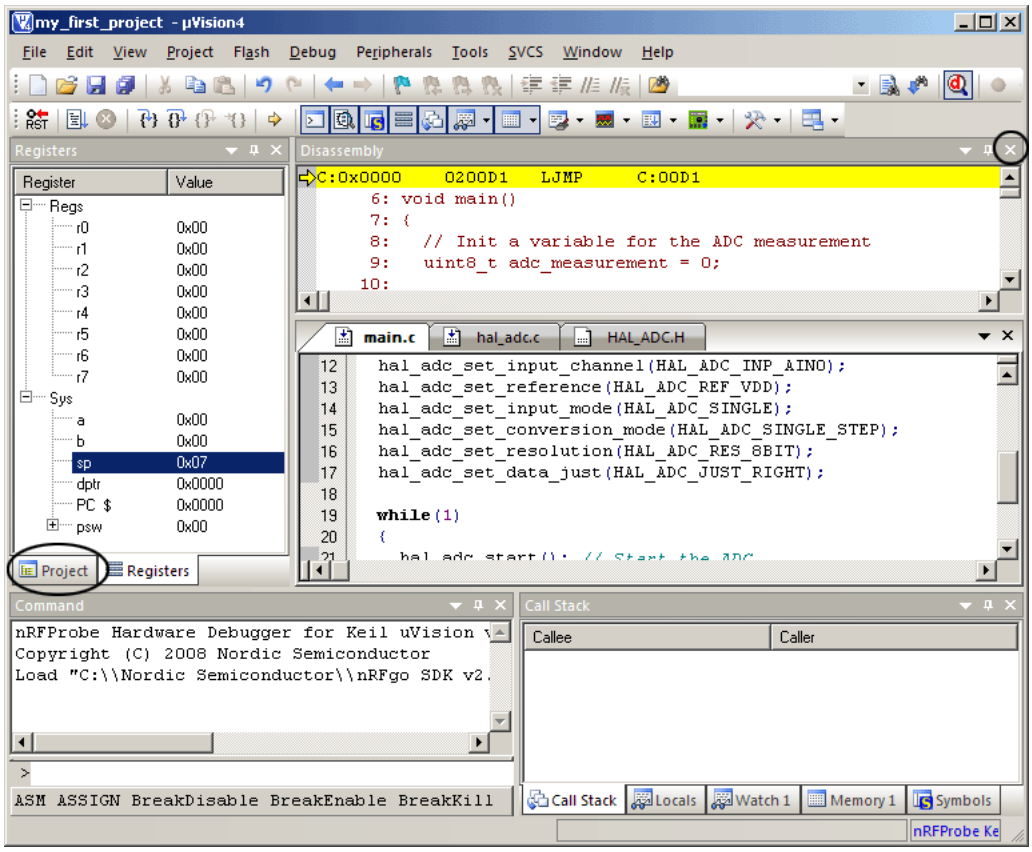


Figure 33. Default debug window

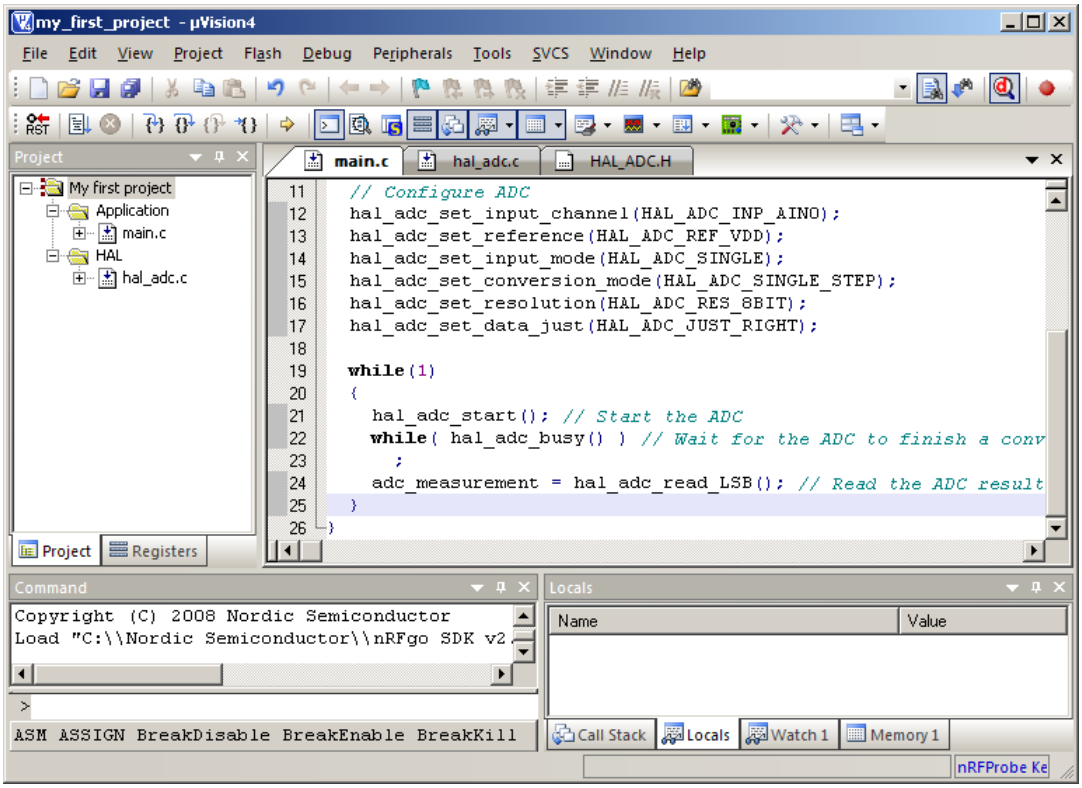


Figure 34. Simple debug window (disassembly closed)

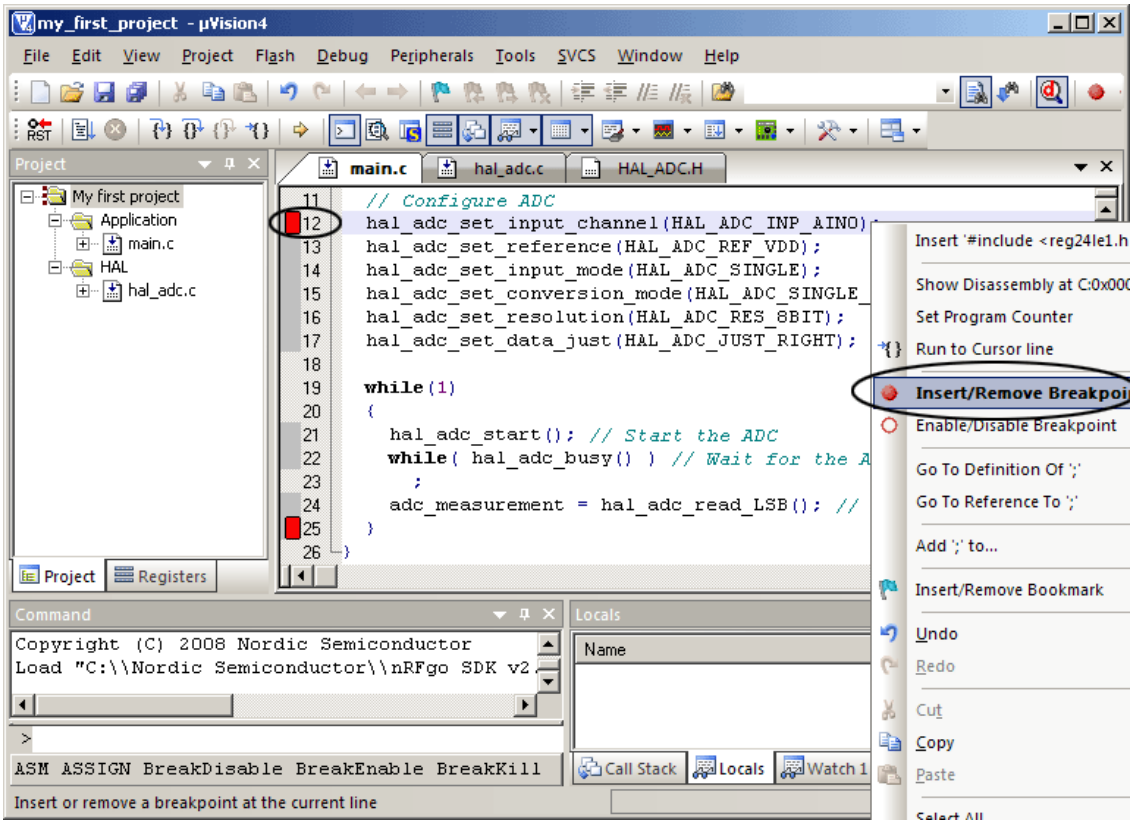


Figure 35. Set breakpoints (shortcut F9)

4. You can set breakpoints in the application by double-clicking on a code line, as illustrated in Figure 35. or right-click and select **Insert/Remove breakpoint**.

Figure 36. Debug toolbar

5. The debug toolbar can be used to reset or run the code. You can also run to line, single step or step into code during debugging. For each step you can read out variables and register values. Move cursor over the buttons to acquaint yourself with the purpose of the debug buttons.
6. Select the **Run Application** function button to run the code. See Figure 36. on page 31. The application will then run until it encounters a breakpoint or stops.

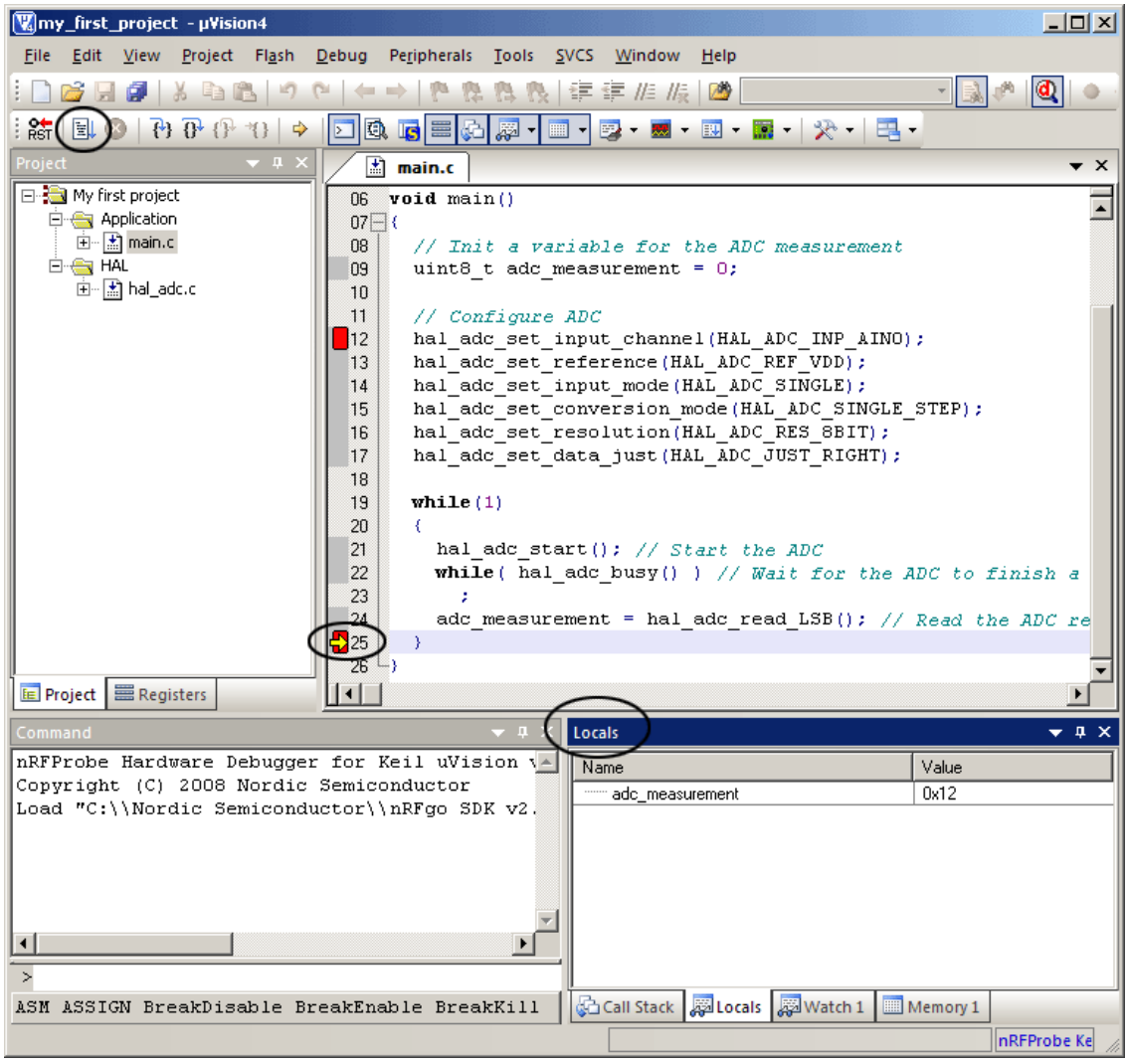


Figure 37. Running to breakpoint and reading out local variables

7. Each time the program stops, a yellow arrow (see encircled area around yellow arrow in Figure 37.) where the program currently is running, and local variables are read out in the “Locals” window to the bottom right of the Keil window.

8. In this example you can step through the loop and see the ADC measurement changes as you change the input, for instance place a finger on the analog input 0 to change reading.

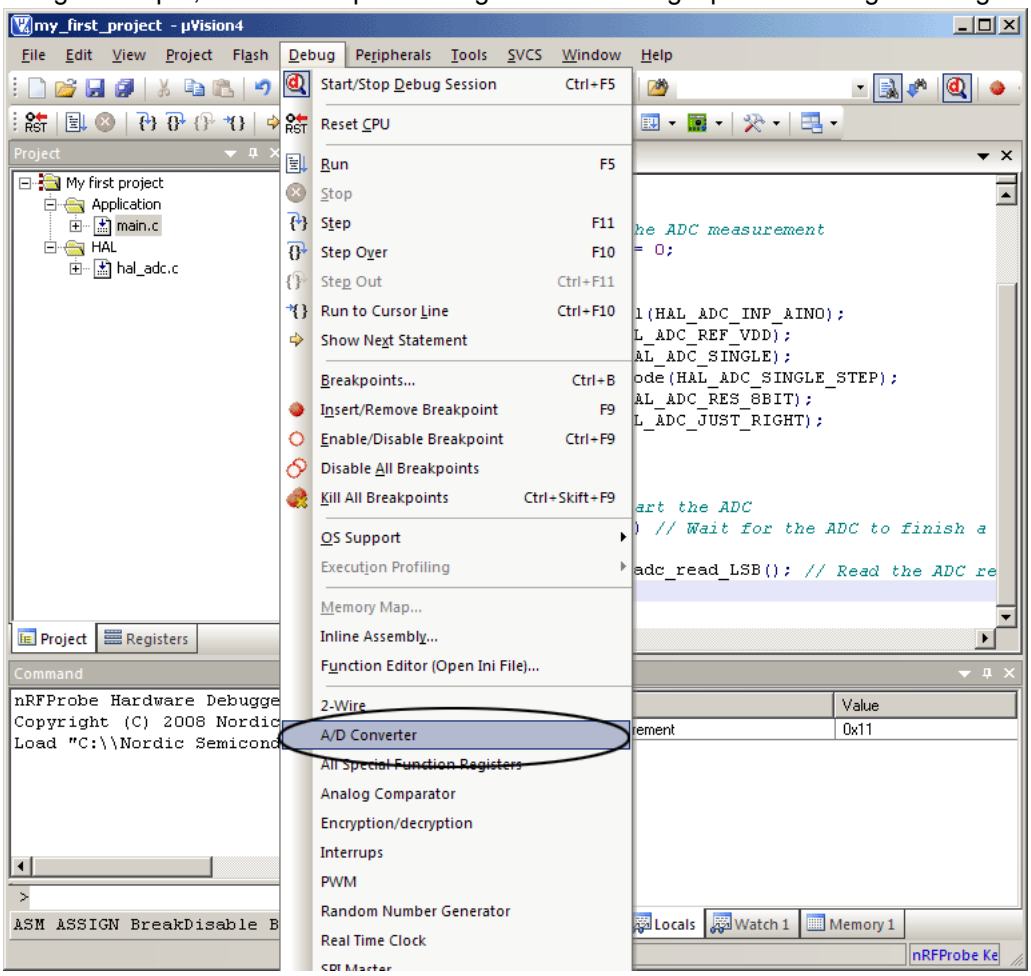


Figure 38. Debugging the A/D converter hardware module directly

9. You can also view the hardware module embedded in the nRF24LE1. See encircled area in Figure 38. for the relevant menu selection.
10. If you want to change configurations directly during run time without re-compiling, this can save time during prototype development to test different configurations.

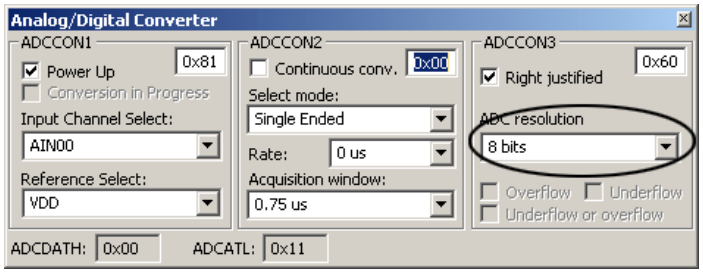


Figure 39. Screenshot showing how to modify the A/D converter configuration during debugging

You can now debug your own project and you have completed the tutorial. For more example projects make sure to view the projects included in the nRFgo SDK.

4 Conclusions

The nRFgo SDK with the Keil compiler provides an easy-to-use environment for programming and developing nRF devices. The nRFgo SDK contains the HAL, libraries and an RF protocol that can be used as-is or as a reference when writing your own applications.

Some of the advantages of using the nRFgo SDK are:

- Ready made software modules that can be re-used for several projects
- Proven and tested code
- Simple functions that are easy to use
- Shorter development time
- Flexible development environment

Appendix A - References

Useful reference projects and documentation during development include:

- Documentation for nRFgo SDK
- nRFgo SDK example projects
- Documentation for nRFgo Starter Kit
- Documentation for nRFprobe

Appendix B - Troubleshooting

The debugging is not working. What has happened?

- If nRFgo Studio is running at the time you are debugging, then debugging will not work.

I am not able to flash the nRF24LE1. What has happened?

- If the jumper on P7 on the Motherboard is missing you will not be able to flash the nRF24LE1.

The UART functionality is not working. What has happened?

- The GPIOs used during debugging cannot be used for other purposes during debugging, so any functionality such as UART will not work if the same pins are used for debugging.

The debugger is not working. What has happened?

- The debugger must be set up correctly to work, so make sure that the nRFprobe driver is selected for both Debug and Utilities in the 'Options for Target' menu in Keil. Also make sure to program the nRF24LE1 before debugging.

How can I run the code without a debugger?

- Make sure to disable the debug bit if you want to run the code without debugger, and vice versa.

  Note: The debug bit is set if the project file is moved.

Nothing seems to work. What has happened?

- Check the supply voltage (in nRFgo Studio) for the board if nothing seems to work.
- Make sure that you have the latest development software from http://www.nordicsemi.com/update/index.php
- Run nRFgo Studio and make sure that the Motherboard is using the latest firmware. You will be prompted to update the firmware if it is outdated.

I am having difficulty debugging in power-down mode. What has happened?

- It is not possible to debug in power down.

I am having difficulty using my GPIO as ADC. What has happened?

- A GPIO cannot be used as ADC if it's already used as UART, SPI, I2C, or something similar.

The error message "Error: Flash Download failed - C:\path\to\keil\BIN\nrfkeil515d.dll" appeared while I was trying to download the program to the Motherboard. What has happened?

If you encounter "Error: Flash Download failed - C:\path\to\keil\BIN\nrfkeil515d.dll" while trying to download your program to the Motherboard, please check your compiler settings. From the menu, select **Project, Options for project**, then verify that the "nRFProbe Keil Driver" is selected from the "Utilities" and "Debug" tab.

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Life support applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

Contact details

For your nearest dealer, please see http://www.nordicsemi.com.

Receive available updates automatically by subscribing to eNews from our homepage or check our website regularly for any available updates.

Main office:

Otto Nielsens veg 12
7004 Trondheim
Phone: +47 72 89 89 00
Fax: +47 72 89 89 89
www.nordicsemi.com

Revision History

Date	Version	Description
June 2011	1.1	Updated application note to reflect upgrade to SDK v. 2.2 and to reflect new convention for application note codes.
August 2010	1.0	