NXP Semiconductors
User's Guide

Document Number: MCUXSDKIMX6ULGSUG
Rev. 0, 03/2017

Getting Started with MCUXpresso SDK for i.MX 6UltraLite Derivatives

Contents

1 Overview

The MCUXpresso Software Development Kit (SDK) provides comprehensive software support for 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 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 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 *MCUXpresso SDK Release Notes Supporting i.MX6 UltraLite Derivatives* (document MCUXSDKIMX6ULRN)

For the latest version of this and other MCUXpresso SDK documents, see the MCUXpresso SDK homepage MCUXpresso-SDK: Software Development Kit for MCUXpresso.

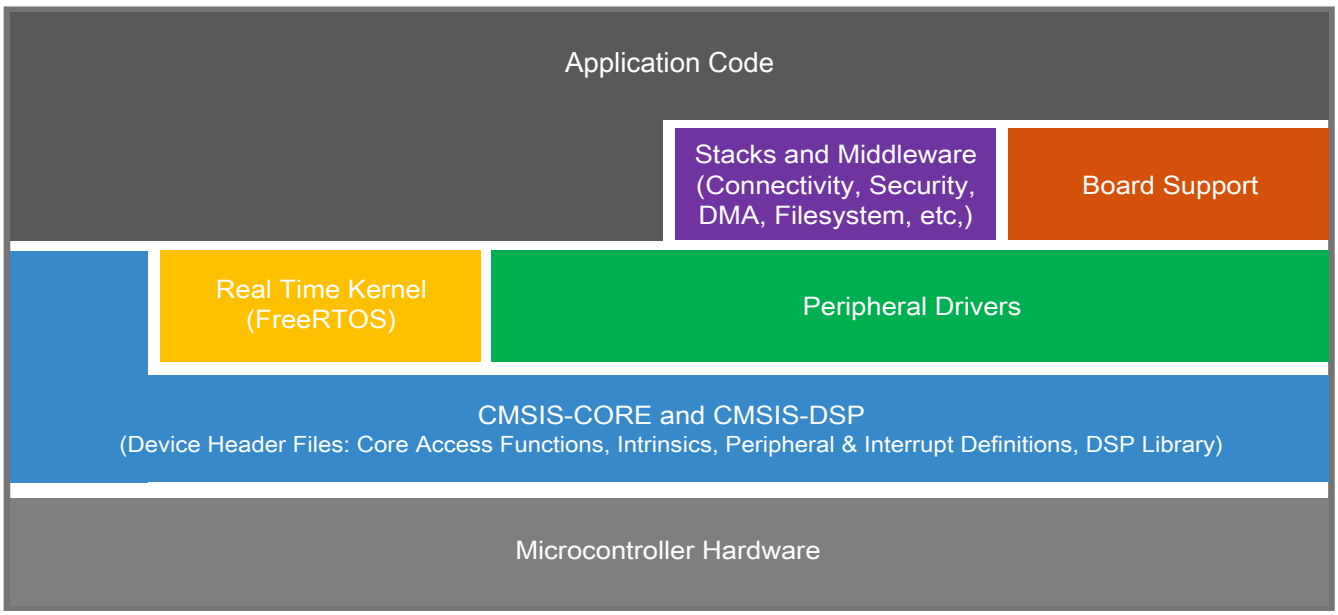

Figure 1. MCUXpresso SDK layers

2 MCUXpresso SDK Board Support Folders

MCUXpressoSDK board support provides example applications for NXP development and evaluation boards. Board support packages are found inside of 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 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 MCUXpresso 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™ OS examples showcasing the use of various RTOS objects (semaphores, queues, and so on) and interfacing with the MCUXpresso 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 MCUXpresso SDK. To get a comprehensive understanding of all MCUXpresso SDK components and folder structure, see the *MCUXpresso SDK API Reference Manual* document (MCUXSDKAPIRM).

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:

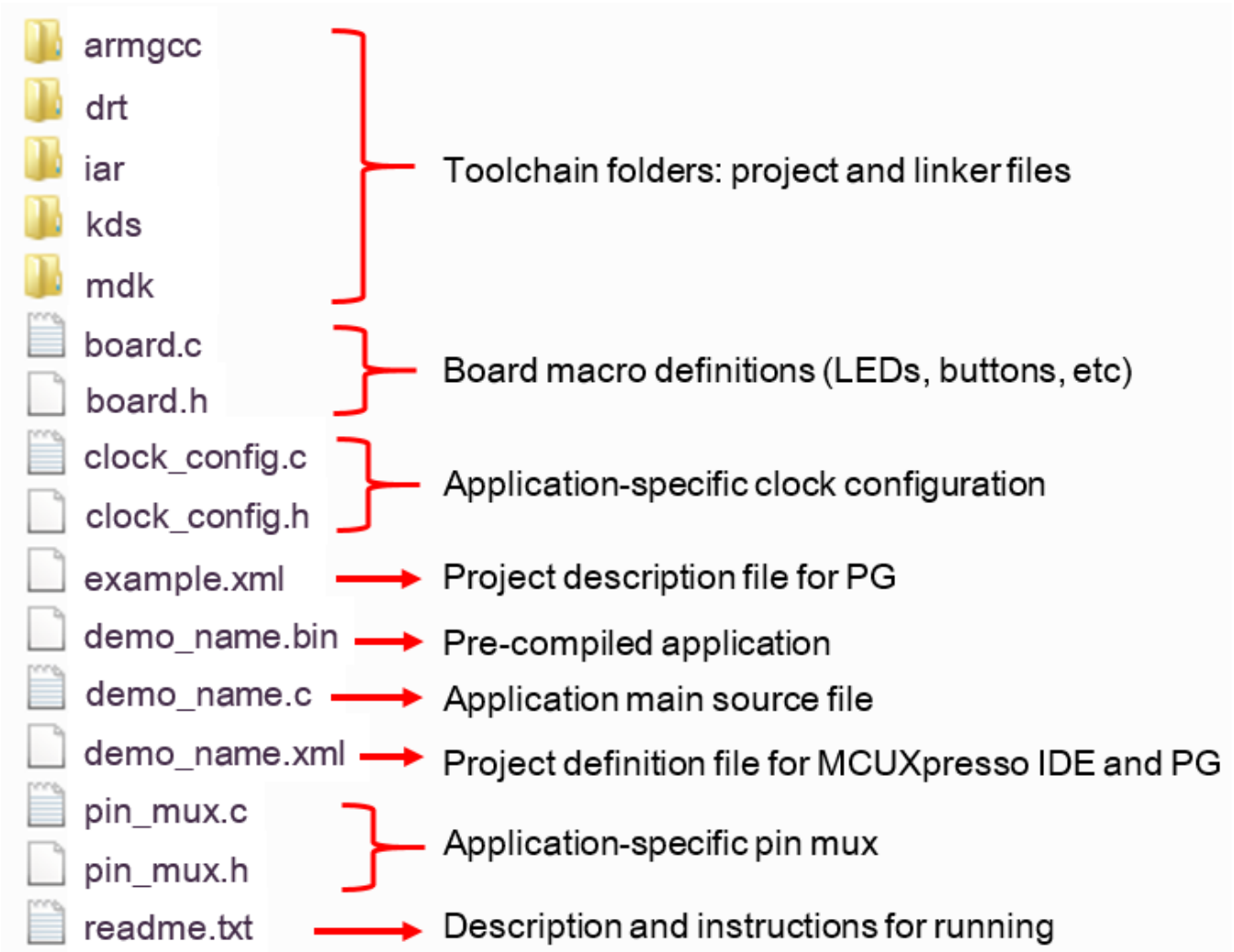

Figure 2. Application folder structure

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 MCUXpresso 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 MCUXpresso 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 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 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 or a RTOS, there are 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 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 MCIMX6UL-EVK hardware platform is used as an example, although these steps can be applied to any example application in the MCUXpresso SDK.

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 MCIMX6UL-EVK hardware platform as an example, the hello_world workspace is located in

   <install_dir>/boards/evkmcimx6ul/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.

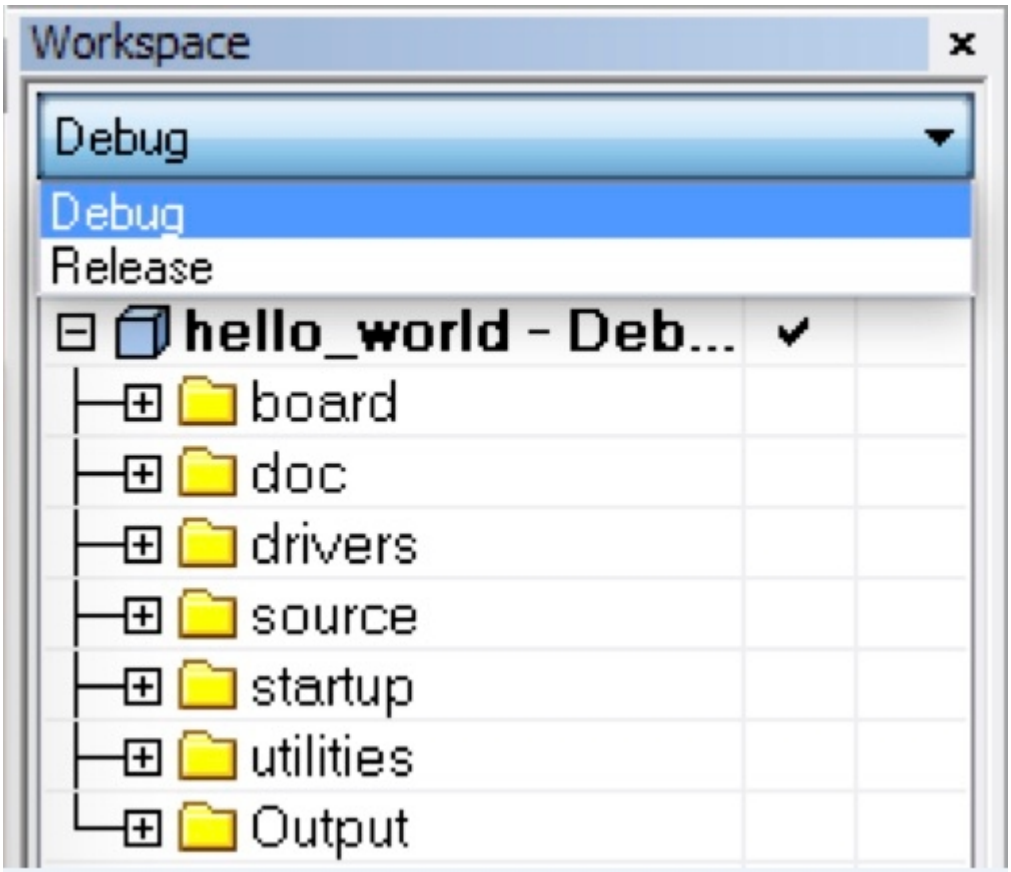

Figure 3. Demo build target selection

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

Figure 4. Build the demo application

4. The build completes without errors.

3.2 Run an example application

Run a demo application using IAR

To download and run the application, perform these steps:

1. This board supports the J-Link debug probe. Before using it, install SEGGER software, which can be downloaded from www.segger.com/jlink-software.html.
2. Connect the development platform to your PC via USB cable between the USB-UART MICRO USB connector and the PC USB connector, then connect 5 V power supply and J-Link Plus to the device.
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

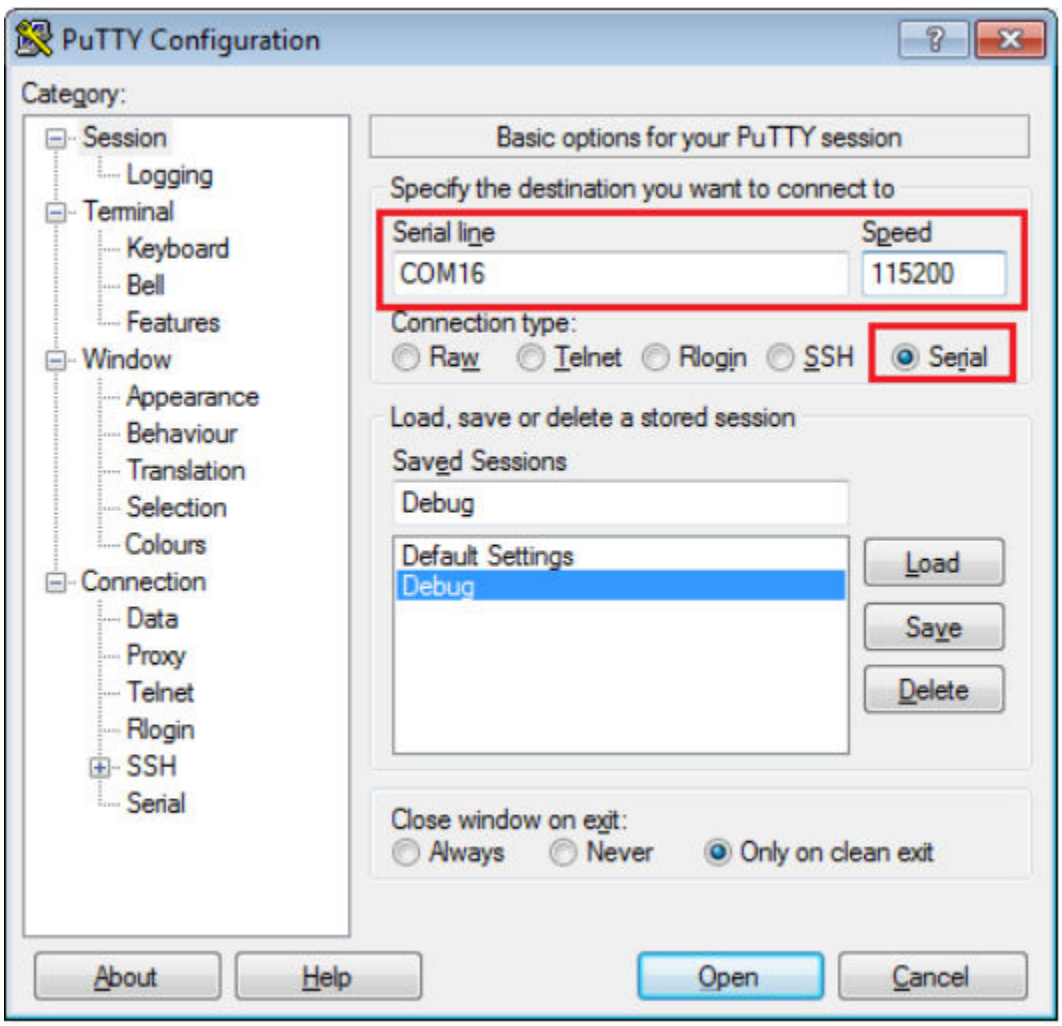

Figure 5. Terminal (PuTTY) configuration

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

Figure 6. Download and Debug button

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

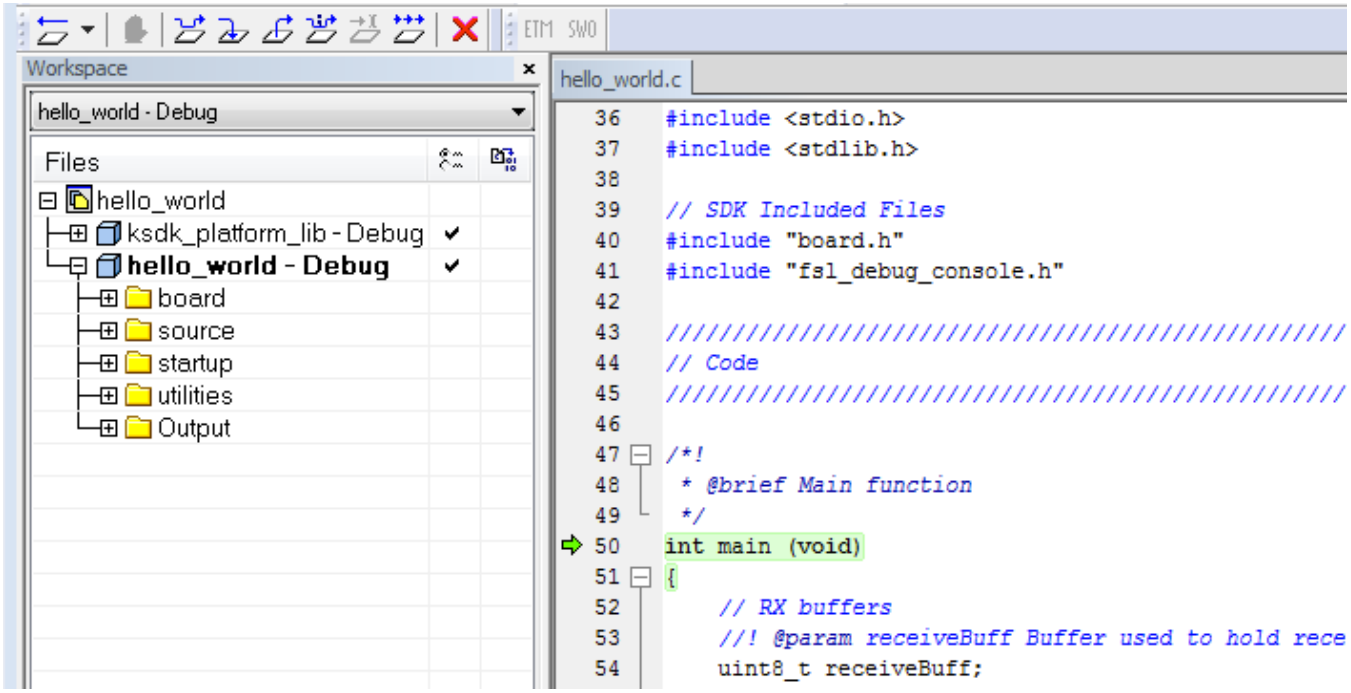

Figure 7. Stop at main() when running debugging

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

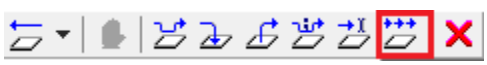

Figure 8. Go button

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.

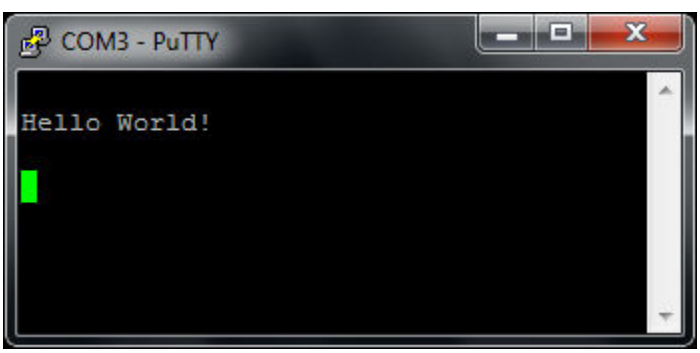

Figure 9. Text display of the hello_world demo

NOTE

If you want to debug QSPI XIP program, use J-Link V6.14 or newer.

4 Run a demo using Manufacturing Tool (MFGTool)

The manufacturing tool, named MFGTool, is a tool that runs on a computer and is used to download images to different devices on an i.MX board. The tar.gz file can be found with the pre-built images.

4.1 Configuring Manufacturing Tool (MFGTool)

The following steps describe how to configure the MFGTool:

1. Download IMX6_L4.1.15_2.0.0_MFG_TOOL from www.nxp.com
2. Extract IMX6_L4.1.15_2.0.0_MFG_TOOL to get mfgtools-with-rootfs and mfgtools-without-rootfs, then continue to extract mfgtools-with-rootfs to get the mfgtools folder. Override the contents in the mfgtools folder with the files provided in the *<sdk_dir>/tools/mfgtools* folder. It is also important to replace the ucl2.xml file.

4.2 Using Manufacturing Tool (MFGTool)

The following steps describe how to use the MFGTool:

1. Build the application and copy the built binary (.bin file) to the *<sdk_dir>/tools/imgutil/<board>* folder and rename to sdk20-app.bin.
2. In the *sdk_dir/tools/imgutil/<board>* folder, run mkimage.sh in mingw32 shell to get bootable image file sdk20-app.img.
   a. **QSPI image:** If the application is built with the RAM link file and wants to be loaded from flash to RAM then run, use "mkimage.sh ram" to create the bootable image.
   b. **QSPI XIP image:** If the application is built with the flash link file and wants to run on flash directly, use "mkimage.sh flash" to create the bootable XIP image.
   c. **SD image:** If the application is built with the RAM link file and wants to be loaded from SD to RAM and run, use "mkimage.sh sd" to create the bootable image.
3. Copy sdk20-app.img file made with imgutil to *Profiles/Linux/OS Firmware/files* folder in MFGTool.
4. Connect a USB cable from a computer to the USB OTG port on the board.
5. Connect a USB cable from the OTG-to-UART port to the computer for console output.
6. Set the boot pin to Serial download mode (see the *Quick Start Guide, Evaluation Kit, Based on i.MX 6UltraLite Applications Processor*). Run mfgtool2-sdk20-mx6ul-evk-qspi-nor-n25q256a.vbs to write sdk20-app.img (built with "mkimage.sh ram" or "mkimage.sh flash") to QSPI flash or run mfgtool2-sdk20-mx6ul-evk-sdcard.vbs to write sdk20-app.img(built with "mkimage.sh sd") to MicroSD.
7. Switch boot mode to Internal Boot and set boot devices to QSPI flash or MicroSD card, then power on the board. Then, the application should be running.

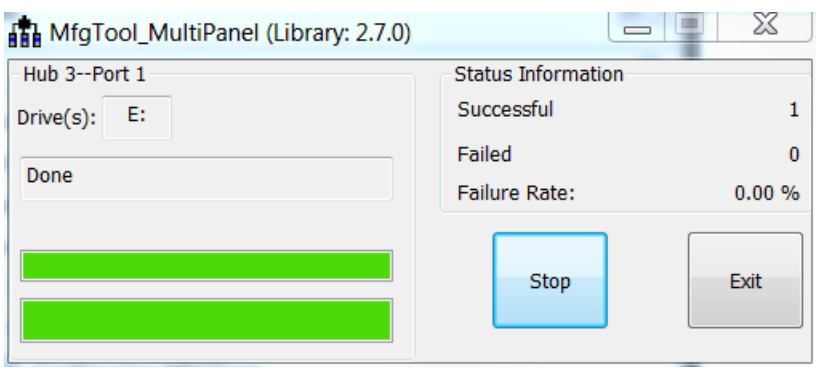

Figure 10. Programming QSPI flash with the manufacturing tool -- image downloading

NOTE

The readme.txt will be found in the imgutil folder and mfgtools folder.

5 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.

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:

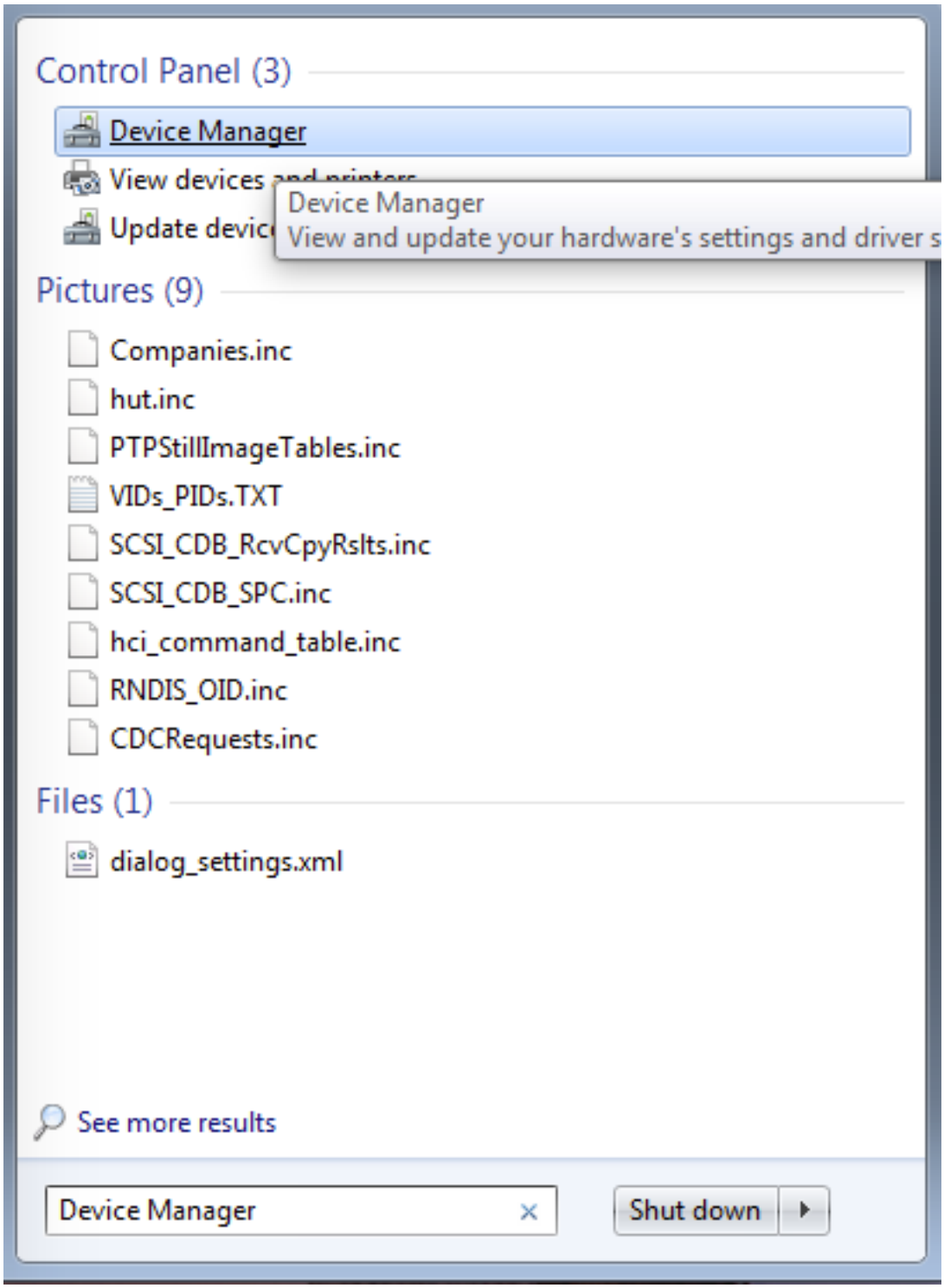

Figure 11. Device manager

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

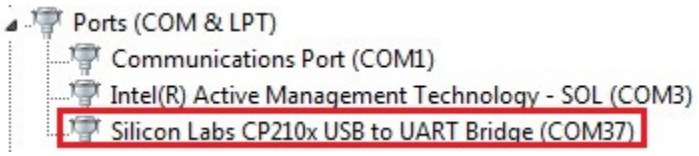

Figure 12. USB-UART interface

6 Revision history

This table summarizes revisions to this document.

Table 1. Revision history

Revision number	Date	Substantive changes
0	03/2017	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number MCUXSDKIMX6ULGSUG
Revision 0, 03/2017