Document Number: KSDK20LPC5411XGSUG
Rev. 0, 06/2016

Getting Started with SDK v.2.0 for LPC5411x Derivatives

1 Overview

The Software Development Kit (SDK) provides comprehensive software support for Microcontrollers. The SDK includes a flexible set of peripheral drivers designed to speed up and simplify development of embedded applications. Along with the peripheral drivers, the SDK provides an extensive and rich set of example applications covering everything from basic peripheral use case examples to full demo applications. The SDK also contains RTOS kernels, a USB host and device stack, and various other middleware to support rapid development on devices.

For supported toolchain versions, see the SDK v.2.0.0 Release Notes for LPC5411x Derivatives (document KSDK20LPC5411XGSUG).

For the latest version of this and other SDK documents, see the SDK homepage www.nxp.com/ksdk

Contents

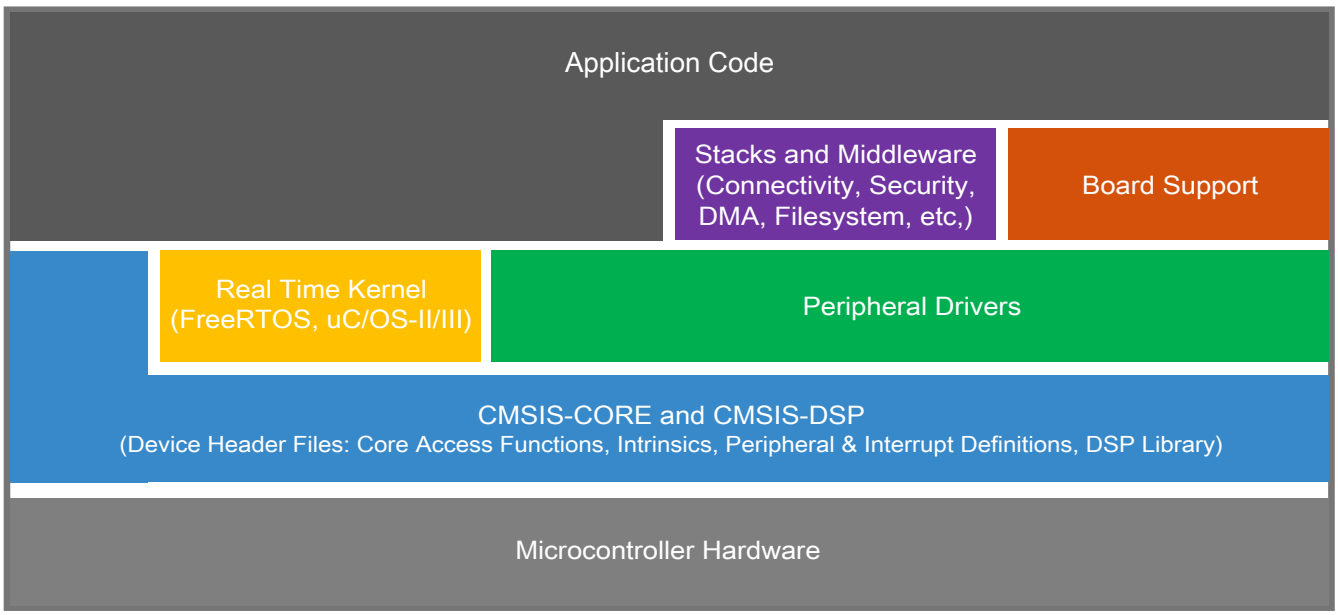

Figure 1. SDK layers

2 SDK Board Support Folders

SDK board support provides example applications for development and evaluation boards. Board support packages are found inside of the top level boards folder, and each supported board has its own folder (a SDK package can support multiple boards). Within each <board_name> folder there are various sub-folders to classify the type of examples they contain. These include (but are not limited to):

- demo_apps: Full-featured applications intended to highlight key functionality and use cases of the target MCU. These applications typically use multiple MCU peripherals and may leverage stacks and middleware.
- driver_examples: Simple applications intended to concisely illustrate how to use the SDK's peripheral drivers for a single use case. These applications typically only use a single peripheral, but there are cases where multiple are used (for example, ADC conversion using DMA).
- rtos_examples: Basic FreeRTOS examples showcasing the use of various RTOS objects (semaphores, queues, and so on) and interfacing with the SDK's RTOS drivers
- usb_examples: Applications that use the USB host/device/OTG stack.

2.1 Example Application Structure

This section describes how the various types of example applications interact with the other components in the SDK. To get a comprehensive understanding of all SDK components and folder structure, see the *Kinetis SDK v.2.0 API Reference Manual* document (KSDK20APIRM).

Each <board_name> folder in the boards directory contains a comprehensive set of examples that are relevant to that specific piece of hardware. We'll discuss the hello_world example (part of the demo_apps folder), but the same general rules apply to any type of example in the <board_name> folder.

In the hello_world application folder you see this:

All files in the application folder are specific to that example, so it's very easy to copy-paste an existing example to start developing a custom application based on a project provided in the SDK.

2.2 Locating Example Application Source Files

When opening an example application in any of the supported IDEs, there are a variety of source files referenced. The SDK *devices* folder is designed to be the "golden core" of the application and is, therefore, the central component to all example applications. Because it's a core component, all of the examples reference the same source files and, if one of these files is modified, it could potentially impact the behavior of other examples.

The main areas of the SDK tree used in all example applications are:

- devices/<device_name>: The device's CMSIS header file, SDK feature file and a few other things.
- devices/<device_name>/drivers: All of the peripheral drivers for your specific MCU.
- devices/<device_name>/<tool_name>: Toolchain-specific startup code. Vector table definitions are here.
- devices/<device_name>/utilities: Items such as the debug console that are used by many of the example applications.

For examples containing middleware/stacks and/or a RTOS, there will be references to the appropriate source code. Middleware source files are located in the *middleware* folder and RTOSes are in the *rtos* folder. Again, the core files of each of these are shared, so modifying them could have potential impacts on other projects that depend on them.

3 Run an example application

To download and run the application, perform these steps:

1. Download and install LPCScrypt or the Windows operating systems driver for LPCXpresso boards from http://www.nxp.com/lpcutilities. This installs required drivers for the board.
2. Connect the development platform to your PC via USB cable between the Link2 USB connector (named Link for some boards) and the PC USB connector. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug COM port (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
   a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in board.h file)
   b. No parity
   c. 8 data bits
   d. 1 stop bit

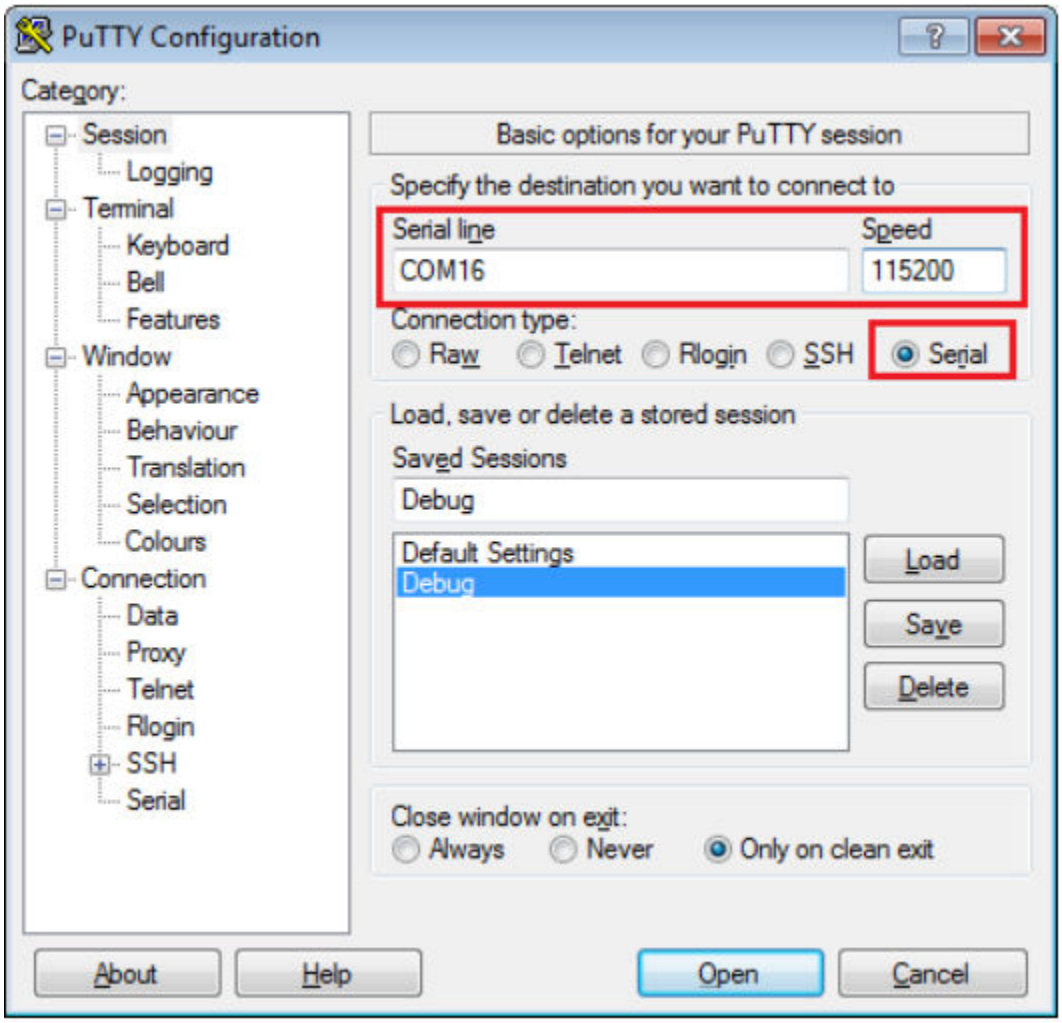

Figure 2. Terminal (PuTTY) configuration

4. In IAR, click the "Download and Debug" button to download the application to the target.

Figure 3. Download and Debug button

5. The application is then downloaded to the target and automatically runs to the main() function.

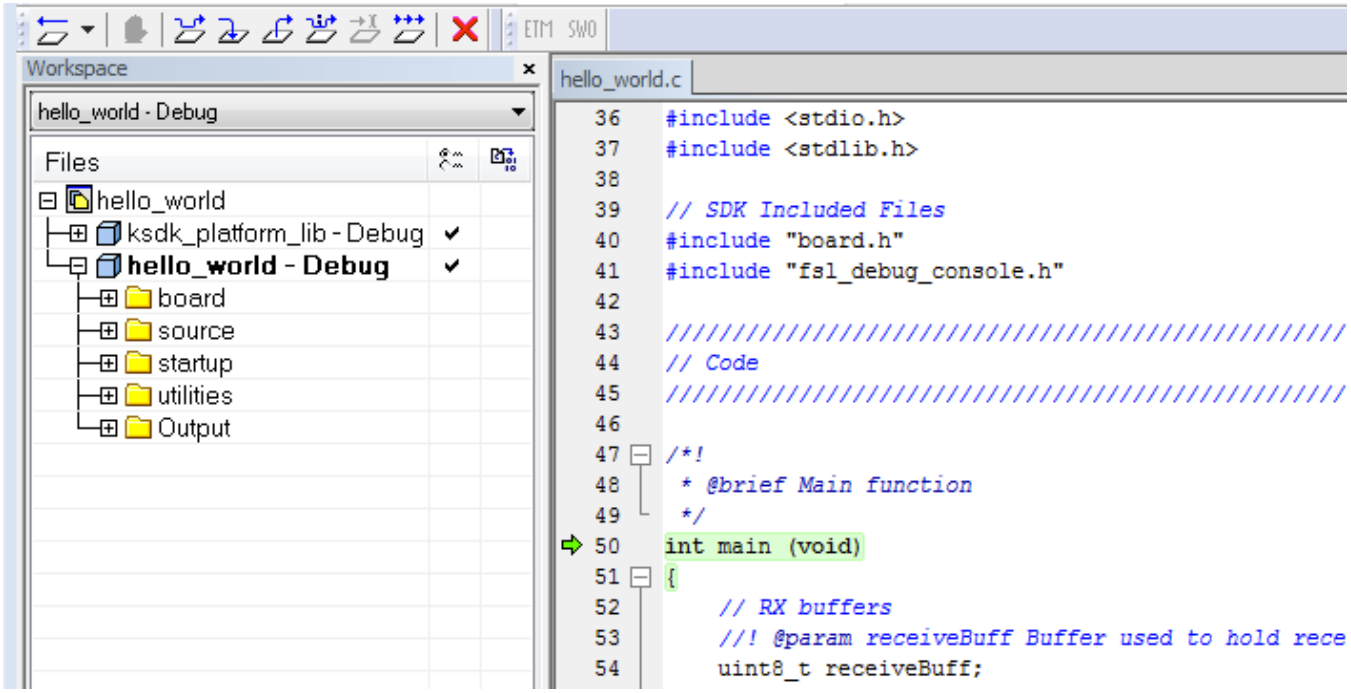

Figure 4. Stop at main() when running debugging

6. Run the code by clicking the "Go" button to start the application.

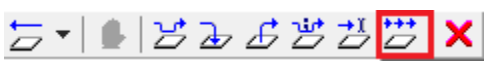

Figure 5. Go button

7. The hello_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.

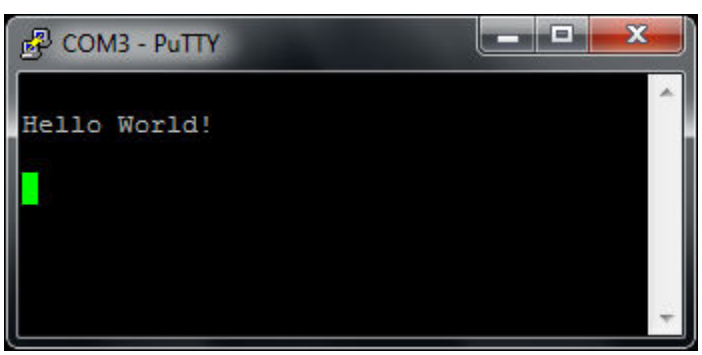

Figure 6. Text display of the hello_world demo

3.1 Build an example application

The following steps guide you through opening the hello_world example application. These steps may change slightly for other example applications as some of these applications may have additional layers of folders in their path.

1. If not already done, open the desired demo application workspace. Most example application workspace files can be located using the following path:

   <install_dir>/boards/<board_name>/<example_type>/<application_name>/iar

Using the LPCXpresso54114 hardware platform as as an example, the hello_world workspace is located in *<install_dir>/boards/lpcxpresso54114/demo_apps/hello_world/iar/hello_world.eww*

2. Select the desired build target from the drop-down. For this example, select the “hello_world – Debug” target.

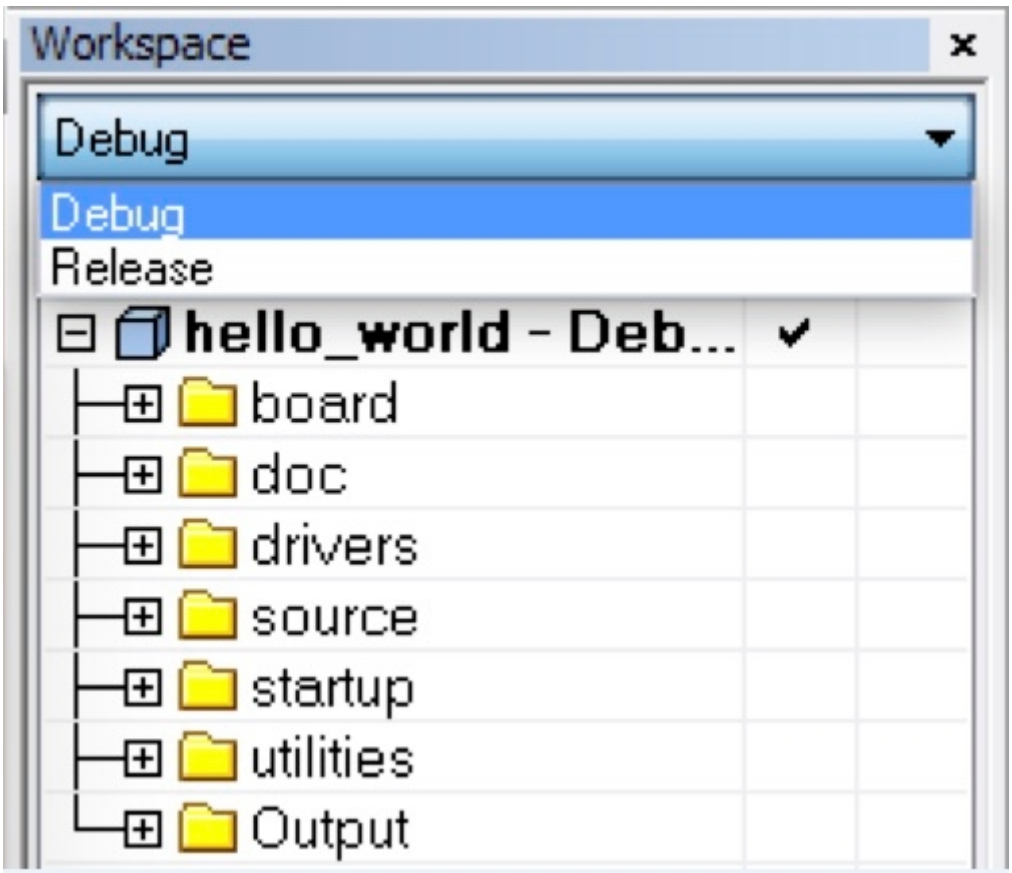

Figure 7. Demo build target selection

3. To build the demo application, click the “Make” button, highlighted in red below.

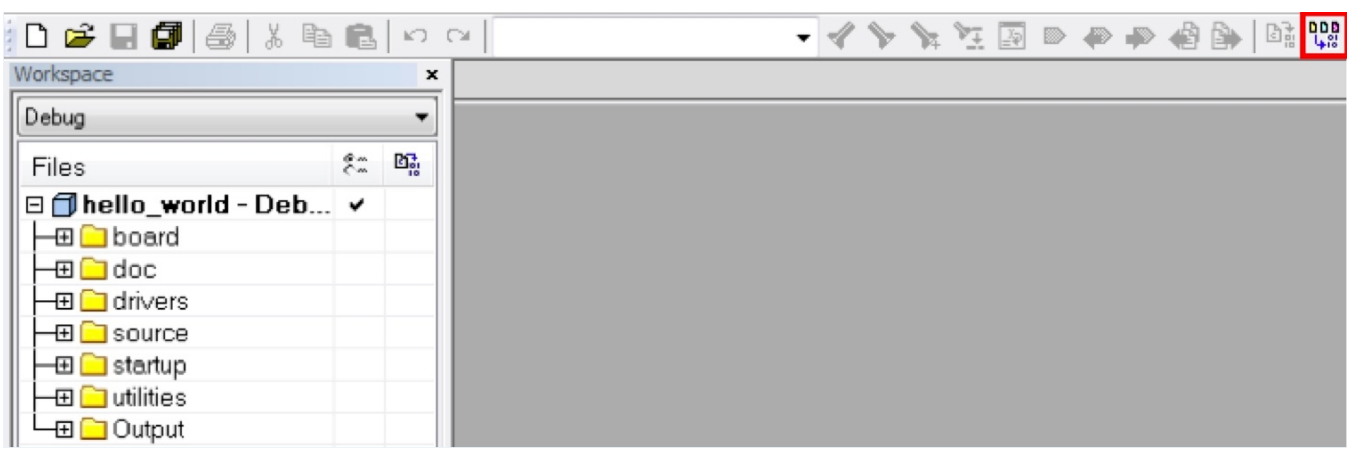

Figure 8. Build the demo application

4. The build completes without errors.

3.2 Run an example application

To download and run the application, perform these steps:

1. Reference the table in Appendix B to determine the debug interface that comes loaded on your specific hardware platform.
   - For boards with CMSIS-DAP/mbed/DAPLink interfaces, visit developer.mbed.org/handbook/Windows-serial-configuration and follow the instructions to install the Windows® operating system serial driver.
   - For boards with P&E Micro interfaces, visit www.pemicro.com/support/downloads_find.cfm and download the P&E Micro Hardware Interface Drivers package.
   - If using J-Link either a standalone debug pod or OpenSDA, install SEGGER software located in <KSDK install_dir>/tools/Setup_JLink_V501jupdate PKE18.zip.
2. Connect the development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug COM port (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
   a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in board.h file)
   b. No parity
   c. 8 data bits
   d. 1 stop bit

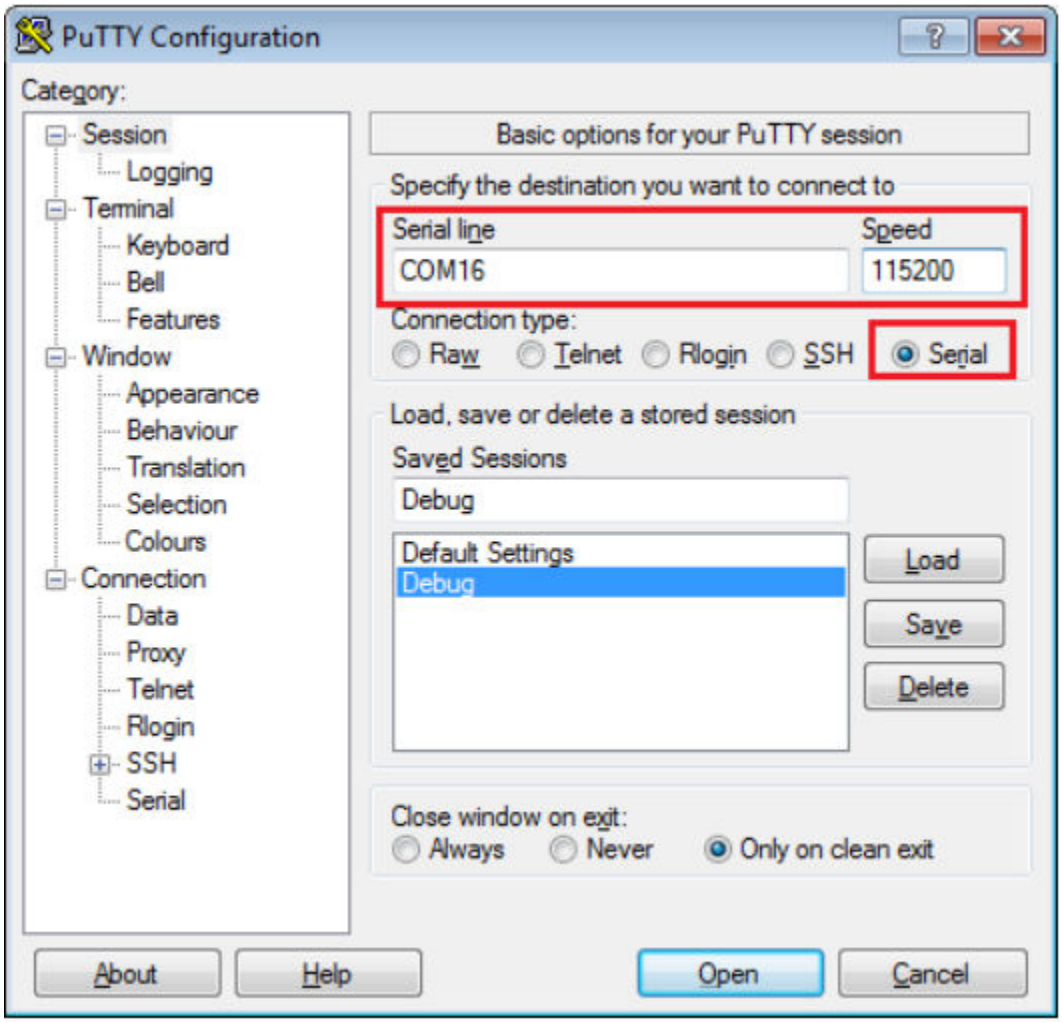

Figure 9. Terminal (PuTTY) configuration

4. In IAR, click the "Download and Debug" button to download the application to the target.

Figure 10. Download and Debug button

5. The application is then downloaded to the target and automatically runs to the main() function.

6. Run the code by clicking the "Go" button to start the application.

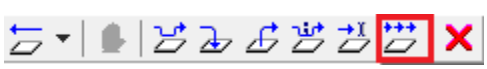

Figure 11. Go button

7. The hello_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.

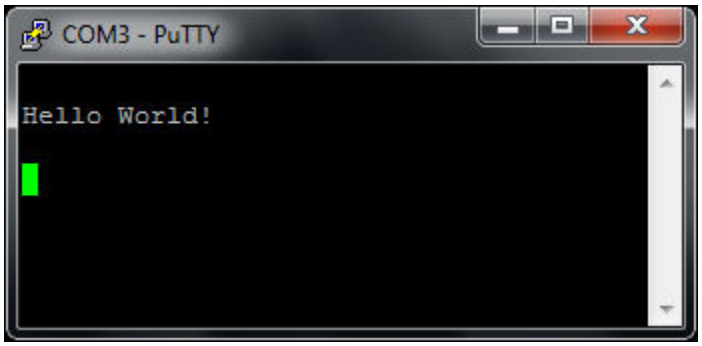

Figure 12. Text display of the hello_world demo

4 Run a demo using Keil® MDK/µVision

This section describes the steps required to build, run, and debug example applications provided in the SDK. The hello_world demo application targeted for the LPCXpresso54114 hardware platform is used as an example, although these steps can be applied to any demo or example application in the SDK.

4.1 Install CMSIS device pack

After the MDK tools are installed, Cortex Microcontroller Software Interface Standard (CMSIS) device packs must be installed to fully support the device from a debug perspective. These packs include things such as memory map information, register definitions and flash programming algorithms. Follow these steps to install the appropriate CMSIS pack.

1. Open the MDK IDE, which is called µVision. In the IDE, select the “Pack Installer” icon.

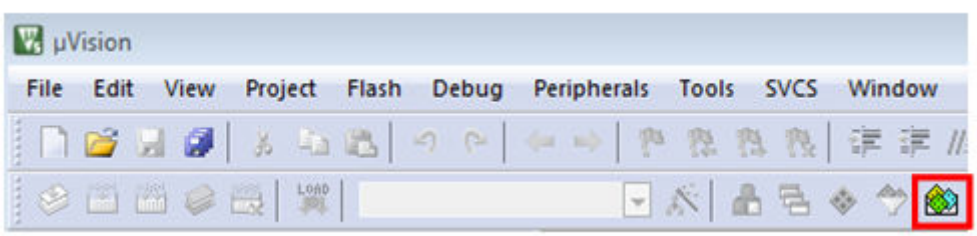

Figure 13. Launch the Pack installer

2. After the installation finishes, close the Pack Installer window and return to the µVision IDE.

4.2 Build an example application

- If not already done, open the desired example application workspace in: <install_dir>/boards/<board_name>/<*example_type*>/<application_name>/mdk

  The workspace file is named <demo_name>.uvmpw, so for this specific example, the actual path is:

  <install_dir>/boards/lpcxpresso54114/demo_apps/hello_world/mdk/hello_world.uvmpw
- To build the demo project, select the "Rebuild" button, highlighted in red.

Figure 14. Build the demo

- The build completes without errors.

4.3 Run an example application

To download and run the application, perform these steps:

1. Download and install LPCScrypt or the Windows driver for LPCXpresso boards from http://www.nxp.com/lpcutilities. This installs required drivers for the board.
2. Connect the development platform to your PC via USB cable between the Link2 USB connector (named Link on some boards) and the PC USB connector. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
3. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). Configure the terminal with these settings:
   a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in board.h file)
   b. No parity
   c. 8 data bits
   d. 1 stop bit

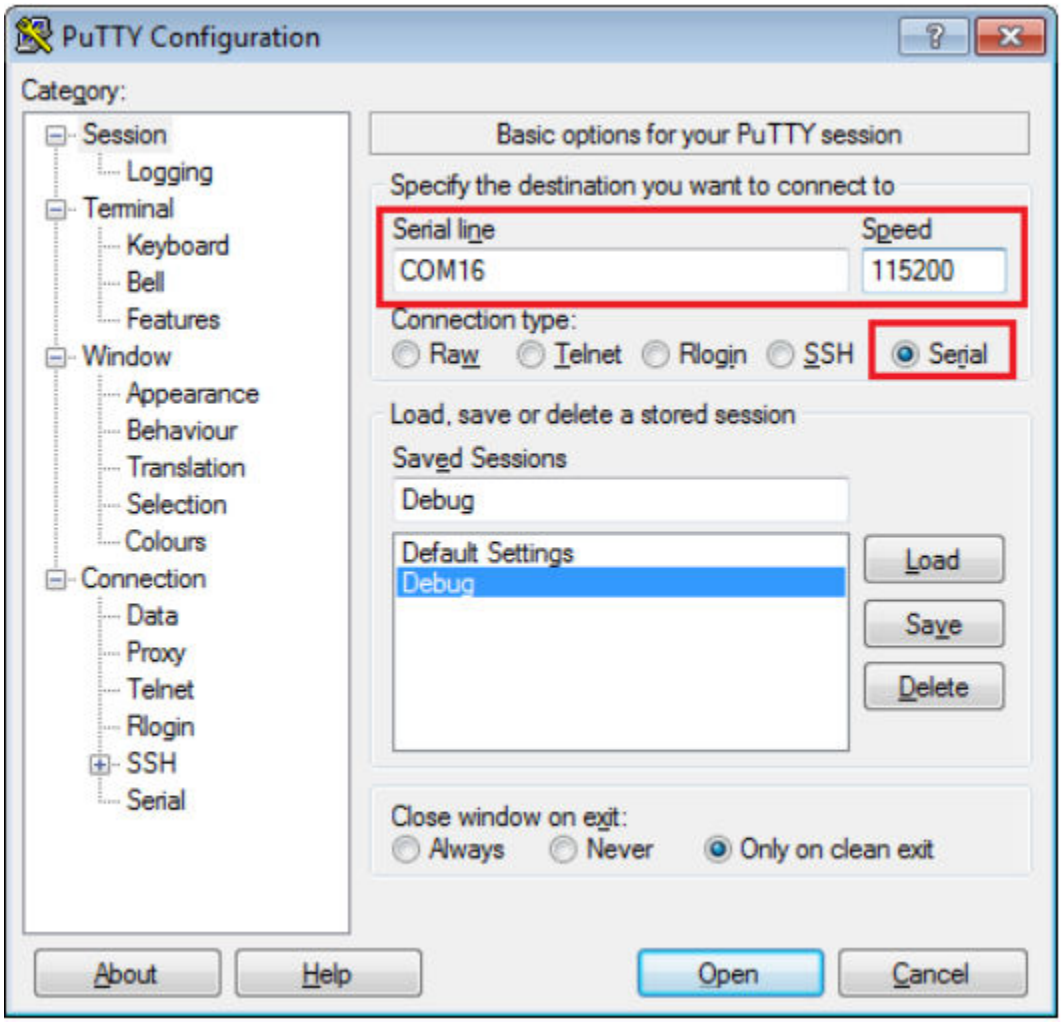

Figure 15. Terminal (PuTTY) configurations

4. In µVision, after the application is properly built, click the "Download" button to download the application to the target.

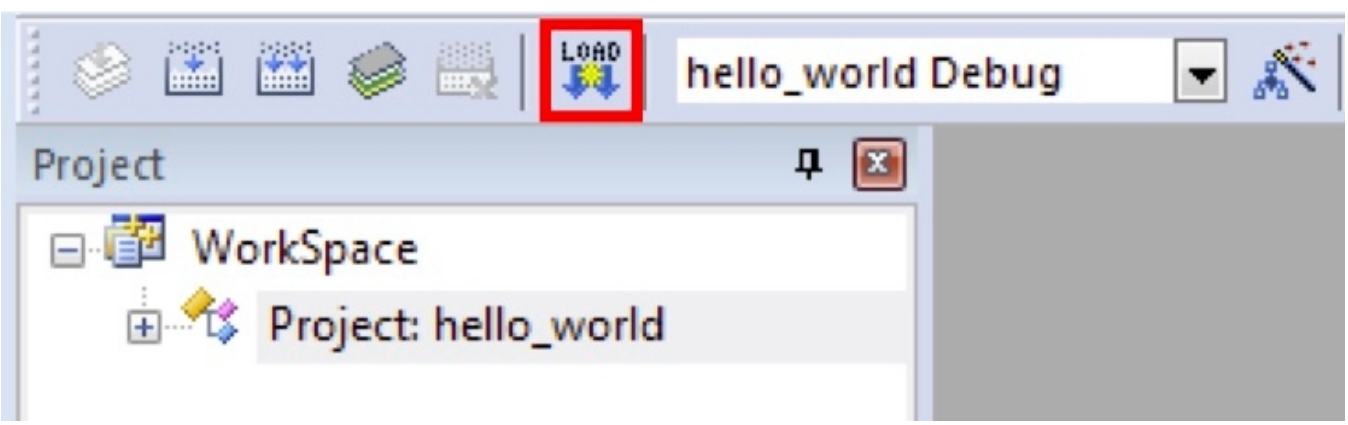

Figure 16. Download button

5. After clicking the “Download” button, the application downloads to the target and should be running. To debug the application, click the “Start/Stop Debug Session” button, highlighted in red.

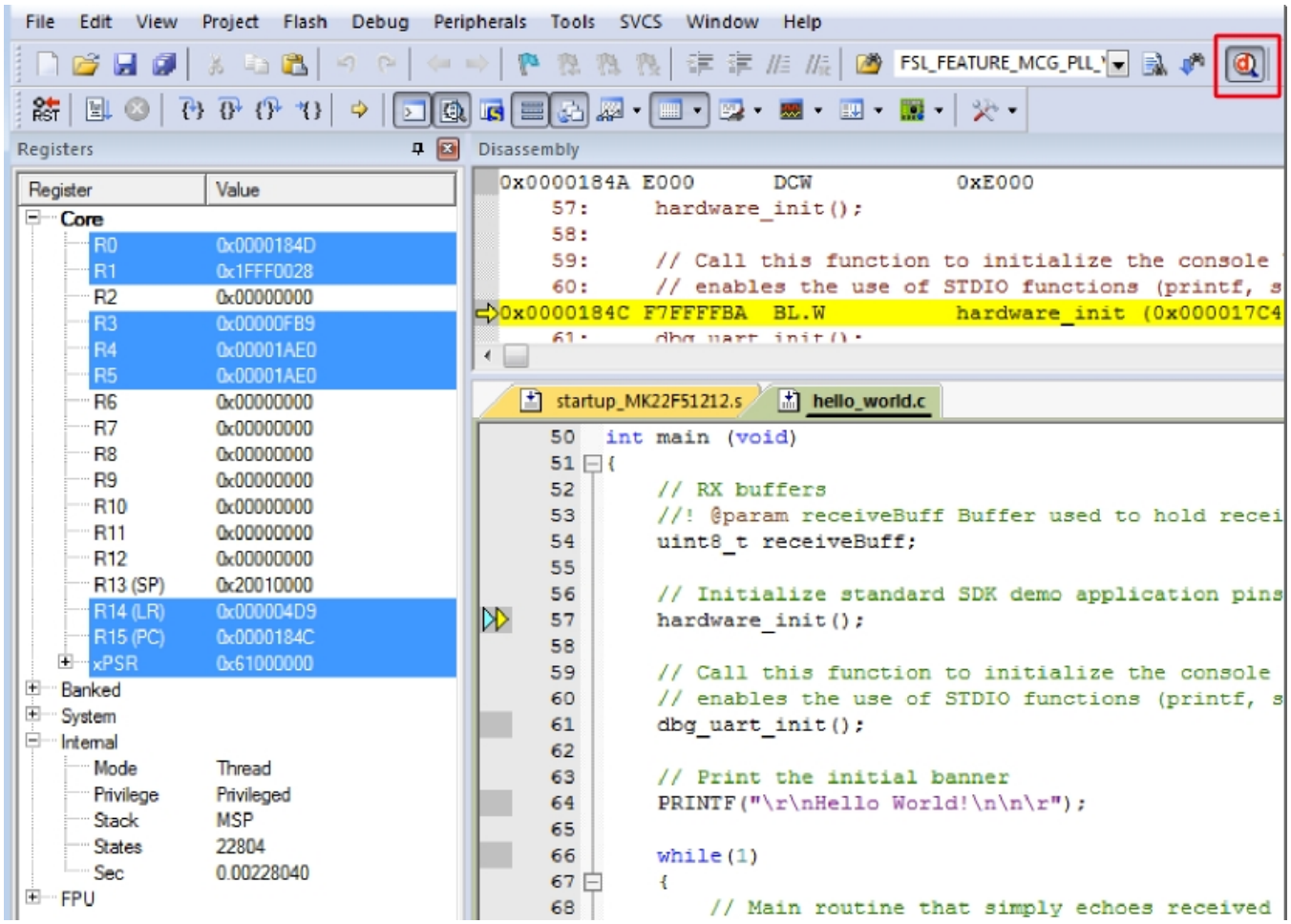

Figure 17. Stop at main() when run debugging

6. Run the code by clicking the “Run” button to start the application.

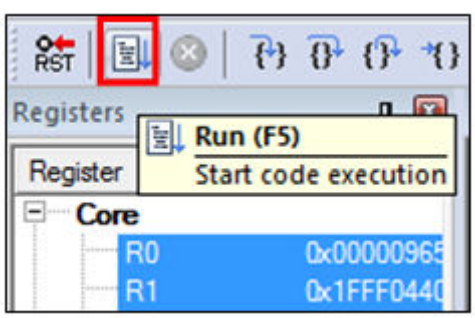

Figure 18. Go button

The hello_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.

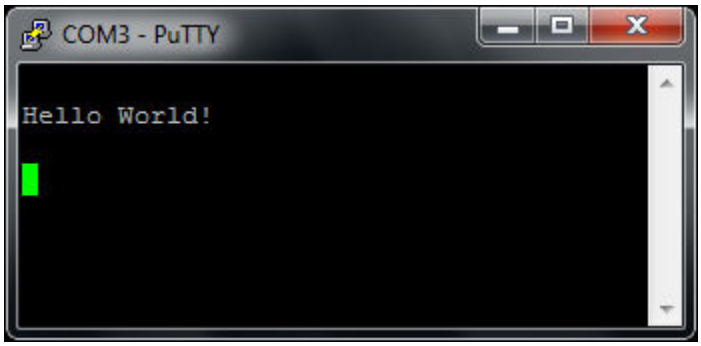

Figure 19. Text display of the hello_world demo

5 Run a demo using LPCXpresso

NOTE

Ensure that you selected the LPCXpresso IDE toolchain when you generate the SDK Package.

This section describes the steps required to configure LPCXpresso IDE to build, run, and debug example applications. The hello_world demo application targeted for the LPCXpresso54114 is used as an example, though these steps can be applied to any example application in the SDK.

5.1 Build an example application

NOTE

The steps required for the Linux® OS and Mac® OS are identical to those for the Windows® operating system. The only difference is that the IDE looks slightly different.

1. Select "File -> Import" from the KDS IDE menu. In the window that appears, expand the "General" folder and select "Existing Projects into Workspace". Then, click the "Next" button.

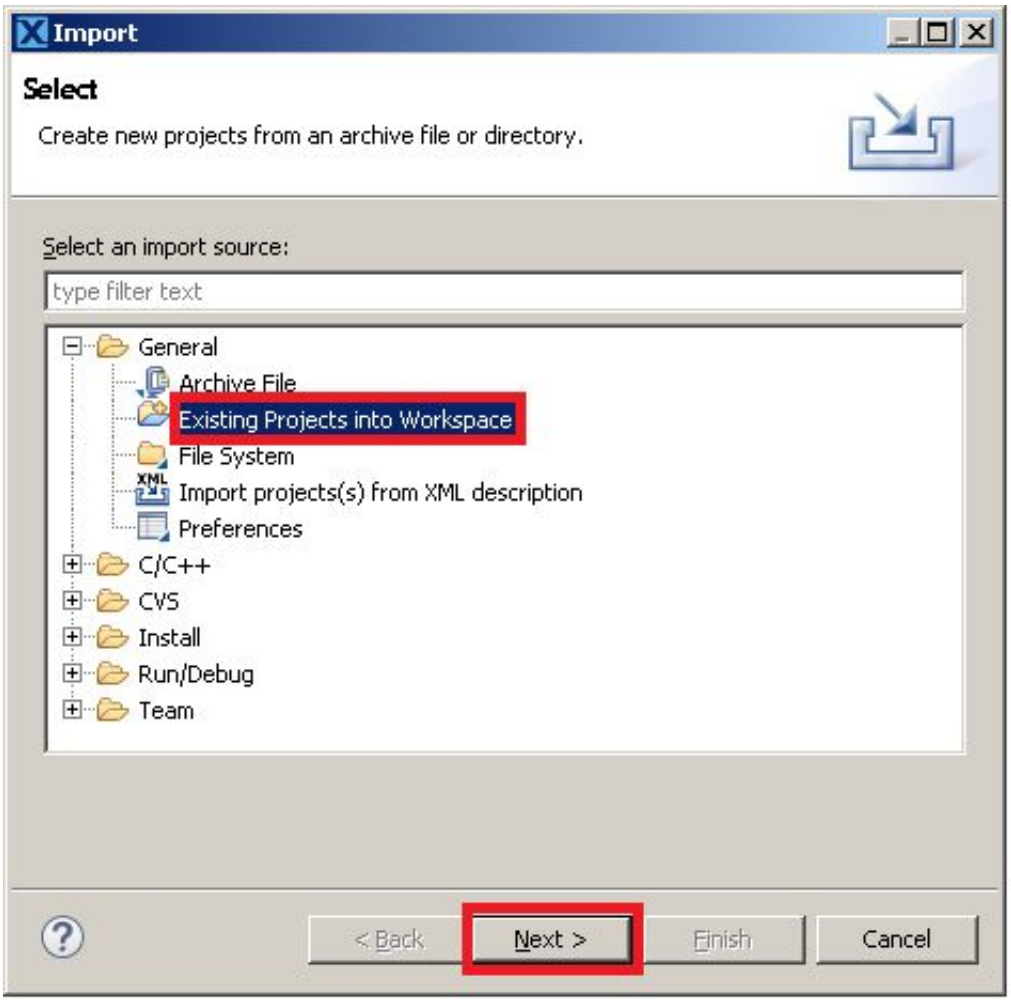

Figure 20. Selection of the correct import type in KDS IDE

2. Click the "Browse" button next to the "Import from file:" option.

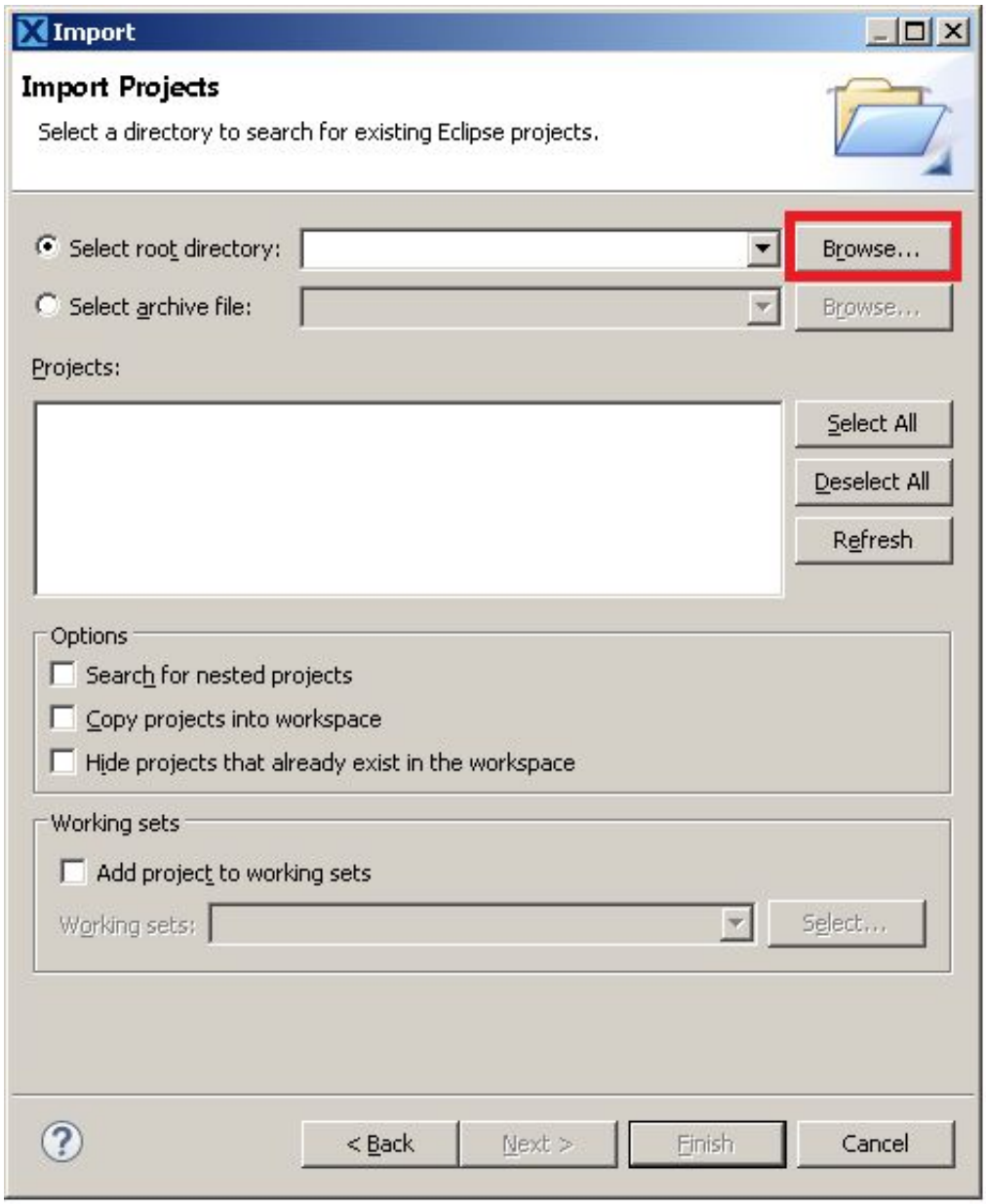

Figure 21. Import project window

3. Point to the example application project, which can be found using this path:

   <install_dir>/boards/<board_name>/<example_type>/<application_name>/lpcx

   For this example, the specific location is:

   <install_dir>/boards/lpcxpresso54114f/demo_apps/hello_world/lpcx

4. After pointing to the correct directory, your "Import Projects" window should look like the figure below. Click the "Finish" button.

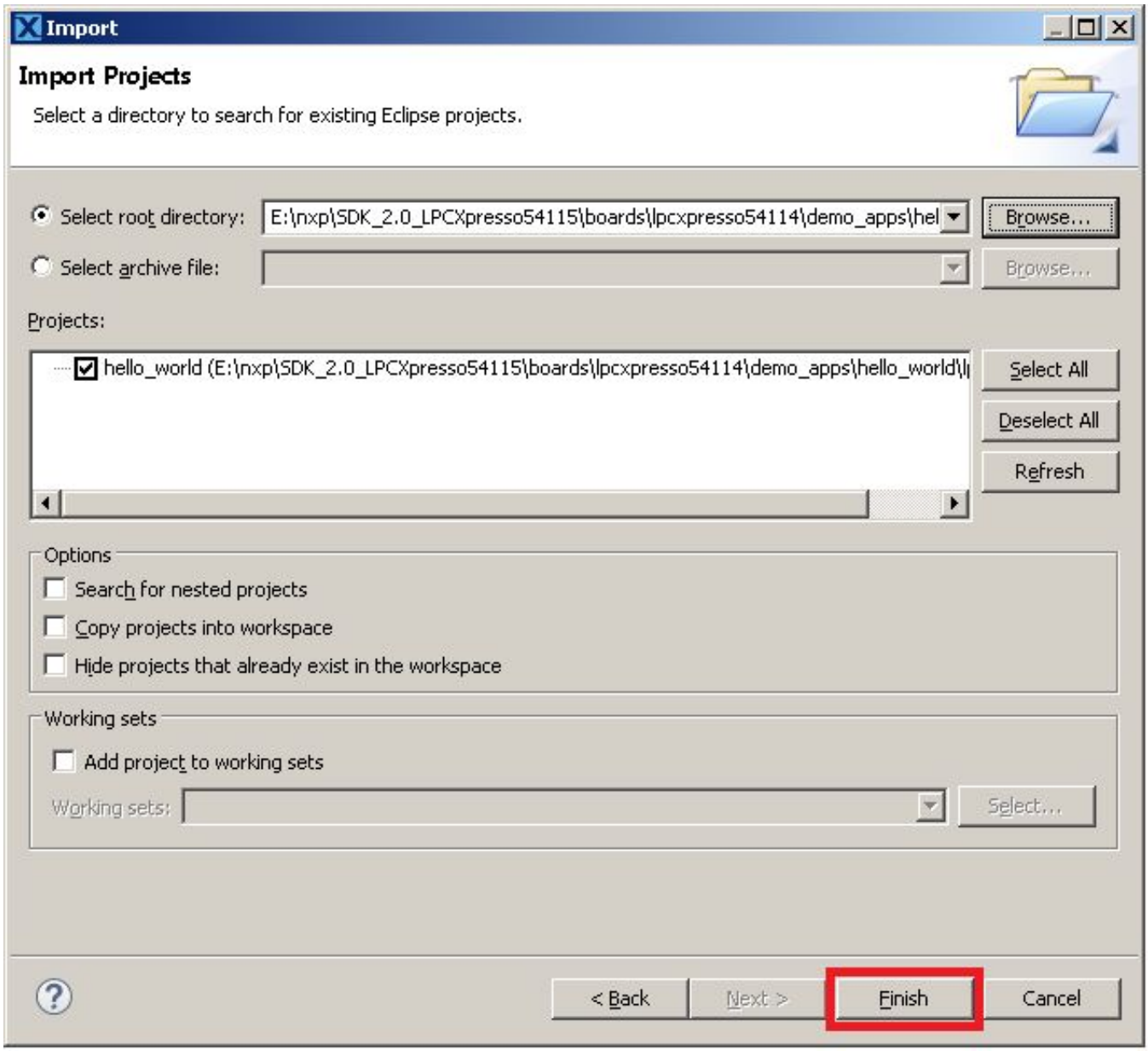

Figure 22. Select project

5. There are two project configurations (build targets) supported for each KSDK project:
   - Debug – Compiler optimization is set to low, and debug information is generated for the executable. This target should be selected for development and debug.
   - Release – Compiler optimization is set to high, and debug information is not generated. This target should be selected for final application deployment.
6. Choose the appropriate build target, "Debug" or "Release", by clicking the downward facing arrow next to the hammer icon, as shown below. For this example, select the "Debug" target.

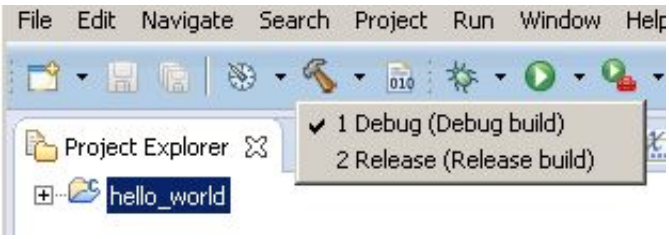

Figure 23. Selection of the build target in LPCXpresso IDE

The library starts building after the build target is selected. To rebuild the project in the future, click the hammer icon (assuming the same build target is chosen).

5.2 Run an example application

NOTE

The steps required for the Linux OS and Mac OS are identical to those for the Windows operating system. The only difference is that the IDE looks slightly different.

NOTE

If the jumper making DFUlink is installed on the board (JP5 on some boards, but consult the board user manual or schematic for specific jumper number), Link2 debug probe boots to DFU mode and LPCXpresso IDE automatically downloads the CMSIS-DAP firmware to the probe. Using DFU mode ensures the most up-to-date/compatible firmware is used with LPCXpresso IDE.

To download and run the application, perform these steps:

1. Connect the development platform to your PC via USB cable between the Link2 USB connector (named Link for some boards) and the PC USB connector. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
2. In the Windows operating system environment, open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the debug serial port number (to determine the COM port number, see Appendix A). For Linux OS, open your terminal application and connect to the appropriate device.

   Configure the terminal with these settings:

   a. 115200 or 9600 baud rate, depending on your board (reference BOARD_DEBUG_UART_BAUDRATE variable in board.h file)
   b. No parity
   c. 8 data bits
   d. 1 stop bit

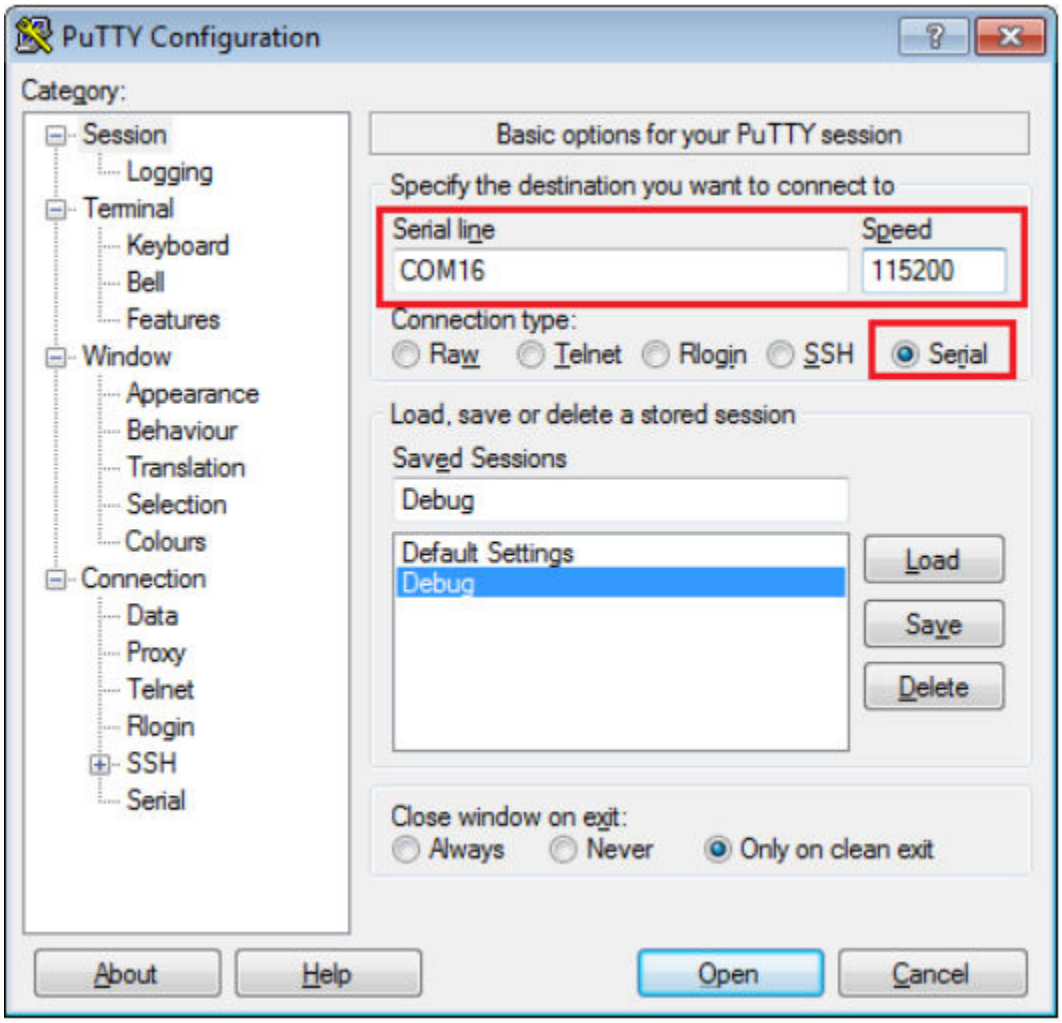

Figure 24. Terminal (PuTTY) configurations

3. In LPCXpresso IDE, ensure that the debugger configuration is correct for the target you're attempting to connect to.
   a. To check the available debugger configurations, click the small downward arrow next to the green "Debug" button and select "Debug Configurations".

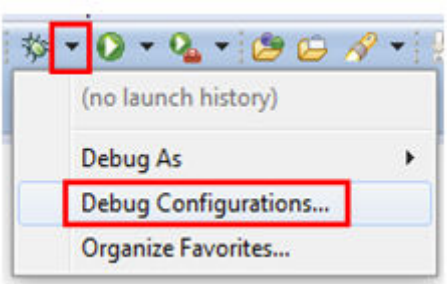

Figure 25. Debug Configurations dialog button

   b. In the Debug Configurations dialog box, select the debug configuration that corresponds to the hardware platform you're using. In this example, select is the CMSIS-DAP option under C/C++ (NXP Semiconductors) MCU Application.

      After selecting the debugger interface, click the "Debug" button to launch the debugger.

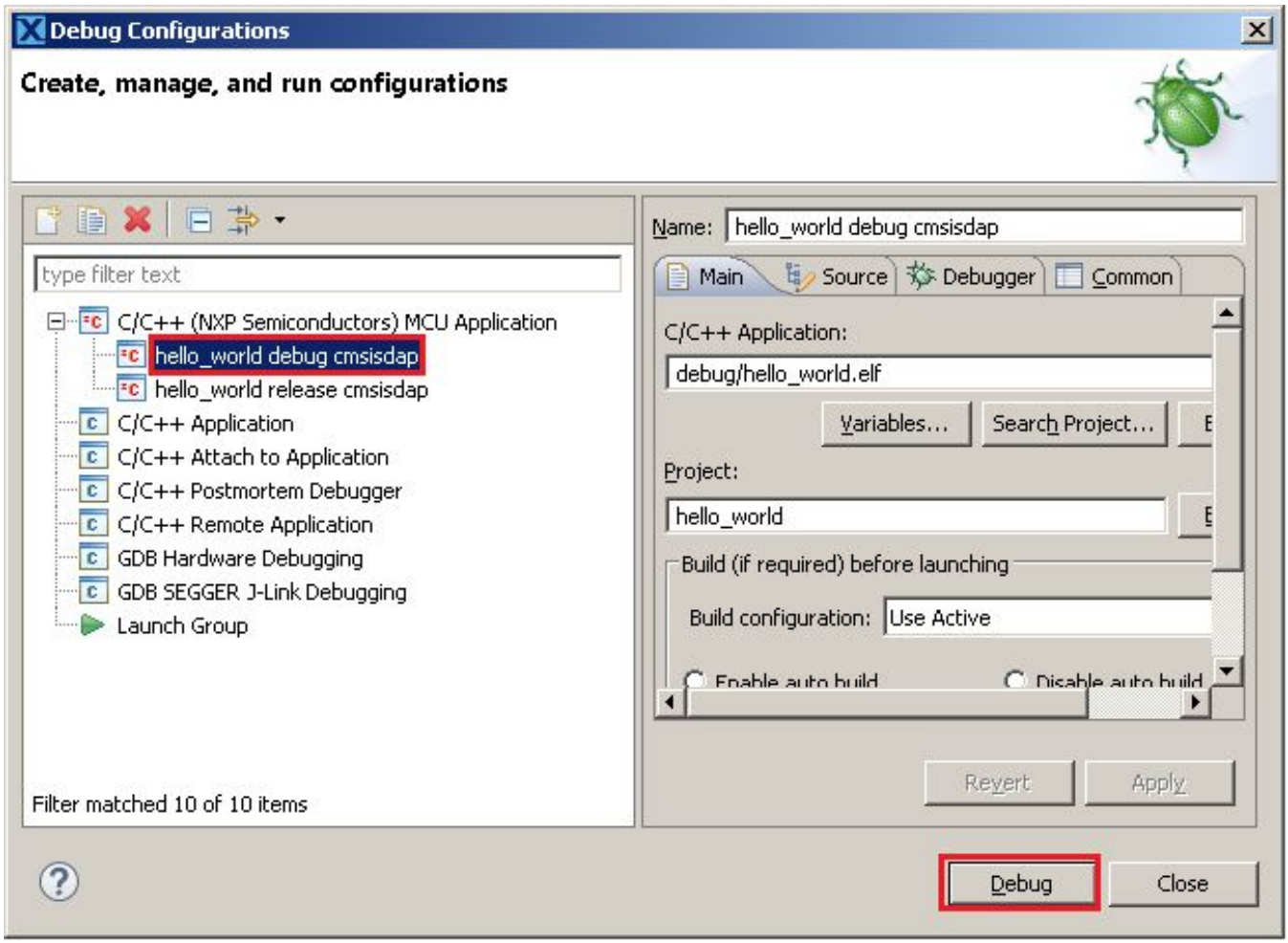

Figure 26. Selection of the debug configuration and debugger launch

4. Additional dialog windows may appear to select LPC-INK2 CMSIS-DAP emulator and core in case of multicore derivatives. Select it and click the "OK" button. The application is downloaded to the target and automatically run to main():

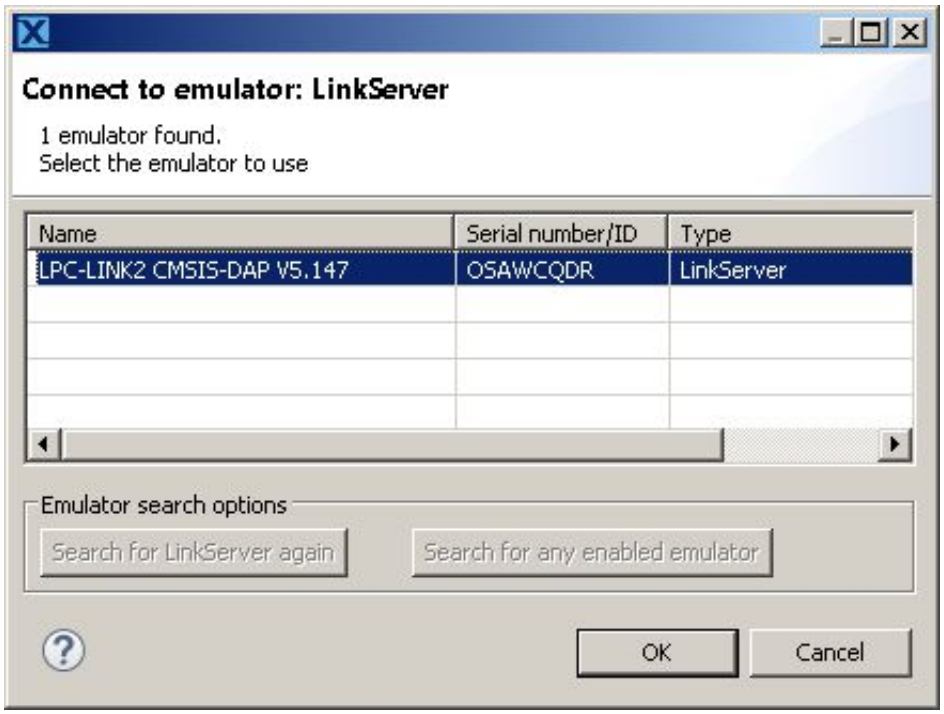

Figure 27. CMSIS-DAP emulator selection dialog

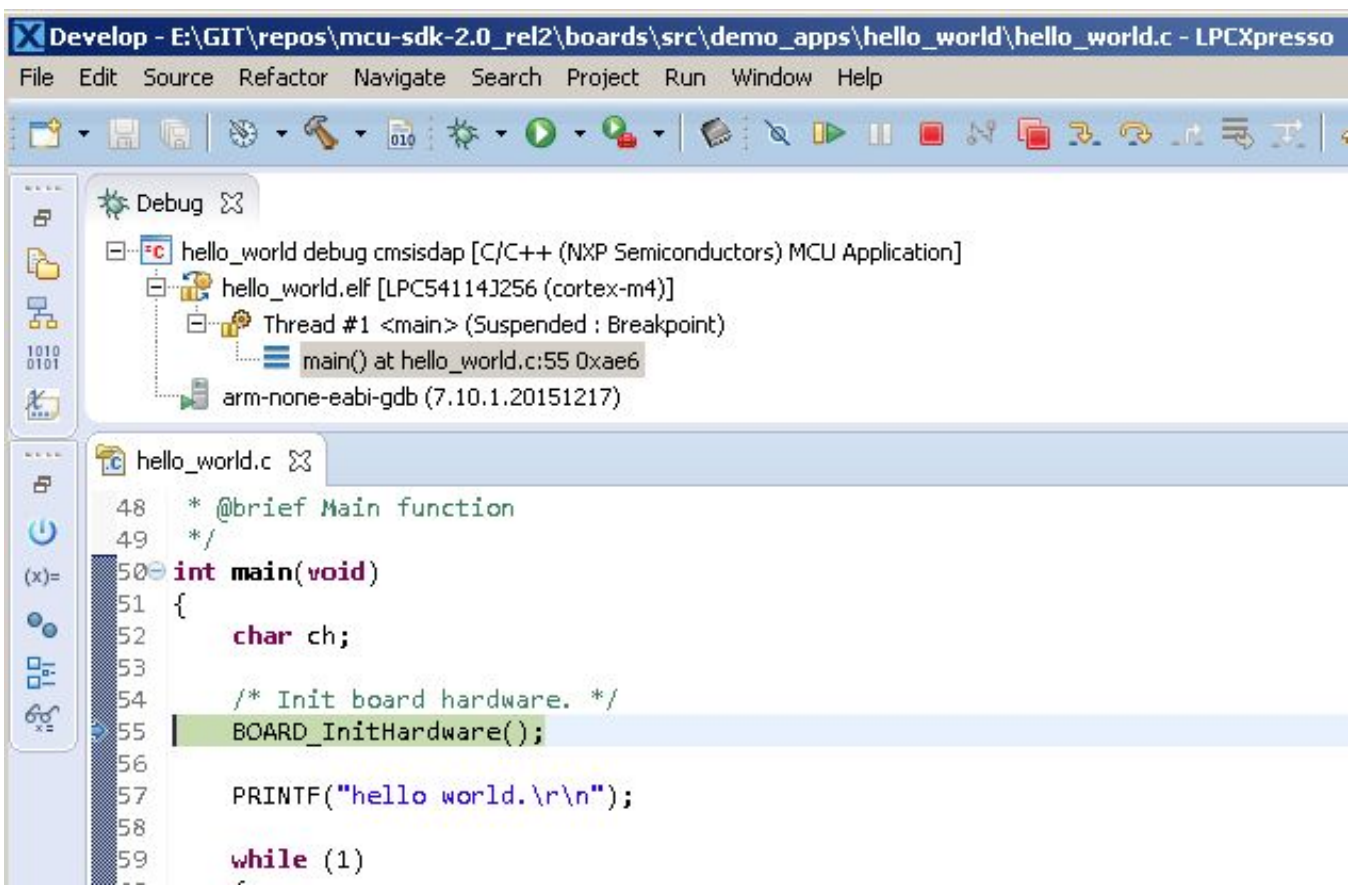

Figure 28. Stop at main() when running debugging

5. Start the application by clicking the "Resume" button:

Figure 29. Resume button

The hello_world application is now running and a banner is displayed on the terminal. If this is not true, check your terminal settings and connections.

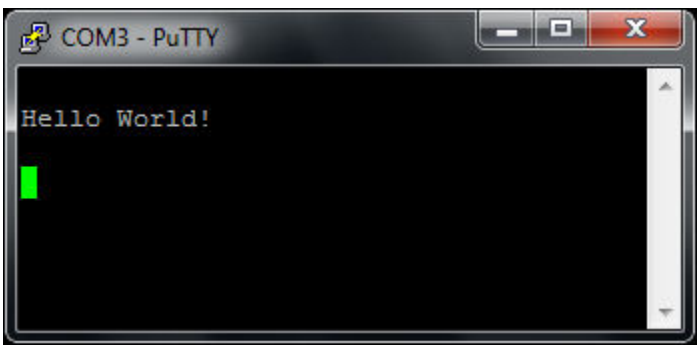

Figure 30. Text display of the hello_world demo

6 Appendix A - How to determine COM port

This section describes the steps necessary to determine the debug COM port number of your NXP hardware development platform. All NXP boards ship with a factory programmed, on-board debug interface LPC Link2.

1. To determine the COM port, open the Windows operating system Device Manager. This can be achieved by going to the Windows operating system Start menu and typing “Device Manager” in the search bar, as shown below:

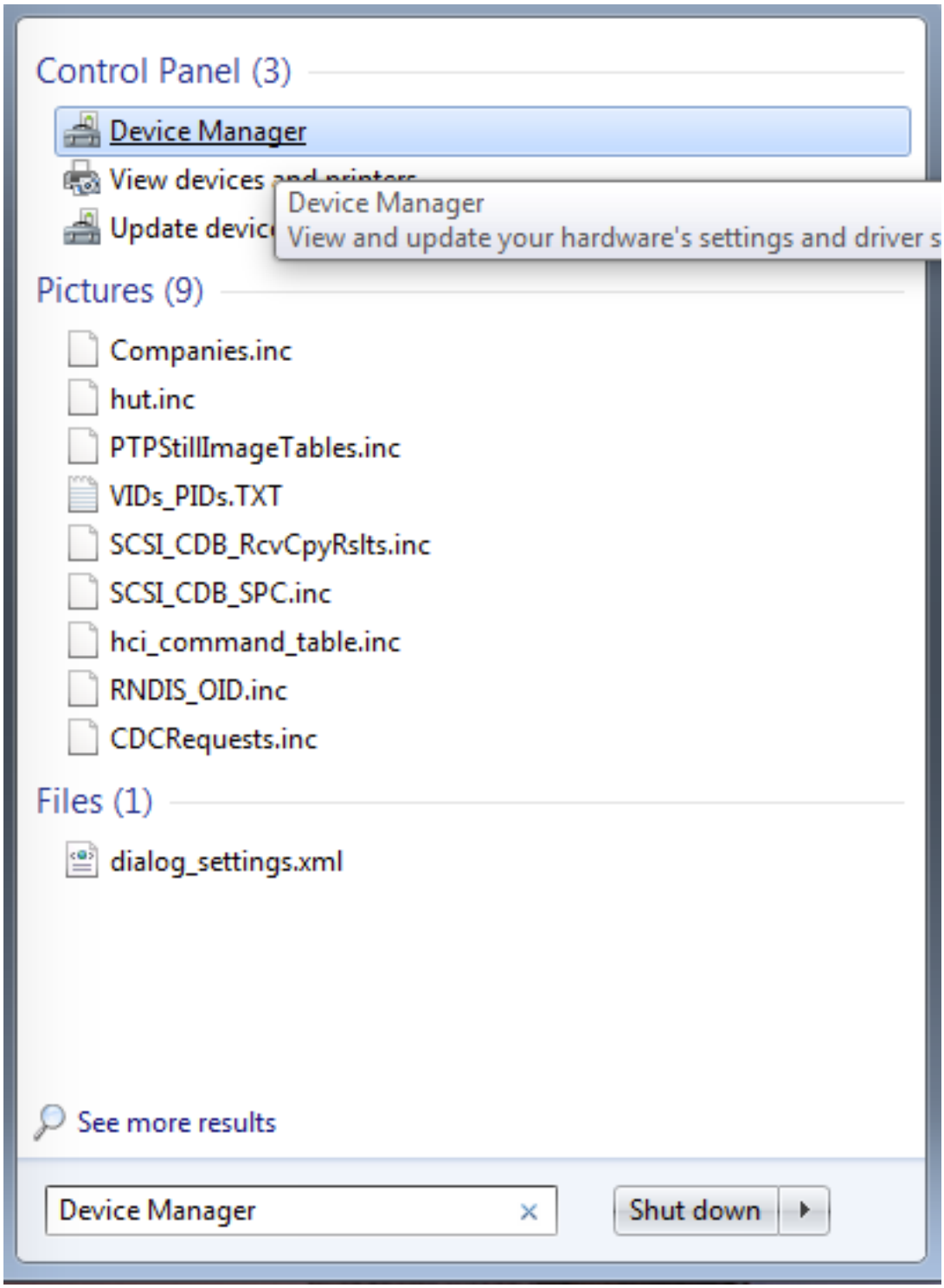

Figure 31. Device manager

2. In the Device Manager, expand the “Ports (COM & LPT)” section to view the available ports. Depending on the NXP board you’re using (see Appendix B), the COM port can be named differently:
   a. CMSIS-DAP interface:

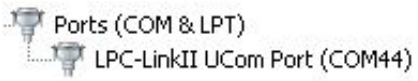

Figure 32. CMSIS-DAP interface

7 Appendix B - Updating Debugger firmware

The NXP LPCXpresso hardware platform comes with a CMSIS-DAP-compatible debug interface that has the ability to update the debugger firmware. This typically means switching from the default application (CMSIS-DAP) to a SEGGER J-Link. This section contains the steps to switch the CMSIS-DAP firmware to a J-Link interface. However, the steps can also be applied to restoring the original image.

NOTE

If LPCXpresso IDE is used and the jumper making DFUlink is installed on the board (JP5 on some boards, but consult the board user manual or schematic for specific jumper number), Link2 debug probe boots to DFU mode, and LPCXpresso IDE automatically downloads the CMSIS-DAP firmware to the probe before flash memory programming (after clicking the "Debug" button). Using DFU mode ensures most up-to-date/ compatible firmware is used with LPCXpresso IDE.

NXP provides LCPScrypt utility, which is the recommended tool for programming the latest versions of CMSIS-DAP and J-Link firmware onto LPC-Link2 or LPCXpresso boards. The utility can be downloaded from www.nxp.com/lpcutilities.

These steps show how to update the debugger firmware on your board for Windows operating system. For Linux OS, follow the instructions described in LPCScrypt user guide (www.nxp.com/lpcutilities, select LPCScrypt, then select documentation tab).

1. Install the LPCScript utility.
2. Unplug the board's USB cable.
3. For LPCXpresso board: make DFU link (install the jumper labelled DFUlink).
4. Connect the probe to the host via USB (use Link USB connector).
5. Open a command shell and call the appropriate script located in the LPCScrypt installation directory (<LPCScrypt install dir>).
   a. To program CMSIS-DAP debug firmware: `<LPCScrypt install dir>/scripts/program_CMSIS`
   b. To program J-Link debug firmware: `<LPCScrypt install dir>/scripts/program_JLINK`
6. Remove DFU link (remove the jumper installed in step 3).
7. Re-power the board by removing the USB cable and plugging it again.

8 Revision History

This table summarizes revisions to this document.

Table 1. Revision History

Revision number	Date	Substantive changes
0	06/2016	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: KSDK20LPC5411XGSUG
Rev. 0
06/2016