NXP Semiconductors
User's Guide

Document identifier: MCUXSDKK32W061GSUG
Rev. 0, December 10, 2020

Getting Started with MCUXpresso SDK for K32W061

Contents

Chapter 1
Overview

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

For supported toolchain versions, see *MCUXpresso SDK Release Notes for K32W061* (document MCUXSDKK32W061RN).

For more details about MCUXpresso SDK, refer to MCUXpresso-SDK: Software Development Kit for MCUXpresso.

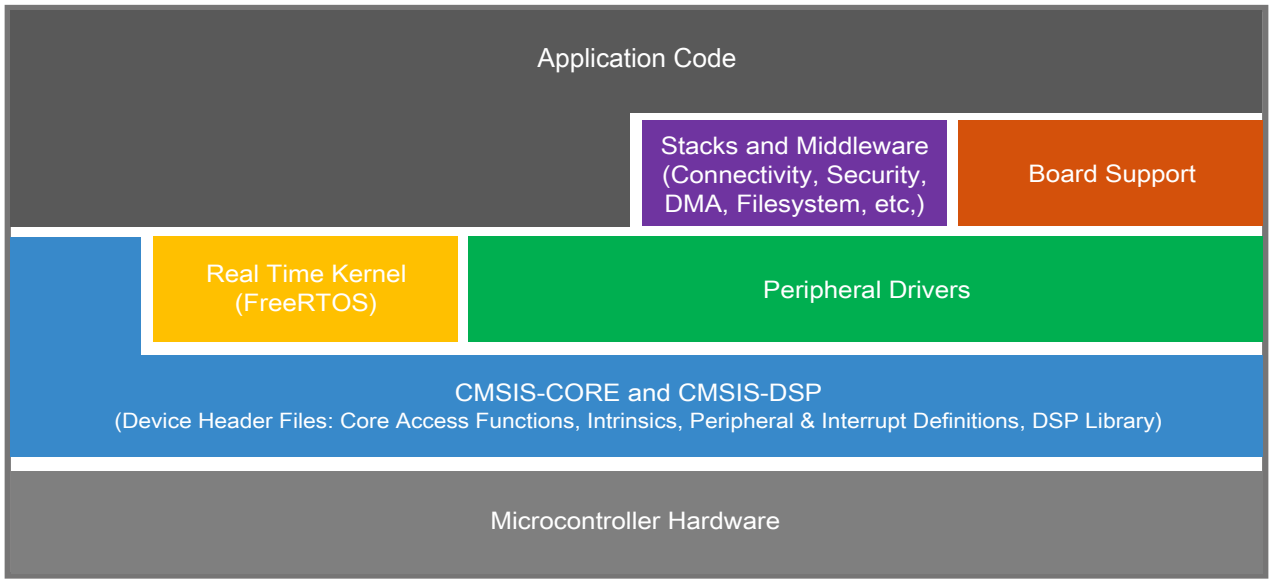

Figure 1. MCUXpresso SDK layers

Chapter 2
MCUXpresso SDK board support package folders

MCUXpresso SDK board support package provides example applications for NXP development and evaluation boards for Arm® Cortex®-M cores including Freedom, Tower System, and LPCXpresso boards. Board support packages are found inside the top level boards folder and each supported board has its own folder (an MCUXpresso SDK package can support multiple boards). Within each `<board_name>` folder, there are various sub-folders to classify the type of examples it contain. These include (but are not limited to):

- `demo_apps`: Full-featured applications that 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 that show how to use the MCUXpresso SDK's peripheral drivers for a single use case. These applications typically only use a single peripheral but there are cases where multiple peripherals are used (for example, SPI conversion using DMA).
- `rtos_examples`: Basic FreeRTOS™ OS examples that show the use of various RTOS objects (semaphores, queues, and so on) and interfaces with the MCUXpresso SDK's RTOS drivers
- `wireless_examples`: Applications that use the bluetooth stack, Zigbee and OpenThread stacks.

2.1 Example application structure

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

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

In the `hello_world` application folder you see the following contents:

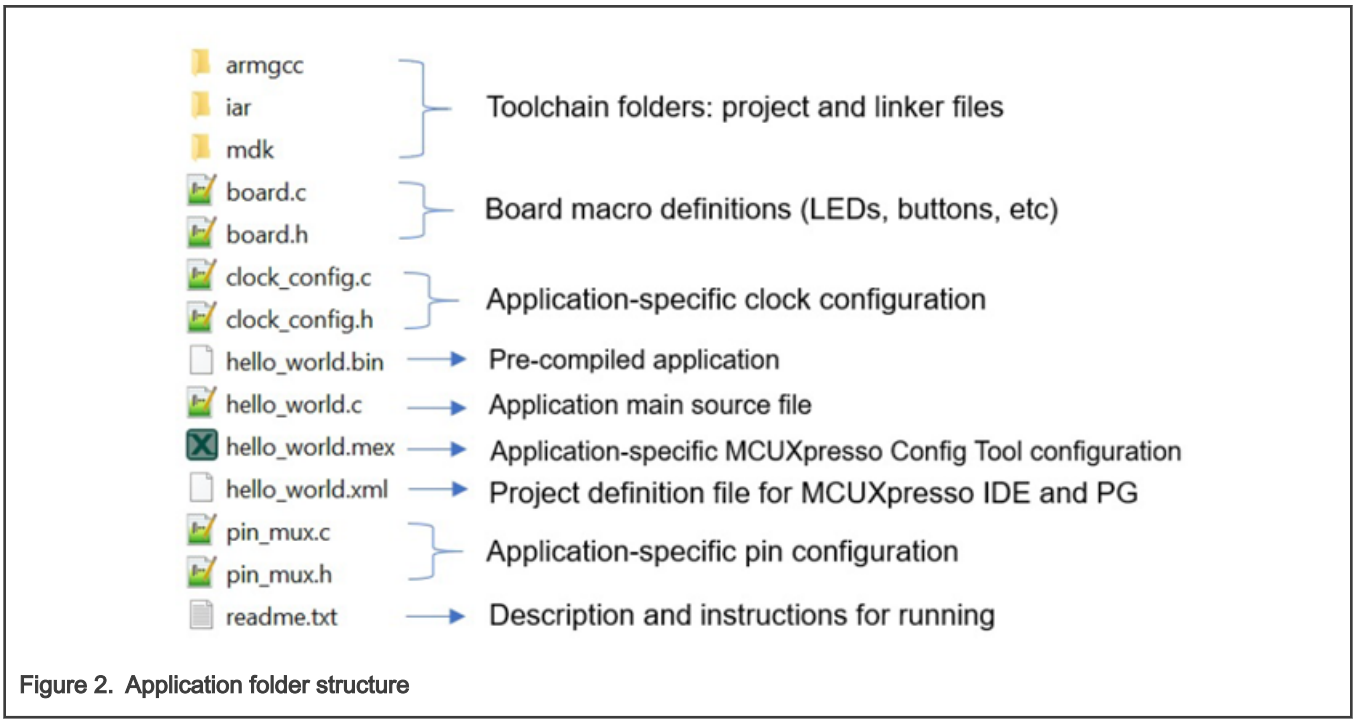

Figure 2. Application folder structure

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

NOTE

To prevent compiling errors, do not use special characters in the path of the SDK such as {!,@,#,$,&,%,^} and space.

2.2 Locating example application source files

When opening an example application in any of the supported IDEs, a variety of source files are referenced. The MCUXpresso SDK devices folder is the central component to all example applications. It means 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 MCUXpresso SDK tree used in all example applications are:

- `devices/<device_name>`: The device's CMSIS header file, MCUXpresso SDK feature file and a few other files
- `devices/<device_name>/drivers`: All of the peripheral drivers for your specific MCU
- `devices/<device_name>/<tool_name>`: Toolchain-specific startup code, including vector table definitions
- `devices/<device_name>/utilities`: Items such as the debug console that are used by many of the example applications
- `devices/<devices_name>/project_template`: Project template used by MCUXpresso IDE to create new projects
- `devices/<devices_name>/image` tool: Post build scripts used by IAR and MCUXpresso IDE

For examples containing an RTOS, there are references to the appropriate source code. RTOSes are in the `rtos` folder. The core files of each of these are shared, so modifying one could have potential impacts on other projects that depend on that file.

Chapter 3
Set up toolchain

This section contains the steps to install the necessary components required to build and run an MCUXpresso SDK demo application with the Arm GCC toolchain, as supported by the MCUXpresso SDK. There are many ways to use Arm GCC tools, but this example focuses on a Windows operating system environment.

3.1 Install GCC Arm Embedded tool chain

Download and run the installer from developer.arm.com/open-source/gnu-toolchain/gnu-rm. This is the actual toolset (in other words, compiler, linker, and so on). The GCC toolchain should correspond to the latest supported version .

3.2 Add the new system environment variables

Create a new `system` environment variable and name it as `ARMGCC_DIR`. The value of this variable should point to the Arm GCC Embedded tool chain installation path. For this example, the path is:

```
C:\GNU Tools Arm Embedded\7 2018-q2-update
```

See the installation folder of the GNU Arm GCC Embedded tools for the exact path name of your installation.

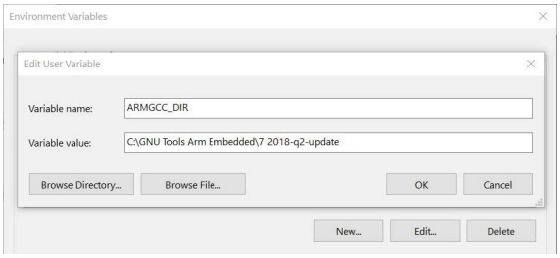

Figure 3. Add `ARMGCC_DIR` system variable

Add the `bin` directory path of the GNU Arm GCC Embedded tools in **System variables** -> **Path**. For this example, the path is:

```
C:\GNU Tools Arm Embedded\7 2018-q2-update\bin
```

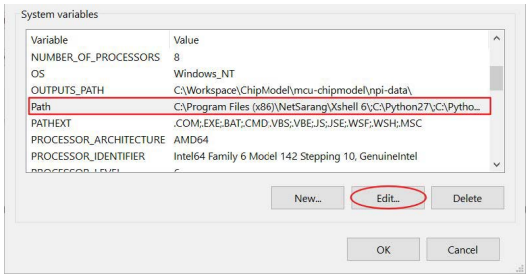

Figure 4. Select Path and click Edit

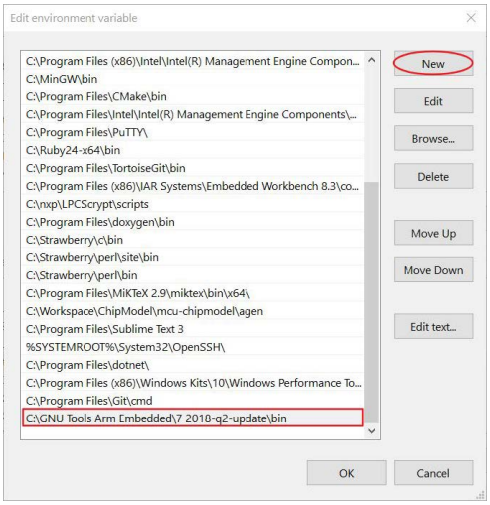

Figure 5. Click New and add the path

3.3 Install Python3

Download and run the **Python3** installer from https://www.python.org/downloads/.

Remember to select the **Add Python 3.7 to PATH** checkbox.

NOTE

Only the version newer than 3.2 is supported while generating the JN518x binaries.

Download and run the **VCForPython** installer from https://download.microsoft.com.

After the installation, please confirm that the installation path is added into **System variables Path**. If not, please do it manually.

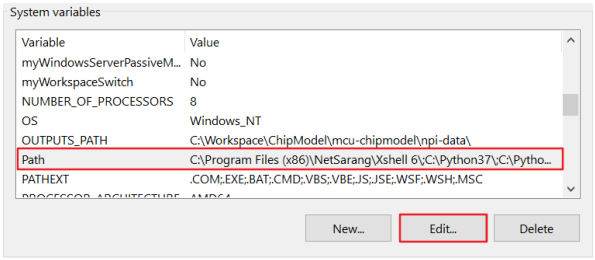

Figure 6. Select Path and click Edit

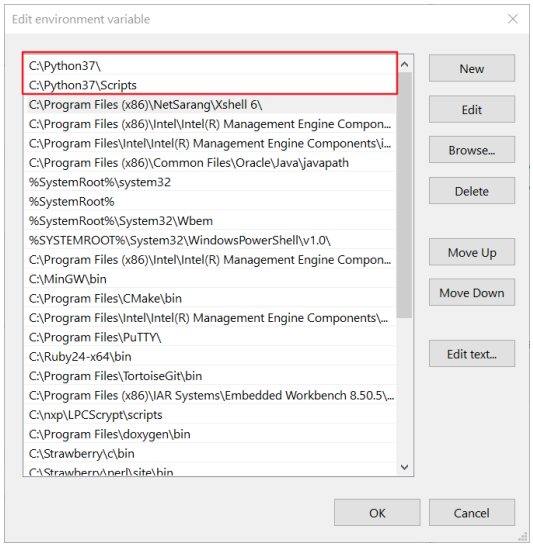

Figure 7. Confirm the path is added

Follow the below steps to install the Crypto library for Python 3.

1. Press Windows+R to open the **Run** box.
2. Type **cmd** and then click **OK** to open a regular Command Prompt.
3. Type and run the command: `pip3 install pycryptodome`.

Chapter 4
Run a demo application using IAR

This section describes the steps required to build, run, and debug example applications provided in the MCUXpresso SDK. The `hello_world` demo application targeted for the k32w061dk6 hardware platform is used as an example, although these steps can be applied to any example application in the MCUXpresso SDK.

4.1 Build an example application

Do the following steps to build the `hello_world` example application.

1. 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 K32W061DK6 hardware platform as an example, the `hello_world` workspace is located in:

   ```
   <install_dir>/boards/k32w061dk6/demo_apps/hello_world/iar/hello_world.eww
   ```

   Other example applications may have additional folders in their path.

2. Select the desired build target from the drop-down menu.

   For this example, select **hello_world** – **debug**.

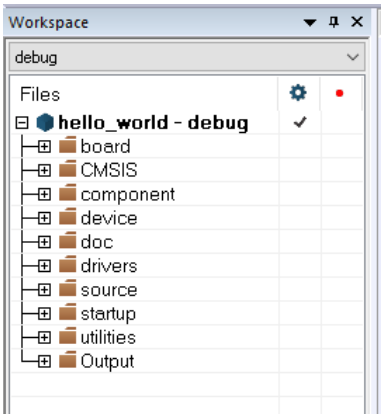

Figure 8. Demo build target selection

3. To build the demo application, click **Make**, highlighted in red in Figure 9.

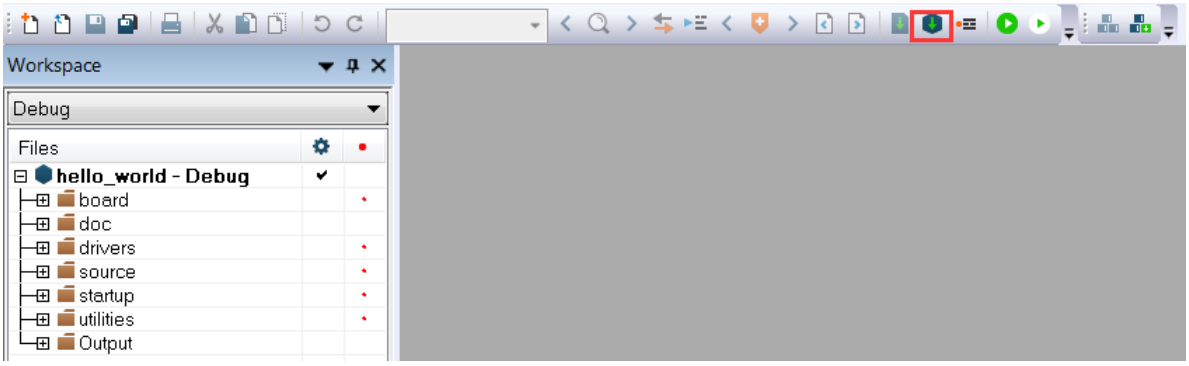

Figure 9. Build the demo application

4. The build completes without errors.

4.2 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 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 (J2) and the PC USB connector. Ensure JP5 is removed so the Link2 boots from internal flash. 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 How to determine COM port). Configure the terminal with these settings:
    a. 115200 or 9600 baud rate, depending on your settings (reference the `BOARD_DEBUG_UART_BAUDRATE` variable in the `board.h` file)
    b. No parity
    c. 8 data bits
    d. 1 stop bit

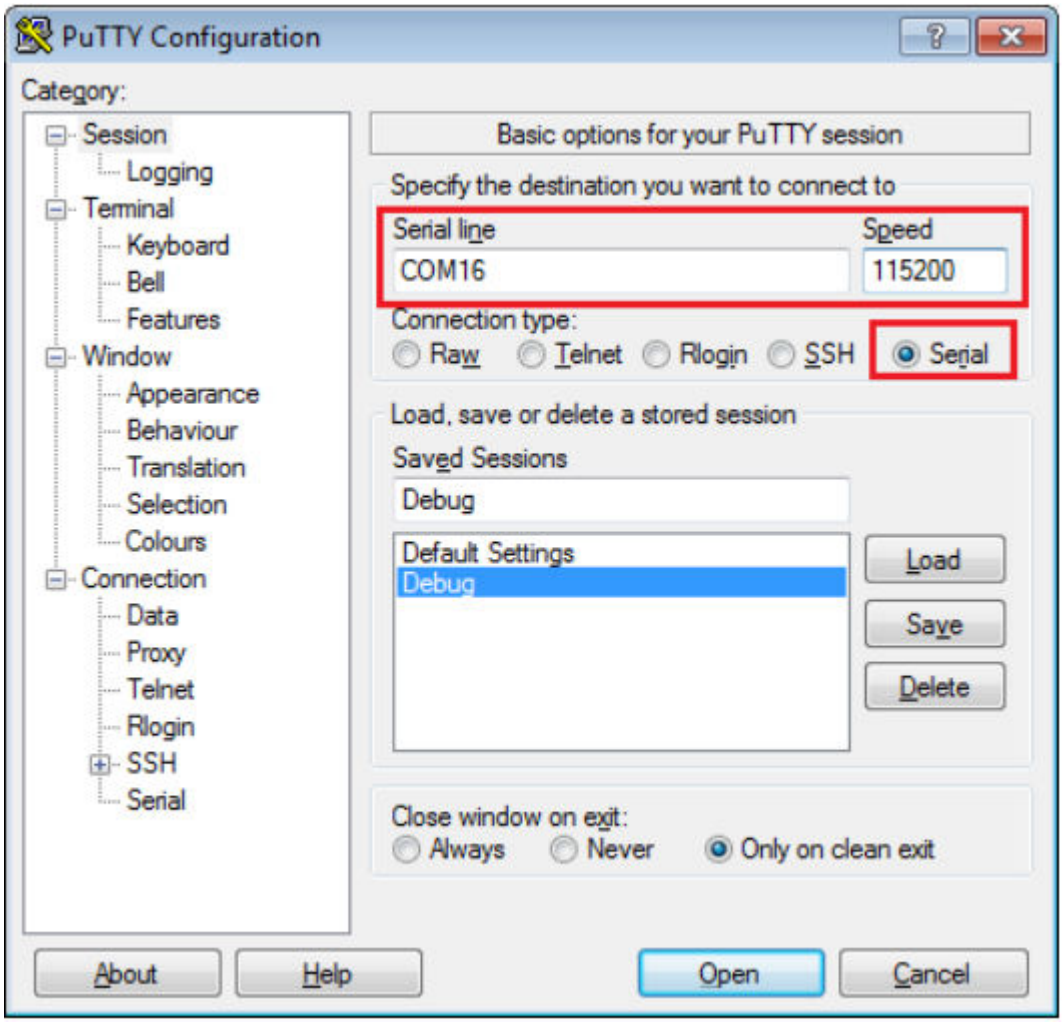

Figure 10. Terminal (PuTTY) configuration

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

Figure 11. Download and Debug button

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

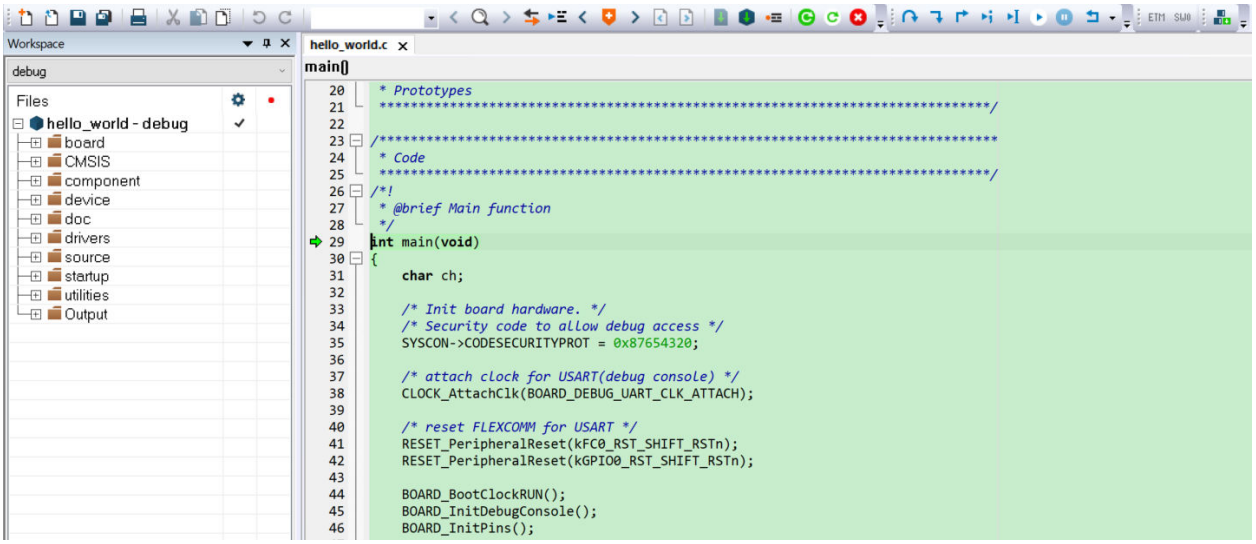

Figure 12. Stop at main() when running debugging

6. Run the code by clicking **Go** to start the application.

Figure 13. 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.

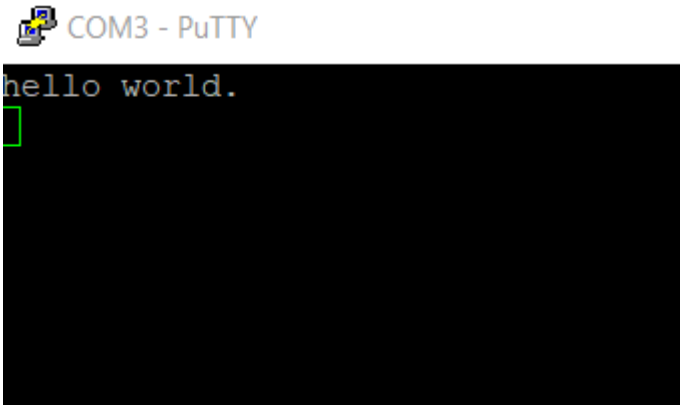

Figure 14. Text display of the hello_world demo

Chapter 5
Run a demo using MCUXpresso IDE

NOTE

Ensure that the MCUXpresso IDE toolchain is included when generating the MCUXpresso SDK package.

This section describes the steps required to configure MCUXpresso IDE to build, run, and debug example applications. The `hello_world` demo application targeted for the K32W061DK6 hardware platform is used as an example, though these steps can be applied to any example application in the MCUXpresso SDK.

5.1 Select the workspace location

Every time MCUXpresso IDE launches, it prompts the user to select a workspace location. MCUXpresso IDE is built on top of Eclipse which uses workspace to store information about its current configuration, and in some use cases, source files for the projects are in the workspace. The location of the workspace can be anywhere, but it is recommended that the workspace be located outside of the MCUXpresso SDK tree.

5.2 Build an example application

To build an example application, follow these steps.

1. Drag and drop the SDK zip file into the **Installed SDKs** view to install an SDK. In the window that appears, click **OK** and wait until the import has finished.

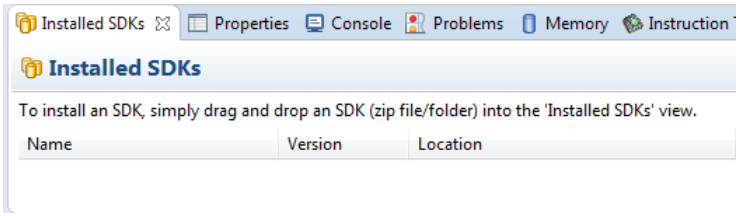

Figure 15. Install an SDK

2. On the **Quickstart Panel**, click **Import SDK example(s)...**.

Figure 16. Import an SDK example

3. In the window that appears, expand the **K32W061** folder and select **K32W061** . Then, select **k32w061dk6** and click **Next**.

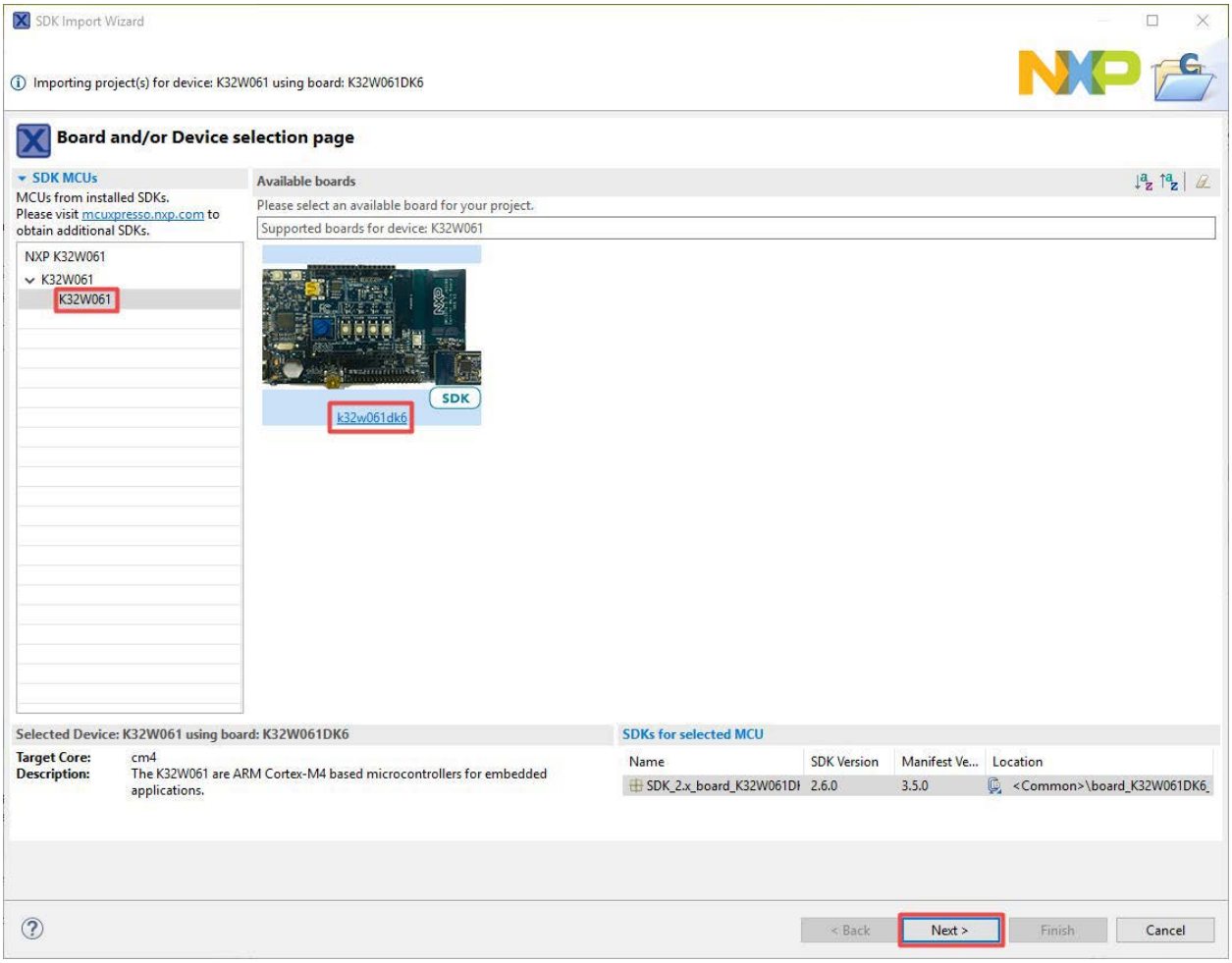

Figure 17. Select K32W061DK6 board

4. Expand the `demo_apps` folder and select `hello_world` . Then, click **Next** . If the **UART** option is enabled by default, click **Finish** to finish importing. There is no need to click **Next** and no need to go with next step.

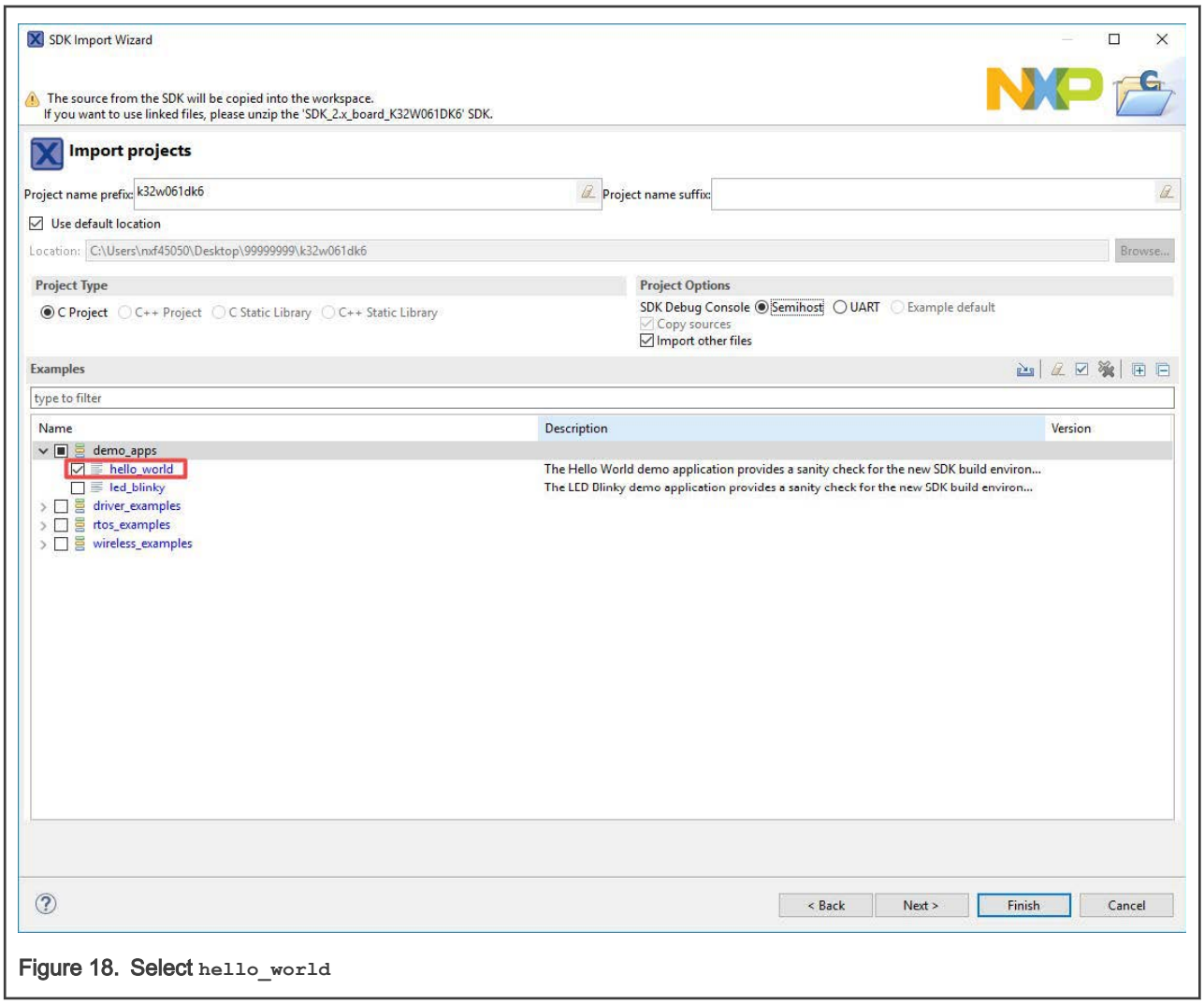

Figure 18. Select `hello_world`

5. Ensure **Redlib: Use floating point version of printf** is selected if the example prints floating point numbers on the terminal for demo applications such as `adc_basic`, `adc_burst`, `adc_dma`, and `adc_interrupt`. Otherwise, it is not necessary to select this option. Then, click **Finish**.

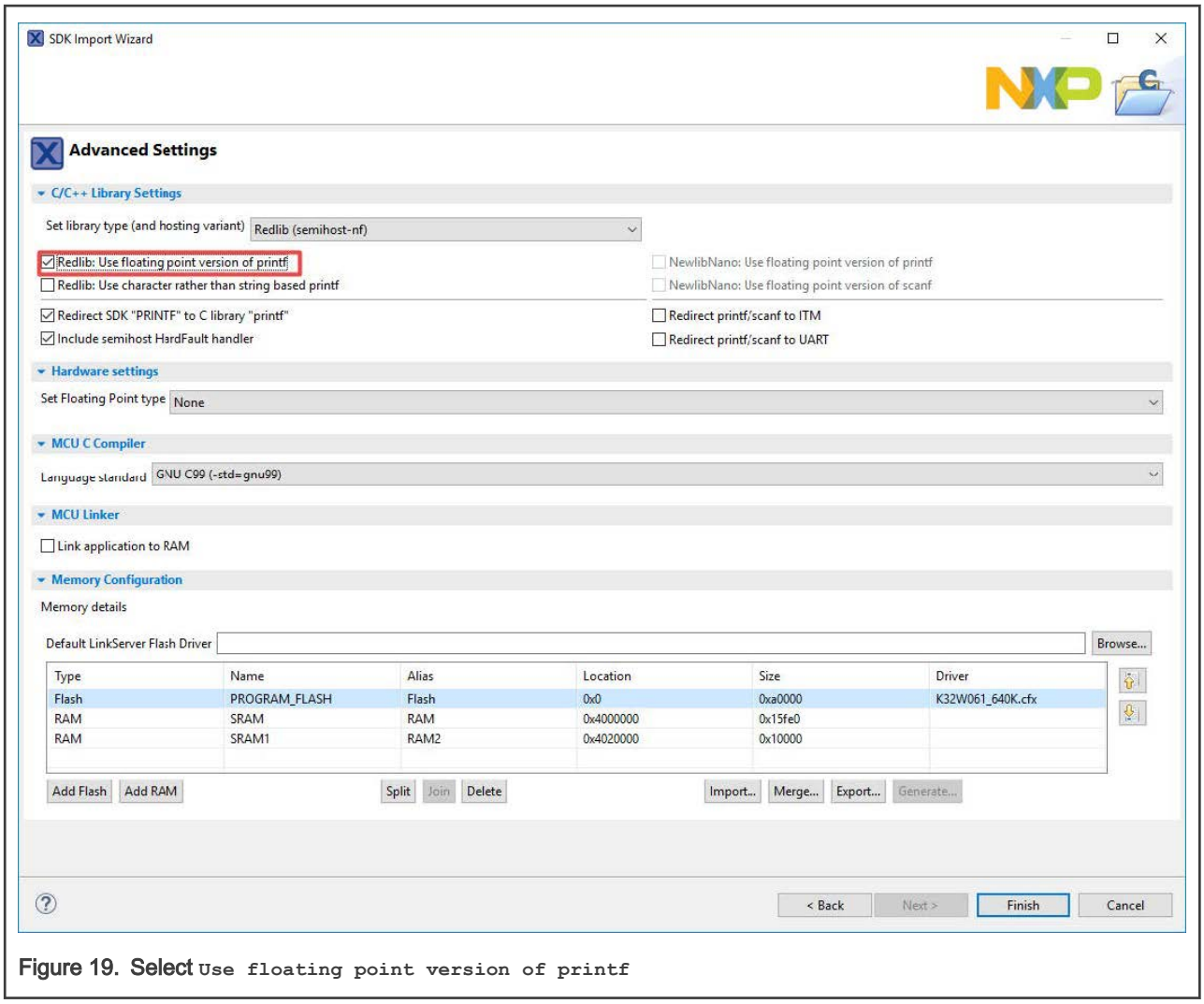

Figure 19. Select `Use floating point version of printf`

5.3 Run an example application

The application can be downloaded into DK6 K32W061 module either:

- via the LPC-LINK2 USB port using an IDE debugger (MCUXpresso or IAR)
- via UART0 using the DK6 Flash Programmer

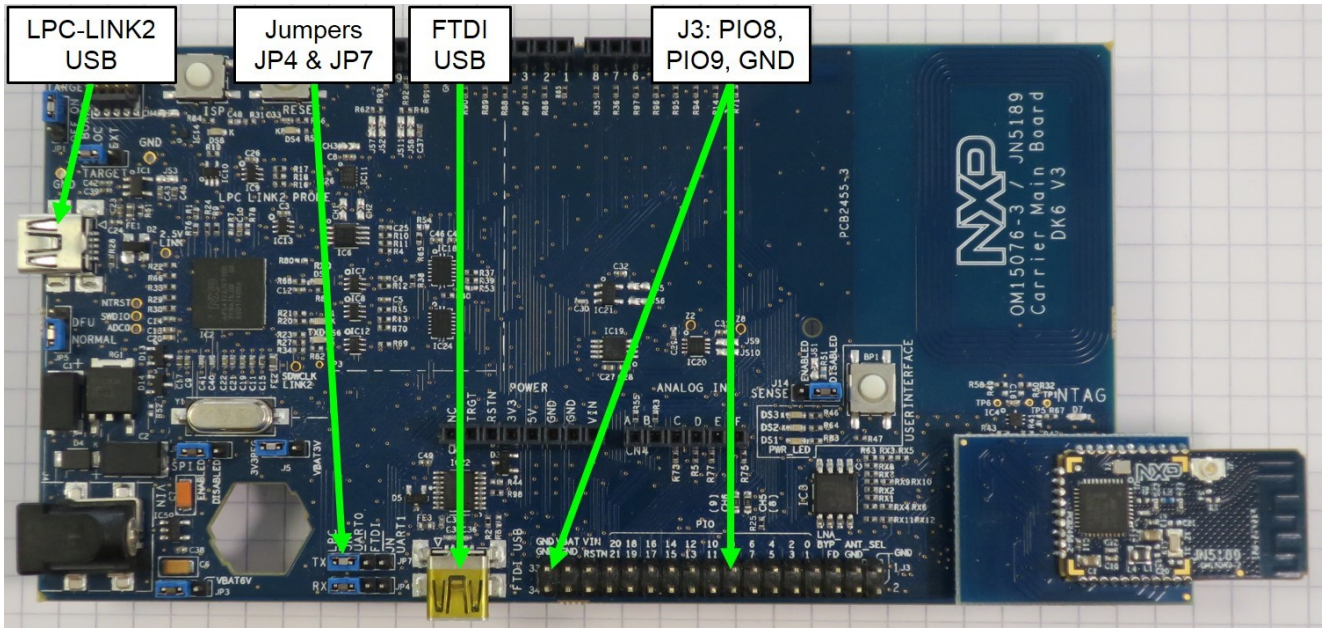

Figure 20. Annotated DK6

UART0 can be routed to several ports of the DK6:

- the LPC-LINK2 USB port (JP4 and JP7 in the leftmost position as shown in Figure 20)
- the FTDI USB connector (JP4 and JP7 in the middle position)
- J3 connector pins 8 and 9 using a USB-to-Serial converter (e.g. Prolific model obtainable from nxp.com)

UART0 is user port and UART1 instance is a debug console. The debug console can be accessed via J3 pins 10 and 11 using a USB-to-Serial converter. The easiest way to utilize BLE applications on the DK6 board is to plug a mini-USB cable into LPC-LINK2 USB connector. This acts as a power source and UART interface that can be used to download the binaries. Refer to the *DK6-UG-3127-Production-Flash-Programmer.pdf* for additional information.

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 or LPC-LINK2 or K32W061 boards) and the PC USB connector. If connecting for the first time, allow about 30 seconds for the devices to enumerate.
2. 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 How to determine COM port). Configure the terminal with these settings:
    a. 115200 or 9600 baud rate, depending on your board (reference `BOARD_DEBUG_UART_BAUDRATE` variable in the `board.h` file)
    b. No parity
    c. 8 data bits
    d. 1 stop bit

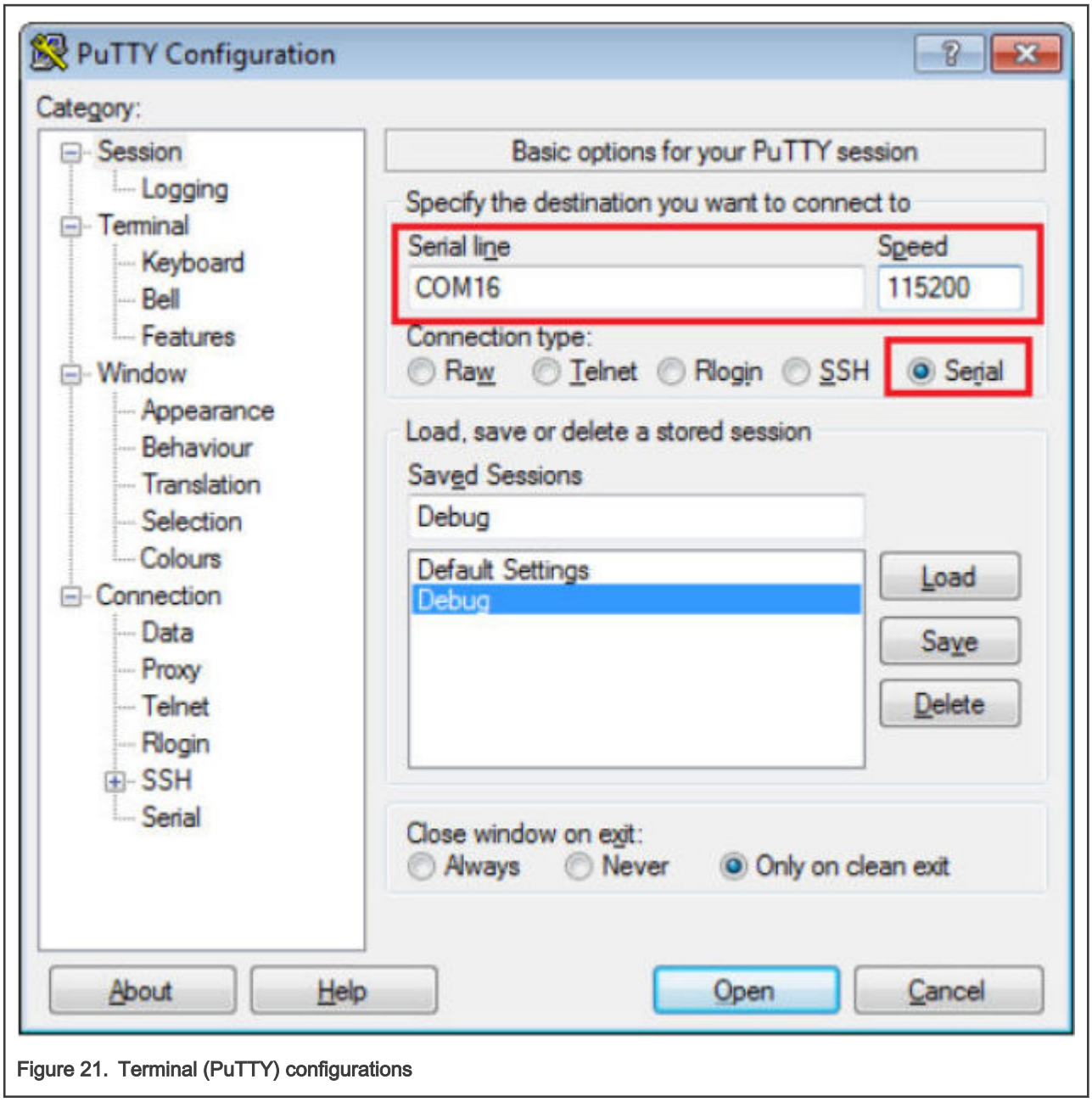

Figure 21. Terminal (PuTTY) configurations

3. On the **Quickstart Panel**, click **Debug**.

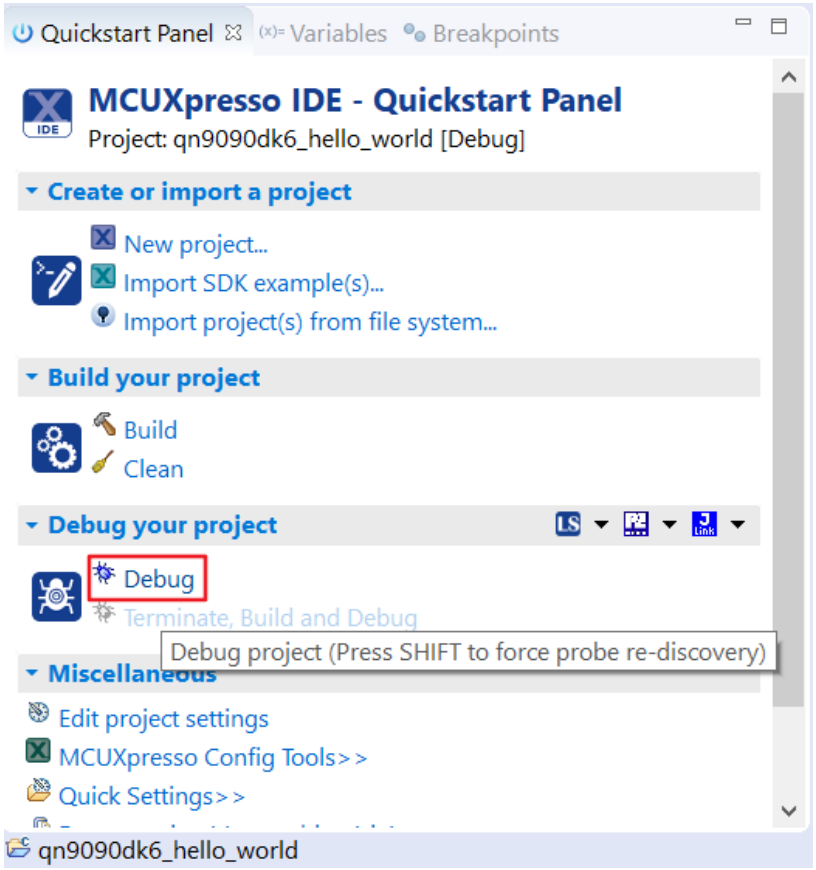

Figure 22. Debug `hello_world` case

4. The first time you debug a project, the **Debug Emulator Selection** dialog is displayed, showing all supported probes that are attached to your computer. Select the probe through which you want to debug and click **OK**. (For any future debug sessions, the stored probe selection is automatically used, unless the probe cannot be found.)

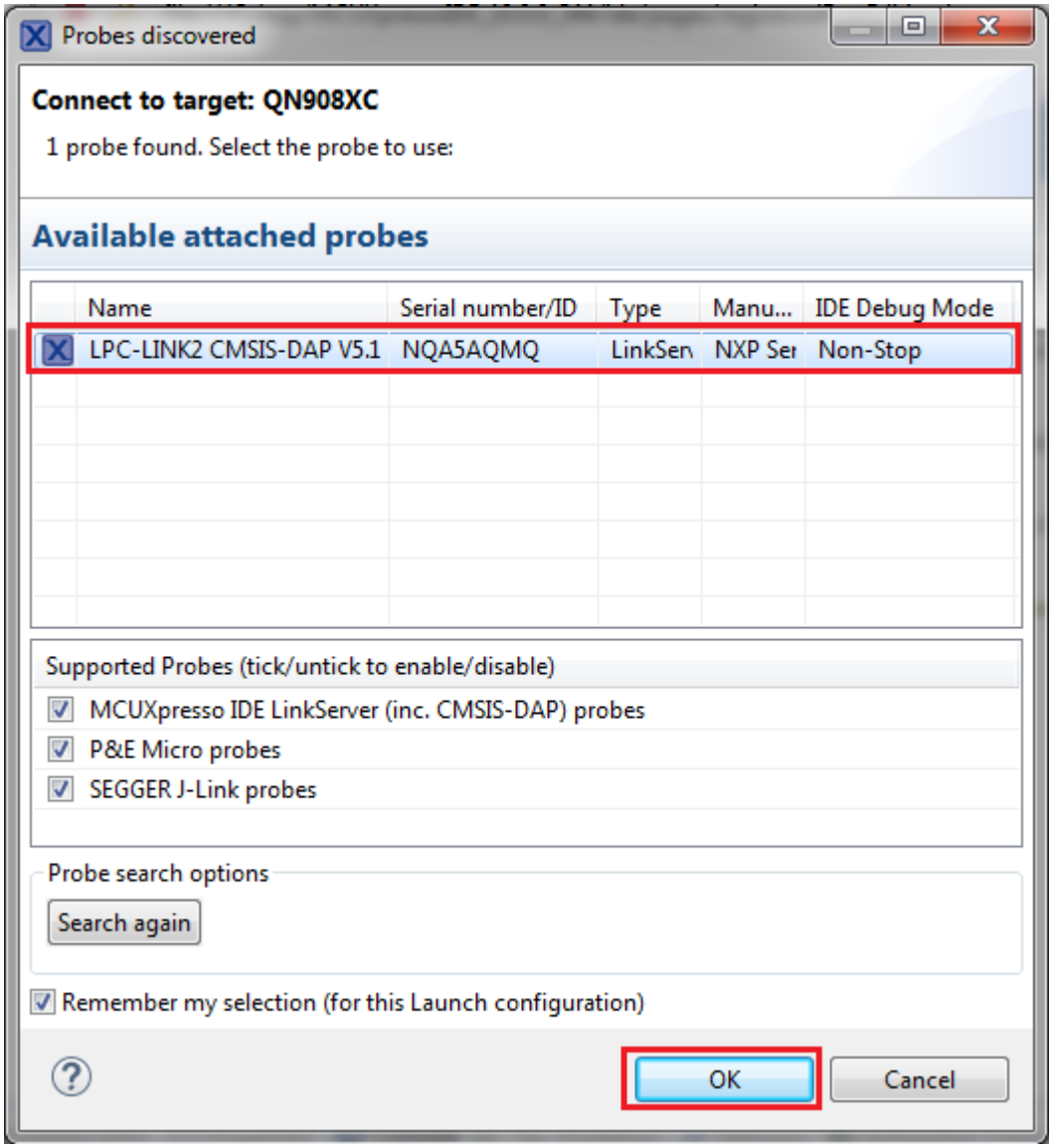

Figure 23. Attached Probes: debug emulator selection

5. The application is downloaded to the target and automatically runs to `main()`:

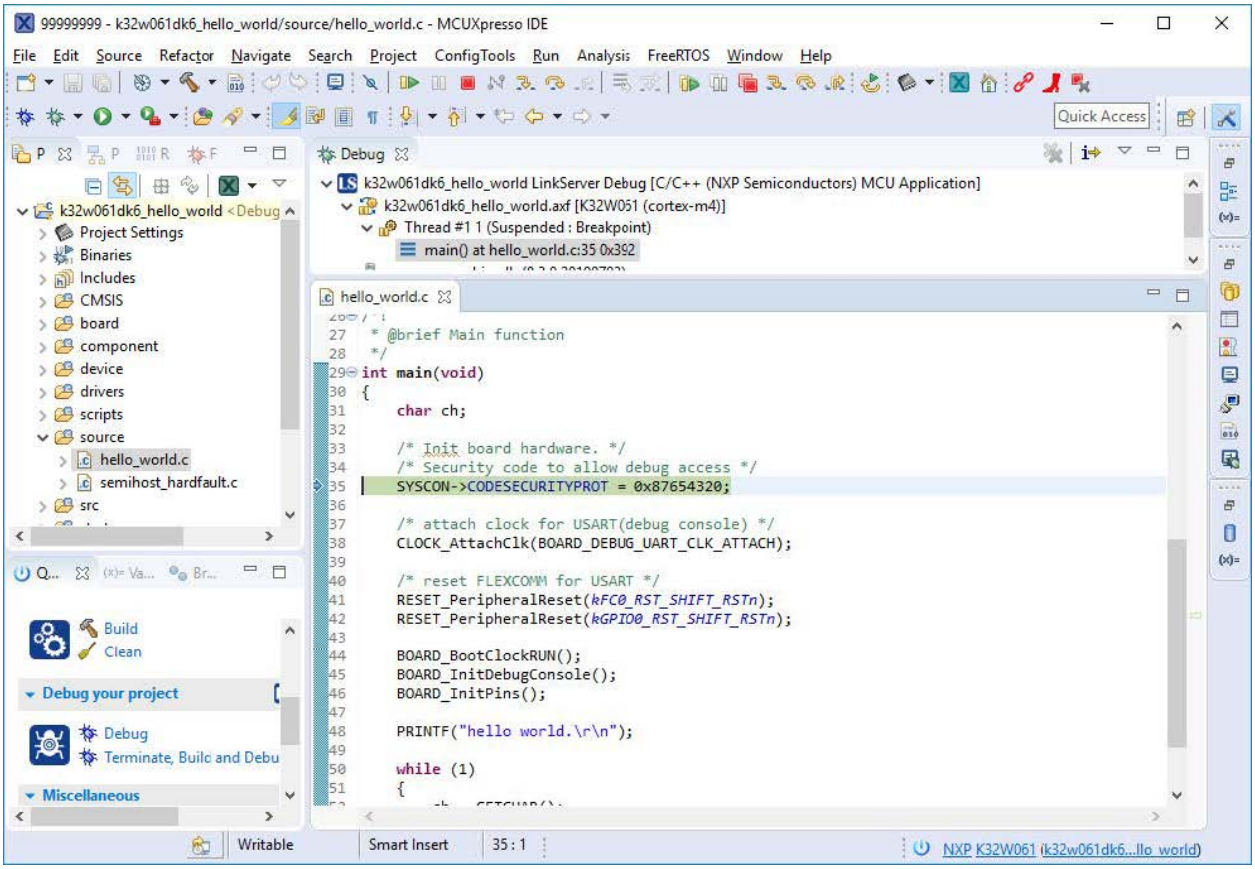

Figure 24. Stop at main() when running debugging

6. Start the application by clicking **Resume**.

Figure 25. Resume button

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

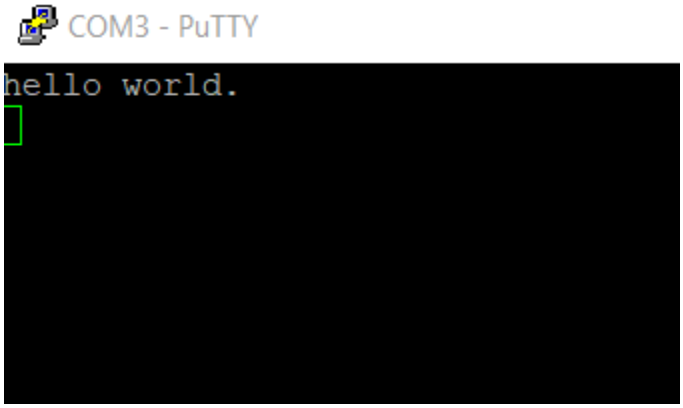

Figure 26. Text display of the `hello_world` demo

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 in Figure 27.

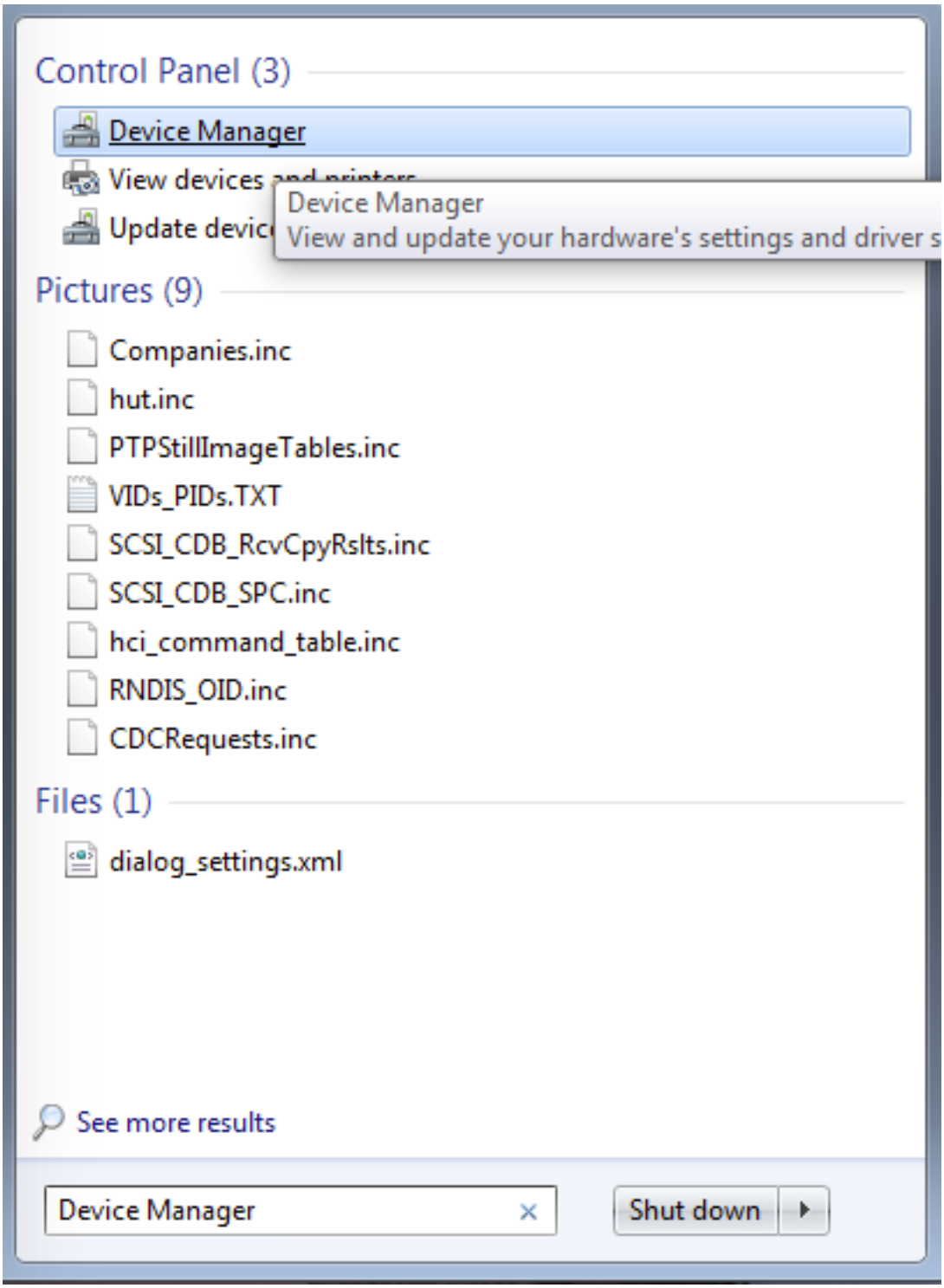

Figure 27. Device manager

2. In the Device Manager, expand the **Ports (COM & LPT)** section to view the available ports.

Appendix B
Updating Debugger firmware

The 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.

NXP provides the 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. Unplug the board's USB cable.
2. Install the LPCScript utility.
3. For LPCXpresso board: make DFU link (install the jumper labelled `DFUlink (JP5)`).
4. Connect the probe to the host via USB (use Link USB connector (J2)).
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.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: December 10, 2020

Document identifier: MCUXSDKK32W061GSUG

arm