# nRF51 Development with GCC and Eclipse

## nAN-29

## Application Note v1.0

This application note is for developers who want to use Eclipse, the GNU C Compiler (GCC), and the GNU Debugger (GDB) to develop and debug programs on Nordic Semiconductor's nRF51 Series devices.

The nRF51 Series Software Development Kits (SDK) come with GCC compatible examples. This application note shows you how to import these examples into Eclipse allowing you to make use of the powerful Eclipse editor with C-language syntax highlighting and completion. You will also learn how to configure and use the GDB plug-in with Eclipse to debug the examples.

# 1 Required tools

The following tools were used in this application note. The version number of each tool is supplied for reference, but as software is updated frequently, it is not required that the specific version be used.

### Sourcery CodeBench Lite version 2012.03-56 for ARM EABI

Download and install from: http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/.

### J-Link software version 4.54d

Download and install from: http://www.segger.com/jlink-software.html.

### Eclipse IDE for C/C++ Developers version 1.5.1.20121004-1506 (Juno Service Release 1)

Download and install from: http://www.eclipse.org/downloads/.

Additional tools used in this document:

- CDT Main Features version 8.1.1.201209170703
- GDB Hardware Debugging version 7.0.0.201209170703
- embsysregview version 0.2.2

**Note:** Instructions for adding these tools are found in ***section 1.1 Setting up Eclipse***.

## 1.1 Setting up Eclipse

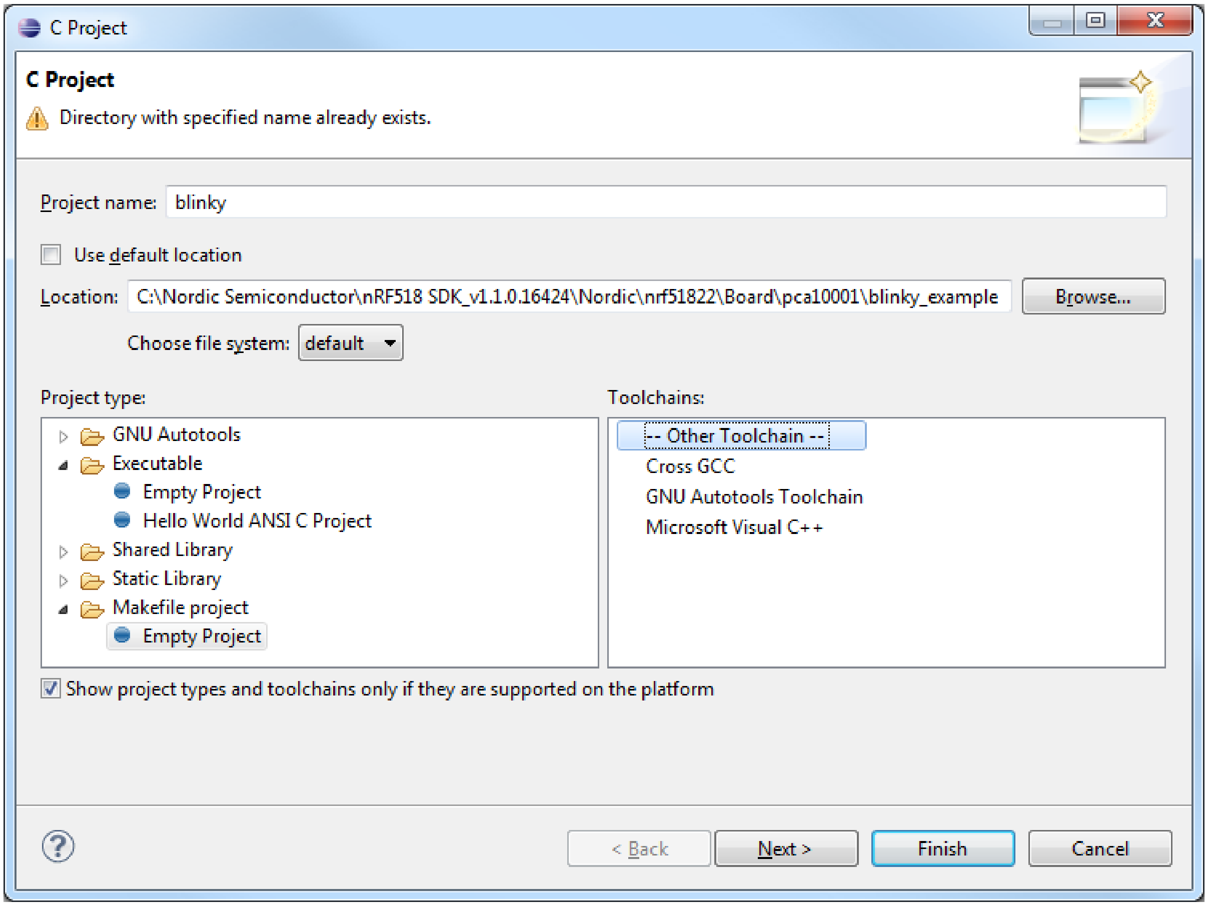
1. Start Eclipse.
2. Click **Help** and select **Install New Software**.
3. Add **http://download.eclipse.org/tools/cdt/releases/juno** to the list of repositories (replace "juno" if you have installed another release of Eclipse).
4. Install **CDT Main Features** and **GDB Hardware Debugging**.
5. Repeat step 2 and add **http://embsysregview.sourceforge.net/update** to the list of repositories.
6. Install **embsysregview**.
7. Copy file **nrf51.xml** from:
   **C:\Nordic Semiconductor\nRF518 SDK_<version>\Nordic\nrf51822\SVD\nrf51.xml**
   to the folder
   **C:\eclipse\plugins\org.eclipse.cdt.embsysregview_<version>\data\cortex-m0\Nordic**.
   (Create a Nordic subfolder if it does not exist.)
   This installs the register view file for Nordic Semiconductor devices.

## 1.2 Import and configure an existing Makefile project in Eclipse

We will use the example project blinky in this guide. The following is a set of instructions to help you configure a Makefile project in Eclipse. An example setup can be seen in ***Figure 1***.

### Create a new C project in Eclipse

1. On the File menu, click **New** and select **C Project**.
2. Type **blinky** in the Project name field.
3. Uncheck **Use default location**.
4. In the Location field, click **Browse**.
5. Select the root folder where the blinky project file is located as the location for the project.
6. In the Project type window, click **Makefile project** and select **Empty project**.
7. In the Toolchains window, select **Other Toolchain**.
8. Click **Finish**.



Directory with specified name already exists.

Project name: blinky

☐ Use default location

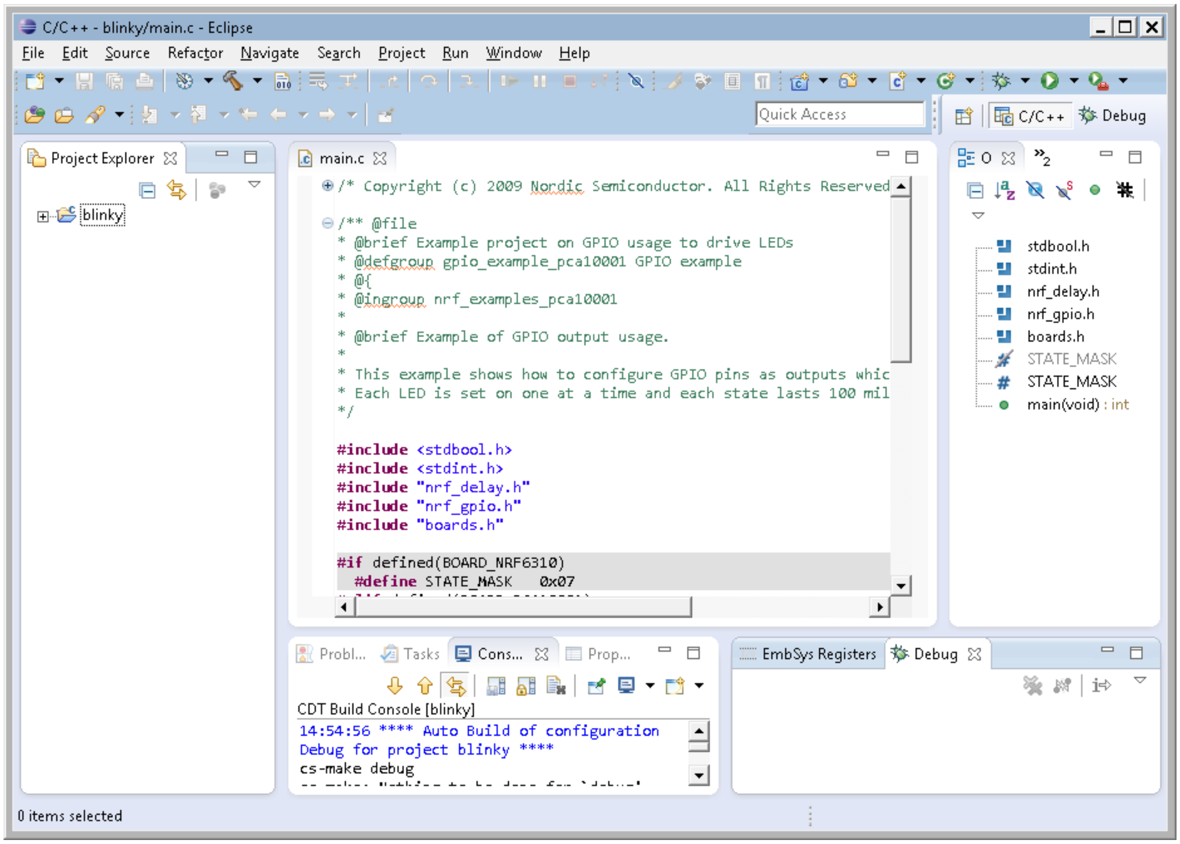
Location: C:\Nordic Semiconductor\nRF518 SDK_v1.1.0.16424\Nordic\nrf51822\Board\pca10001\blinky_example

Choose file system: default

☑ Show project types and toolchains only if they are supported on the platform

*Figure 1 Setting up a Makefile project*

1. In the Project Explorer, right click **blinky** and select **Properties**.

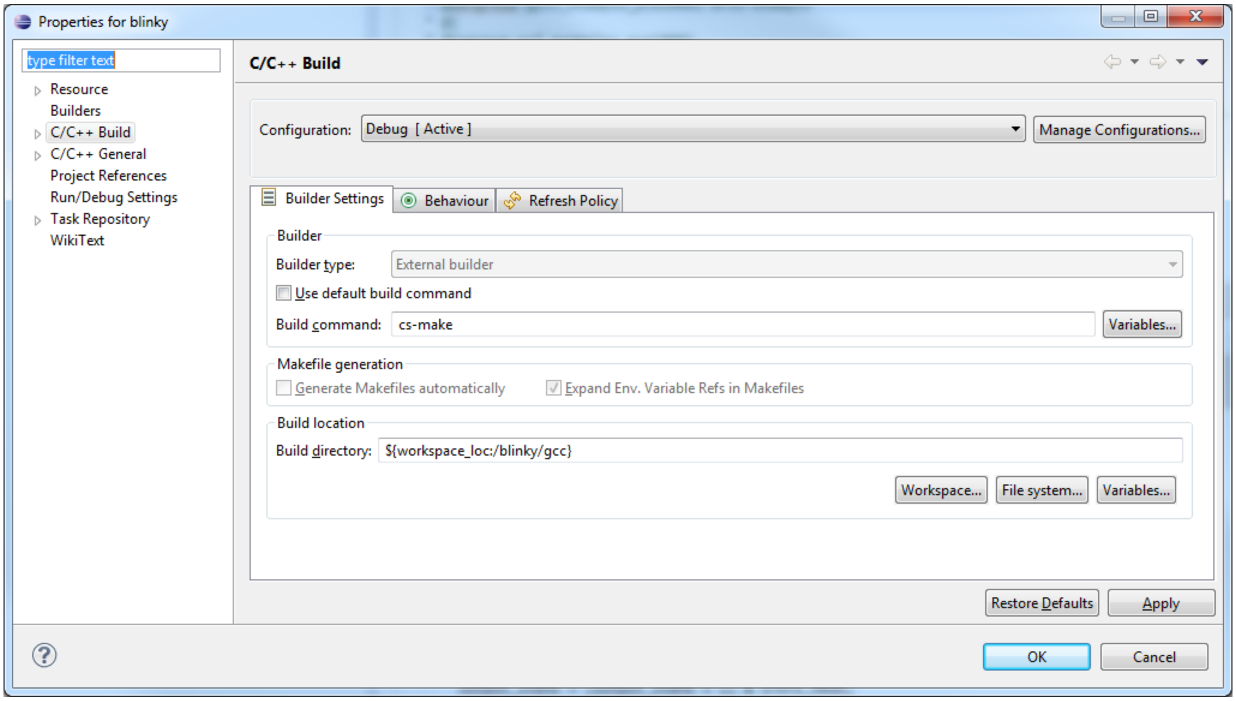


```
/* Copyright (c) 2009 Nordic Semiconductor. All Rights Reserved
/** @file
 * @brief Example project on GPIO usage to drive LEDs
 * @defgroup gpio_example_pca10001 GPIO example
 * @{
 * @ingroup nrf_examples_pca10001
 *
 * @brief Example of GPIO output usage.
 *
 * This example shows how to configure GPIO pins as outputs whic
 * Each LED is set on one at a time and each state lasts 100 mil
 */

#include <stdbool.h>
#include <stdint.h>
#include "nrf_delay.h"
#include "nrf_gpio.h"
#include "boards.h"

#if defined(BOARD_NRF6310)
  #define STATE_MASK   0x07
```

```
CDT Build Console [blinky]
14:54:56 **** Auto Build of configuration
Debug for project blinky ****
cs-make debug
```

*Figure 2 Project Explorer*

2. Go to the C/C++ Build section. In the Builder Settings tab change **Build command** to **cs-make**.
3. Click **Workspace** and change the build location to the **GCC subfolder** of the blinky project.
4. Click **Manage Configurations** and change the Configuration name to **Debug**.



Configuration: Debug [ Active ]

Builder type: External builder

☐ Use default build command

Build command: cs-make

☐ Generate Makefiles automatically

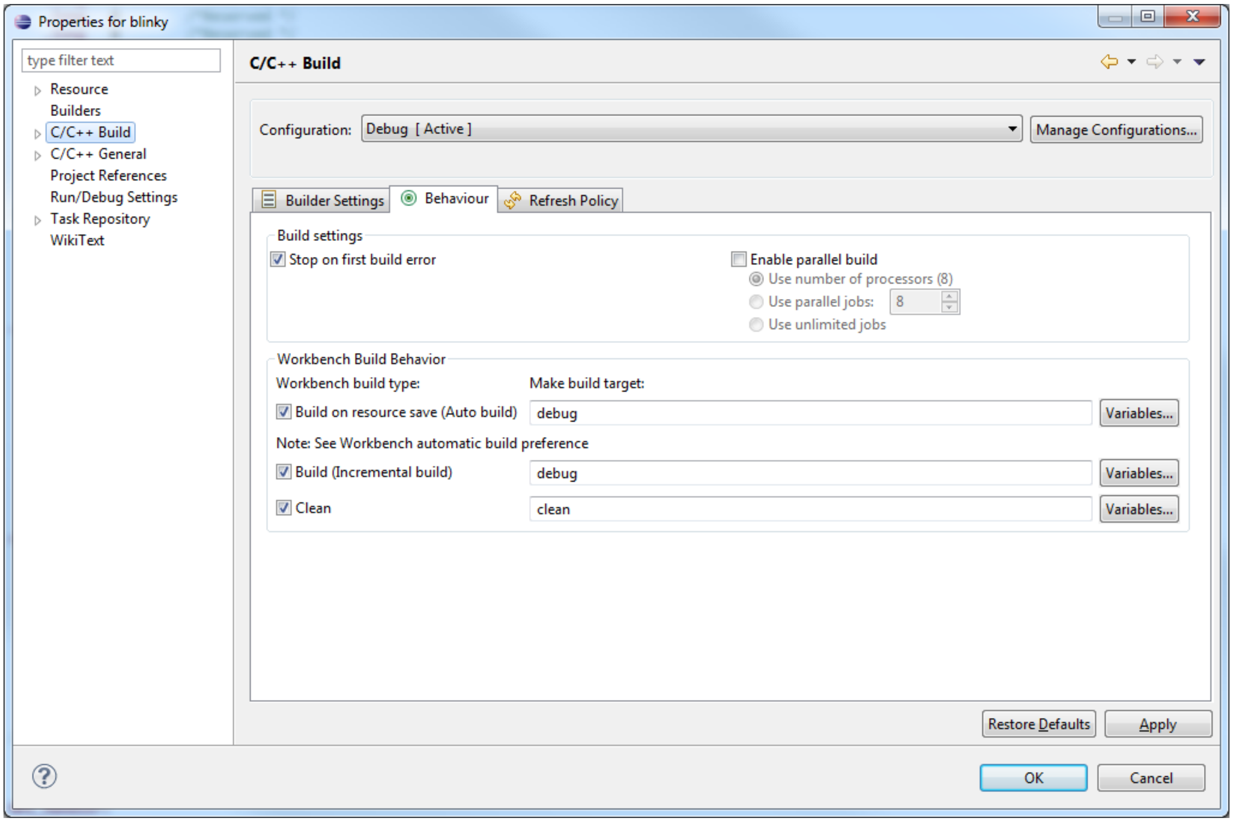
☑ Expand Env. Variable Refs in Makefiles

Build directory: ${workspace_loc:/blinky/gcc}

*Figure 3* Build properties

The C/C++ build in Eclipse uses the target **all** in the Makefile. This must be changed to **debug** in the Nordic make files.

1. Click the **Behaviour** tab.
2. Check **Build on resource save (Auto build)**.
3. Change **Make build target** to **debug**.
4. Repeat these steps for **Build (Incremental build)**.



Configuration: Debug [ Active ]

Build settings

☑ Stop on first build error

☐ Enable parallel build
● Use number of processors (8)
○ Use parallel jobs: 8
○ Use unlimited jobs

Workbench Build Behavior

Workbench build type: | Make build target:

☑ Build on resource save (Auto build) | debug

Note: See Workbench automatic build preference

☑ Build (Incremental build) | debug

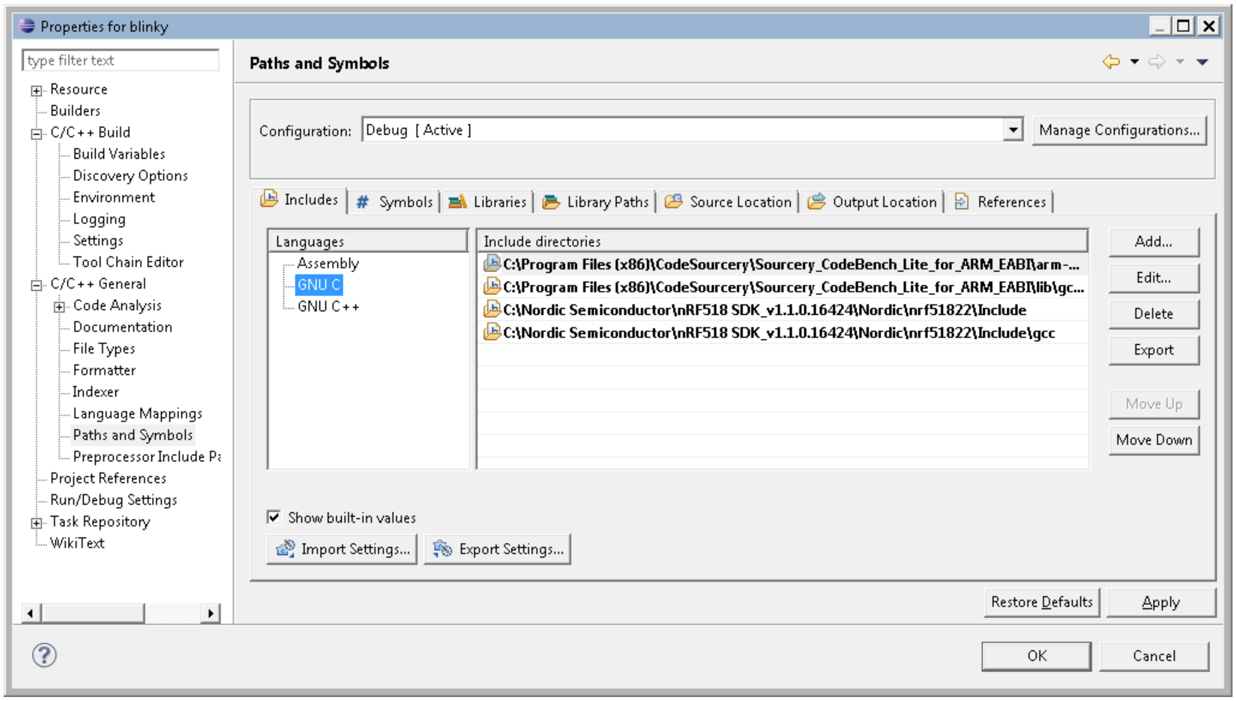
☑ Clean | clean

*Figure 4 Implementing the Nordic Build Properties*

1. In the left menu, expand **C/C++ General** and click **Paths and Symbols**.
2. Fill in the include paths to the nRF SDK and the Sourcery CodeBench as shown in ***Figure 5***.

**Note:** These directories depend on which version of Sourcery CodeBench and the Nordic Software Development Kit (SDK) you have installed. The following is an example of the directories:

- **C:\Program Files (x86)\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_EABI\arm-none-eabi\include**
- **C:\Program Files (x86)\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_EABI\lib\gcc\arm-none-eabi\4.6.3\include**
- **C:\Nordic Semiconductor\nRF518 SDK_<version>\Nordic\nrf51822\Include**
- **C:\Nordic Semiconductor\nRF518 SDK_<version>\Nordic\nrf51822\Include\gcc**



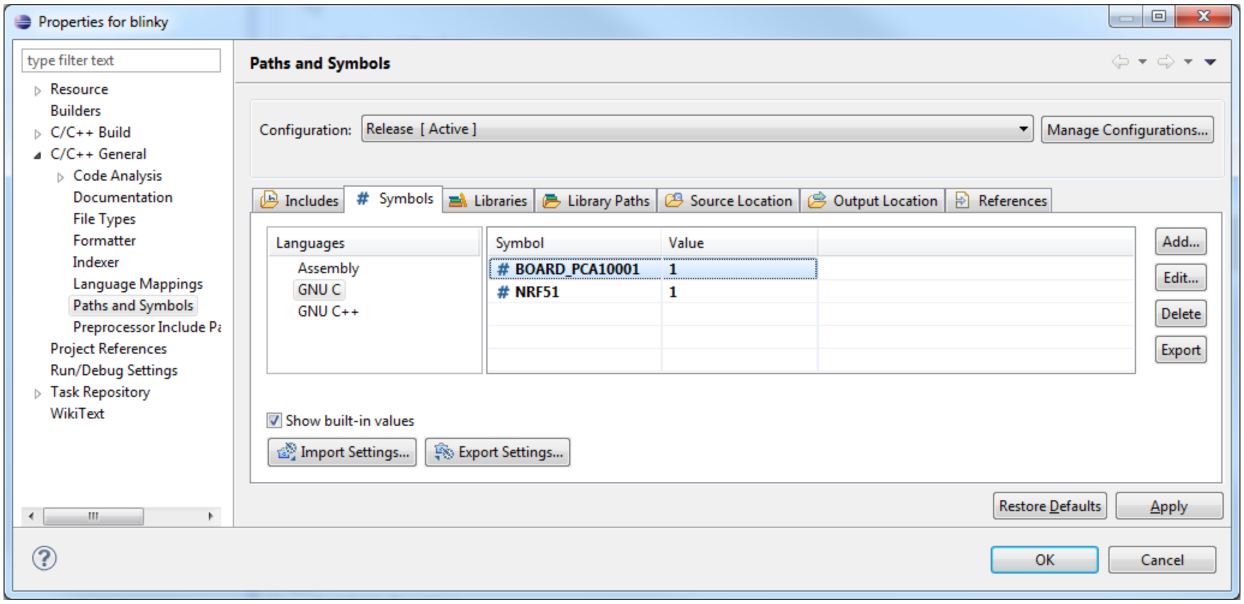
Configuration: Debug [ Active ]

Include directories
- C:\Program Files (x86)\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_EABI\arm-...
- C:\Program Files (x86)\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_EABI\lib\gc...
- C:\Nordic Semiconductor\nRF518 SDK_v1.1.0.16424\Nordic\nrf51822\Include
- C:\Nordic Semiconductor\nRF518 SDK_v1.1.0.16424\Nordic\nrf51822\Include\gcc

☑ Show built-in values

***Figure 5*** *Filling in the include paths*

3. Select the **Symbols** tab and add **NRF51**. Depending on the board you are using, add either **BOARD_NRF6310** or **BOARD_PCA10001**. See *Figure 6*.



Properties for blinky

Paths and Symbols

Configuration: Release [ Active ]

| Languages |
|---|
| Assembly |
| GNU C |
| GNU C++ |

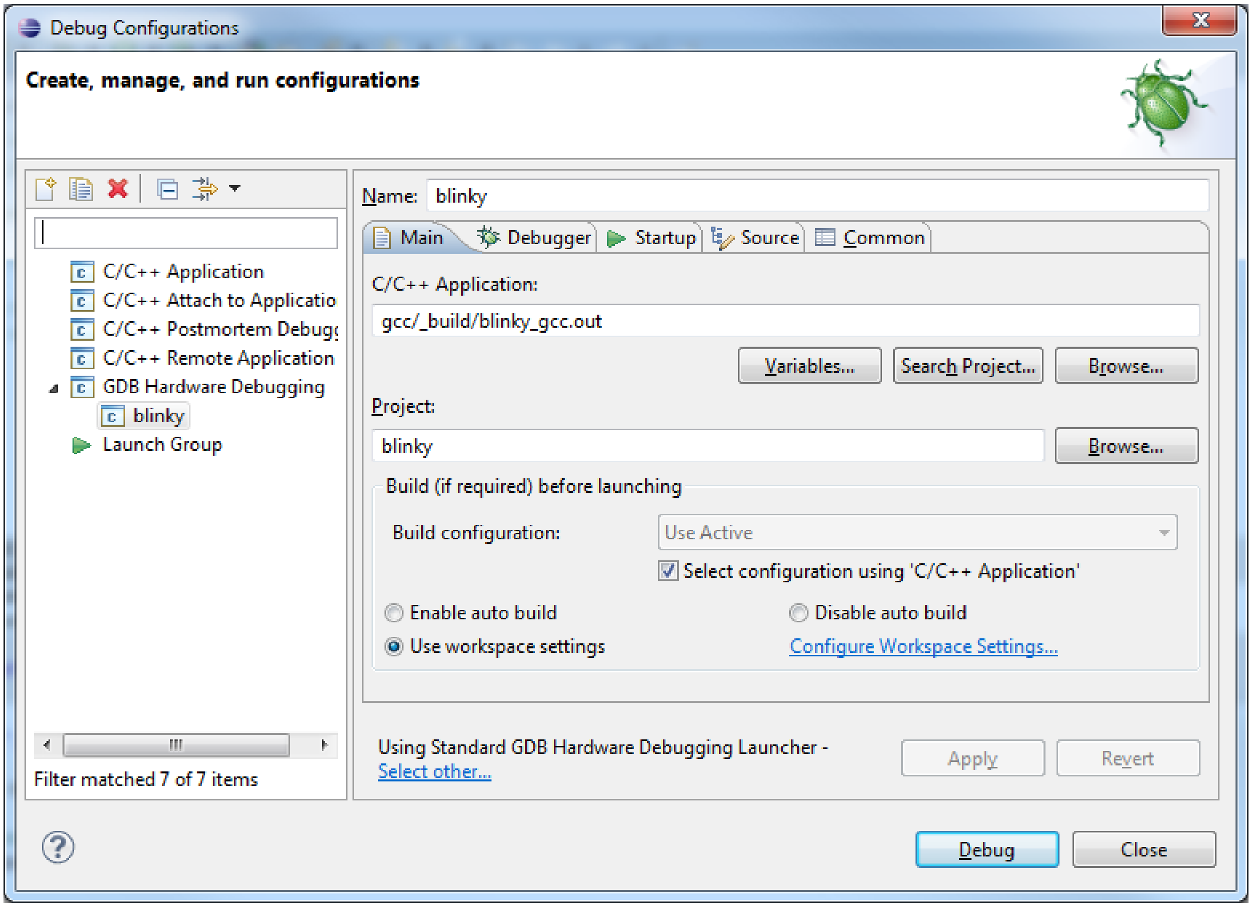
| Symbol | Value |
|---|---|
| # BOARD_PCA10001 | 1 |
| # NRF51 | 1 |

☑ Show built-in values

***Figure 6*** *Adding symbols*

# 2 Setting up a project for debugging in Eclipse

1. In the **Project Explorer** select the **blinky** project.
2. Open the **Run** menu and click **Debug Configurations**.
3. Right click **GDB Hardware Debugging** and select **New**.
4. Change the name to blinky.
5. Under C/C++ Application, click **Browse**.
6. Select the **blinky_gcc.out** file in the **_build folder** of the project.
7. Click **Select other** and change the preferred launcher to **Standard GDB Hardware Debugging Launcher**. See *Figure 7*.



Debug Configurations

Create, manage, and run configurations

Name: blinky

C/C++ Application:
gcc/_build/blinky_gcc.out

Project:
blinky

Build (if required) before launching

Build configuration: Use Active

☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

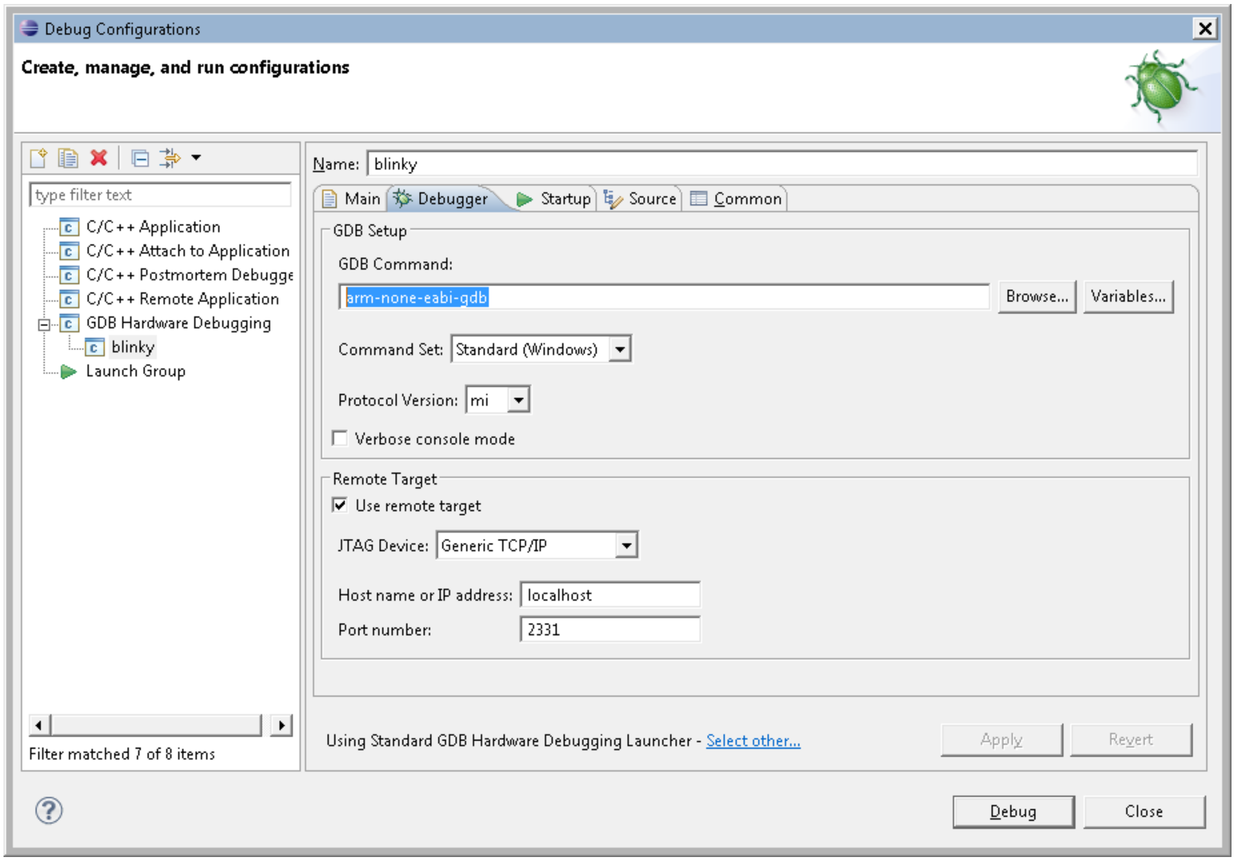
● Use workspace settings

Using Standard GDB Hardware Debugging Launcher - Select other...

Filter matched 7 of 7 items

***Figure 7*** *Debug configuration*

8. Select the **Debugger** tab. Change the **GDB Command** to **arm-none-eabi-gdb**.
9. In the Remote Target box, change the **Port number** to **2331** as shown in ***Figure 8***.



Name: blinky

GDB Command: arm-none-eabi-gdb

Command Set: Standard (Windows)

Protocol Version: mi

☐ Verbose console mode

☑ Use remote target

JTAG Device: Generic TCP/IP

Host name or IP address: localhost

Port number: 2331

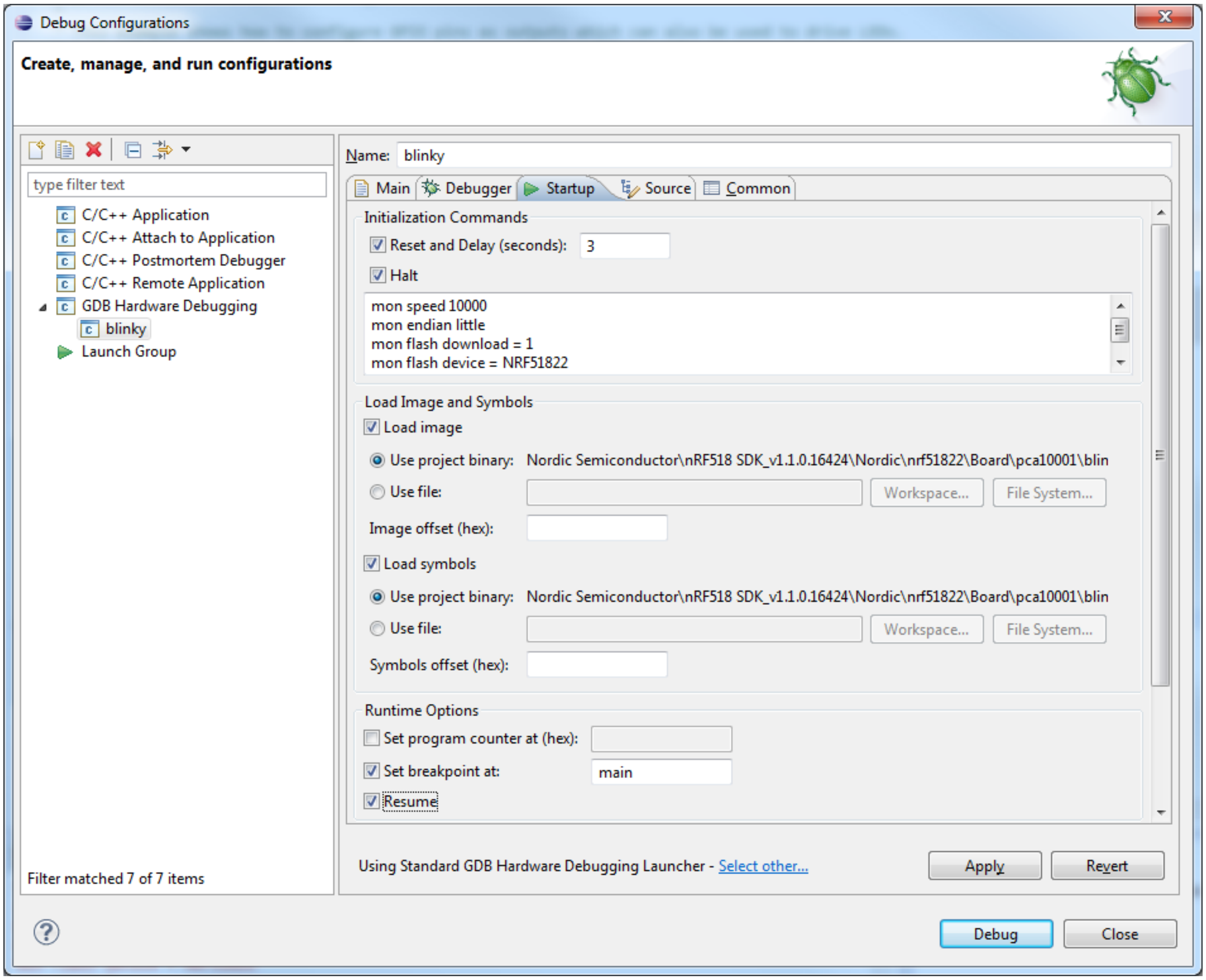
Using Standard GDB Hardware Debugging Launcher - Select other...

***Figure 8*** *Entering GDB command and TCP/IP port number*

10. Select the **Startup** tab.
11. Enter the following commands in the **Initialization Commands box**:

    - **mon speed 10000** (Sets the clock speed on the programming pins to 10 MHz)
    - **mon endian little** (Selects little endian format)
    - **mon flash download = 1** (Tells the debugger that our device supports flash download)
    - **mon flash device = NRF51822** (Sets the device to NRF51822)
    - **mon reset 0** (Selects reset type 0)
12. Check **Set breakpoint at**.
13. In the **Set breakpoint at** field type in main.
14. Check **Resume**.
15. Click **Debug** to begin debugging, as shown in ***Figure 9***.

**Note:** You must make sure the J-Link GDB server is running. A shortcut named **J-Link GDB Server via SWD** can be found under the SEGGER entry on the Start menu in Windows.



Name: blinky

Initialization Commands

☑ Reset and Delay (seconds): 3

☑ Halt

```
mon speed 10000
mon endian little
mon flash download = 1
mon flash device = NRF51822
```

Load Image and Symbols

☑ Load image

● Use project binary: Nordic Semiconductor\nRF518 SDK_v1.1.0.16424\Nordic\nrf51822\Board\pca10001\blin

○ Use file:

Image offset (hex):

☑ Load symbols

● Use project binary: Nordic Semiconductor\nRF518 SDK_v1.1.0.16424\Nordic\nrf51822\Board\pca10001\blin

○ Use file:

Symbols offset (hex):

Runtime Options

☐ Set program counter at (hex):

☑ Set breakpoint at: main

☑ Resume

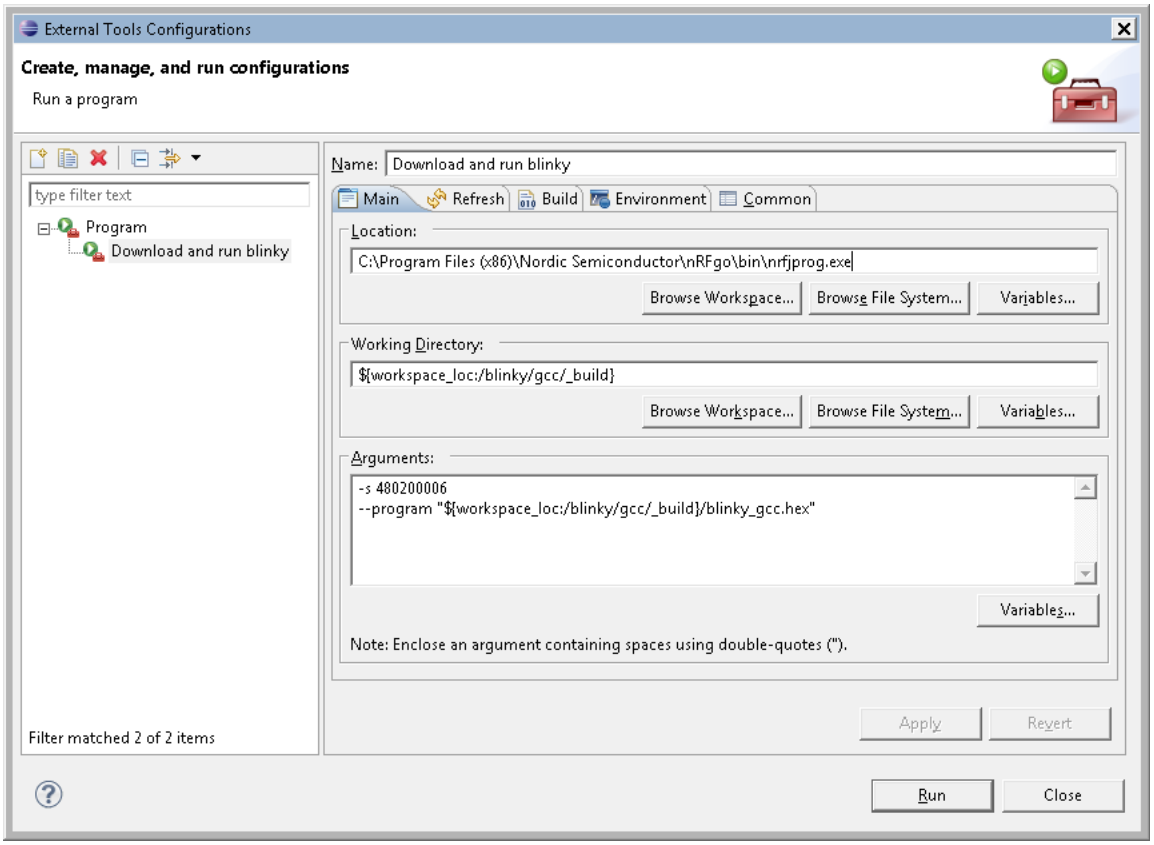
Using Standard GDB Hardware Debugging Launcher - Select other...

***Figure 9*** *Configuring initialization commands*

# 3 Flash download

Flash downloading and running a program on your device without debugging is possible by using the **nrfjprog.exe** tool directly from Eclipse. Follow these steps for flash downloading.

1. On the **Run** menu, click **External Tools**, click **External Tools Configurations**, and then click **New**.
2. In the **Name** field type **Download and run blinky** as the name for the configuration.
3. In the **Location** field type the full path to **nrfjprog.exe** (default **C:\Program Files (x86)\Nordic Semiconductor\nrf51\bin\nrfjprog.exe**).
4. In the **Working Directory** field select the output folder for blinky (${workspace_loc:/blinky/gcc/_build}).
5. In the **Arguments** field enter **--program "${workspace_loc:/blinky/gcc/_build}/blinky_gcc.hex**.
6. If you have more than one board connected, you can also select the board by defining the serial number with the –s option. See *Figure 10*.
7. When you are finished click **Apply** and close the dialog box. The command can now be found on the Run menu.



External Tools Configurations

**Create, manage, and run configurations**

Run a program

Name: Download and run blinky

Location:

C:\Program Files (x86)\Nordic Semiconductor\nRFgo\bin\nrfjprog.exe

Working Directory:

${workspace_loc:/blinky/gcc/_build}

Arguments:

```
-s 480200006
--program "${workspace_loc:/blinky/gcc/_build}/blinky_gcc.hex"
```

Note: Enclose an argument containing spaces using double-quotes (").

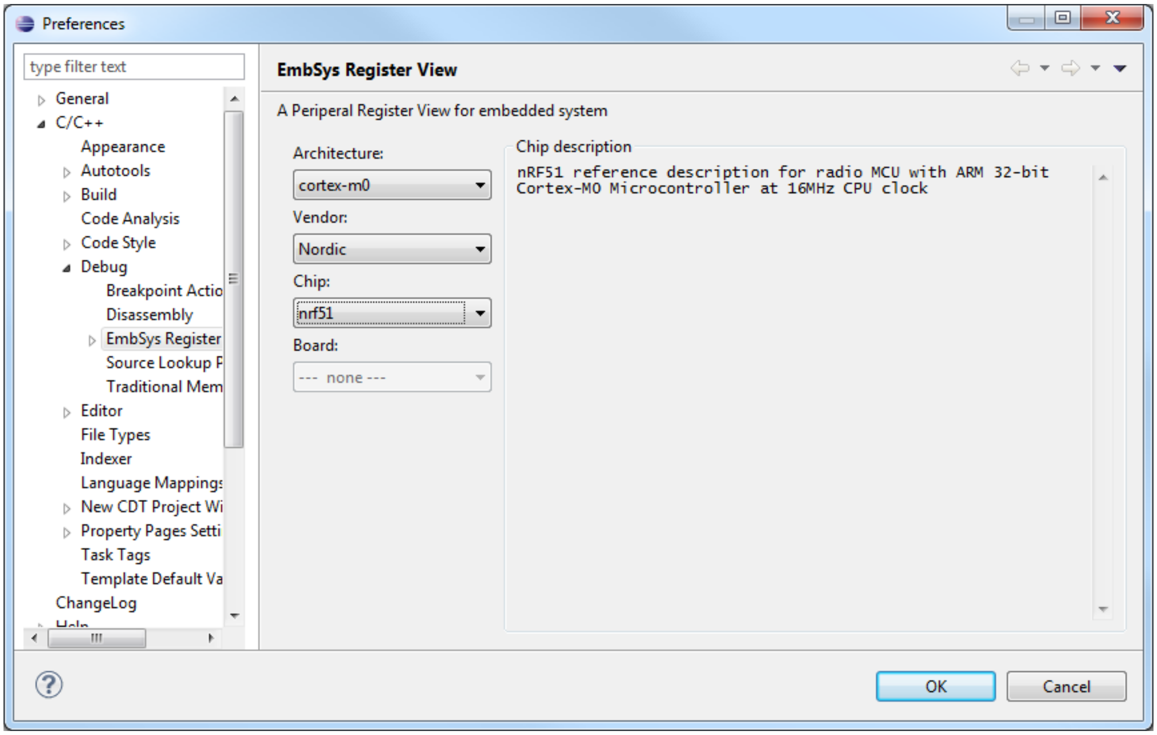
Filter matched 2 of 2 items

*Figure 10 External tool configuration*

# 4 Enabling the EmbSys Register Viewer

EmbSys Register Viewer is a plug-in for Eclipse that enables you to monitor and modify the peripheral registers on the chip you are debugging. For more information see http://embsysregview.sourceforge.net/.

1. Open the Window menu in Eclipse and click **Preferences**.
2. In the left navigation bar expand **C/C++**, then expand **Debug.**
3. Select **EmbSys Register View**.
4. In the **Architecture** drop down, select **cortex-m0**.
5. In the **Vendor** drop down, select **Nordic**.
6. In the **Chip** drop down, select **nrf51**. See *Figure 11*.



EmbSys Register View

A Periperal Register View for embedded system

Architecture: cortex-m0

Vendor: Nordic

Chip: nrf51

Board: --- none ---

Chip description

```
nRF51 reference description for radio MCU with ARM 32-bit
Cortex-M0 Microcontroller at 16MHz CPU clock
```

*Figure 11 Configuring Emb Sys Register view*

**Note:** To show the EmbSys Register View when debugging:

- On the Window menu select **Show View**, then click **Other**, then select **EmSys Registers**.

## 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

Nordic Semiconductor's 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.
Information regarding product updates, downloads, and technical support can be accessed through your My Page account on our homepage.

**Main office:** Otto Nielsens veg 12
7052 Trondheim
Norway

Phone: +47 72 89 89 00
Fax: +47 72 89 89 89

**Mailing address:** Nordic Semiconductor
P.O. Box 2336
7004 Trondheim
Norway

## Revision History

| Date | Version | Description |
|---|---|---|
| November 2012 | 1.0 | First release |