

Integration of the USB Stack and Kinetis SDK

1 Overview

This document describes how to compile the USB stack and examples, download a binary image, and run the examples. The TWR-K22F120M Tower System module is used as an example board.

Contents

| | | |
|----------|--|-----------|
| 1 | Overview | 1 |
| 2 | Requirements for Building USB Examples | 2 |
| 3 | Board Jumper Settings | 3 |
| 4 | USB Code Structure | 7 |
| 5 | Compiling or Running the USB Stack and Examples | 8 |
| 6 | USB Stack Configuration | 29 |
| 7 | Revision History | 32 |

2 Requirements for Building USB Examples

The TWR-K22F120M Tower System module is used as an example in this document. Compiling, downloading, and running examples are similar on all other boards.

2.1 Hardware

- TWR-K22F120M Tower System module
- (Optional) TWR-SER Tower System module and Elevator
- J-Link debugger(optional)
- USB cables

2.2 Software

- KSDK release package
- IAR Embedded Workbench for ARM® Version 7.20.2
- Keil® µVision®5 Integrated Development Environment Version 5.11, available for Kinetis ARM Cortex®-M4 devices
- Kinetis Design Studio IDE v2.0
- Atollic® TrueSTUDIO® 5.2

3 Board Jumper Settings

This document focuses on the USB-related jumper settings on the board. For other jumper settings, see the board-related user's guide. The board jumper settings are provided for all supported boards.

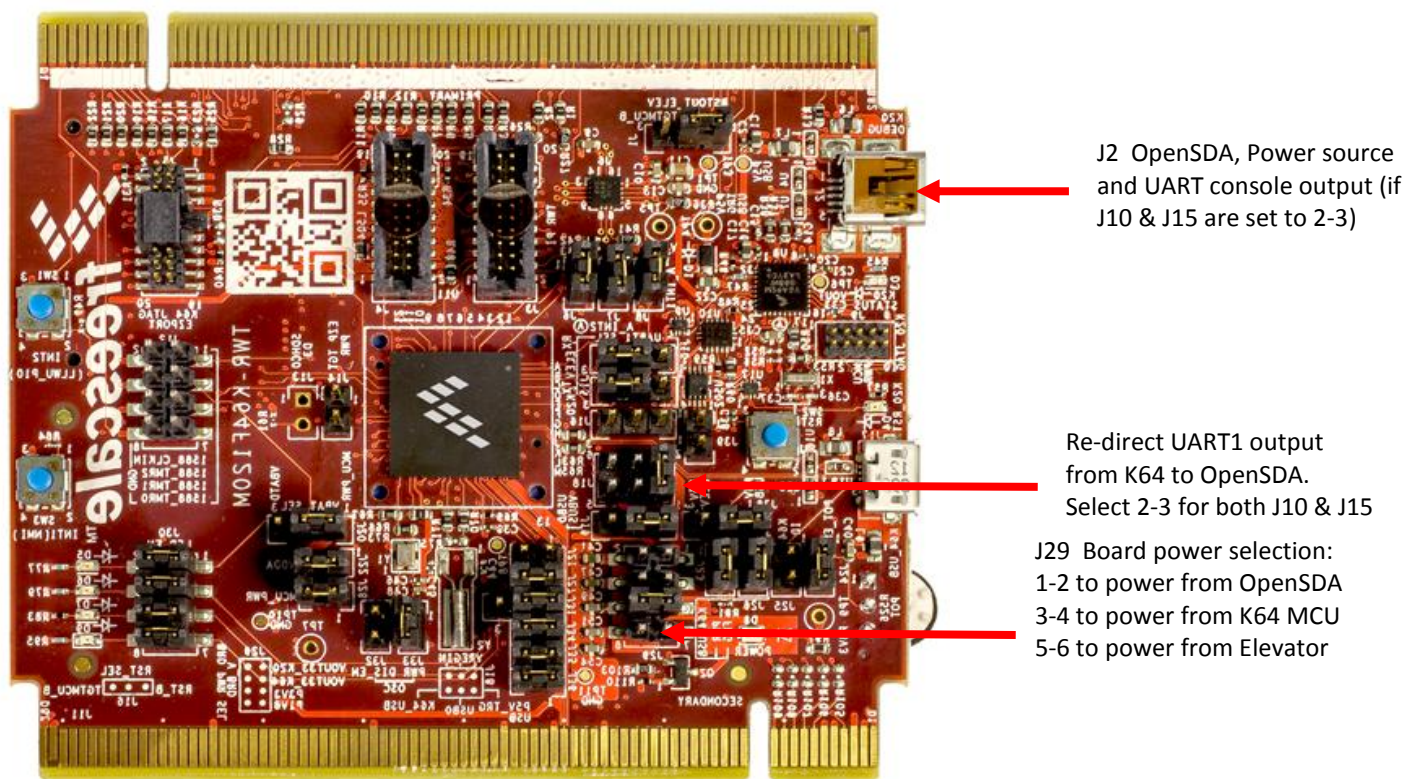


Figure-1 TWR-K64 Tower System module

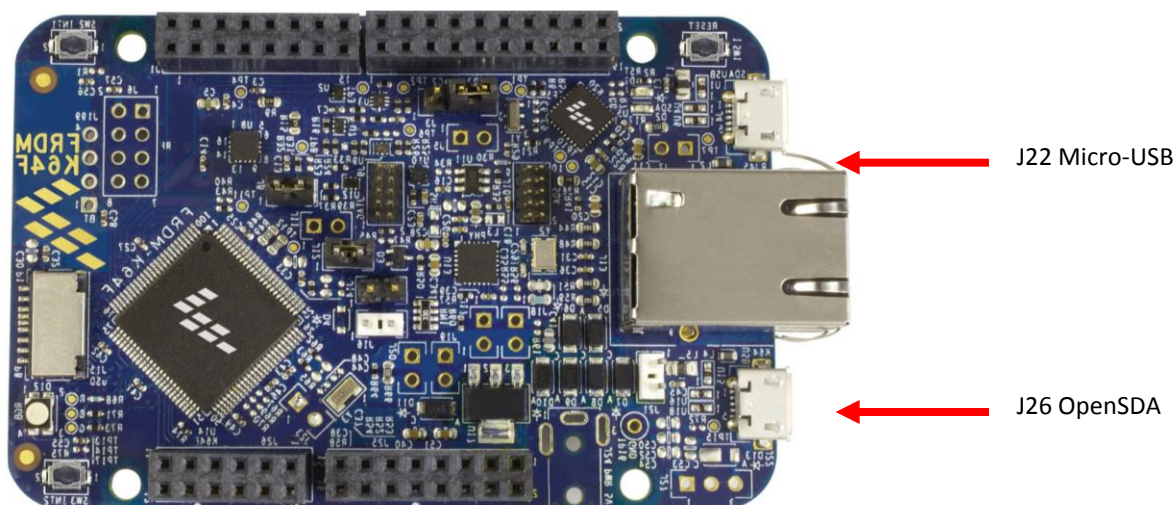


Figure-2 Freescale Freedom FRDM-K64F platform

- Connect J21
- Install a 2.2uF capacitor in the position C33 as shown.

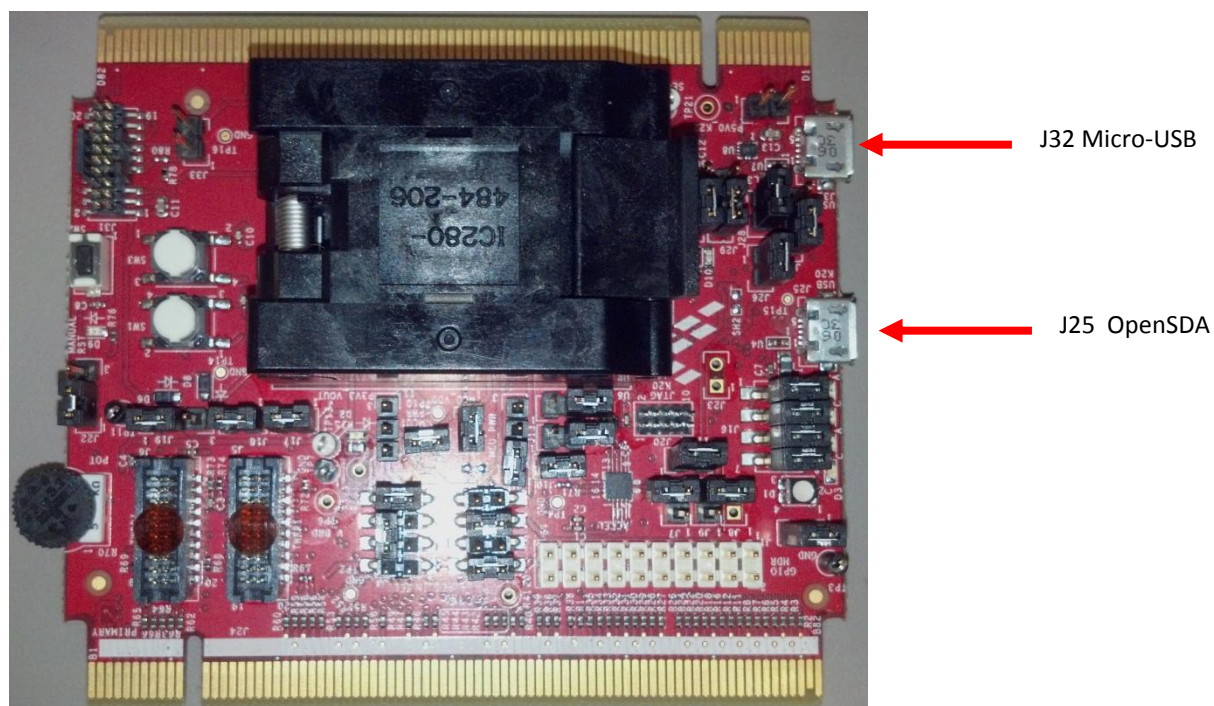
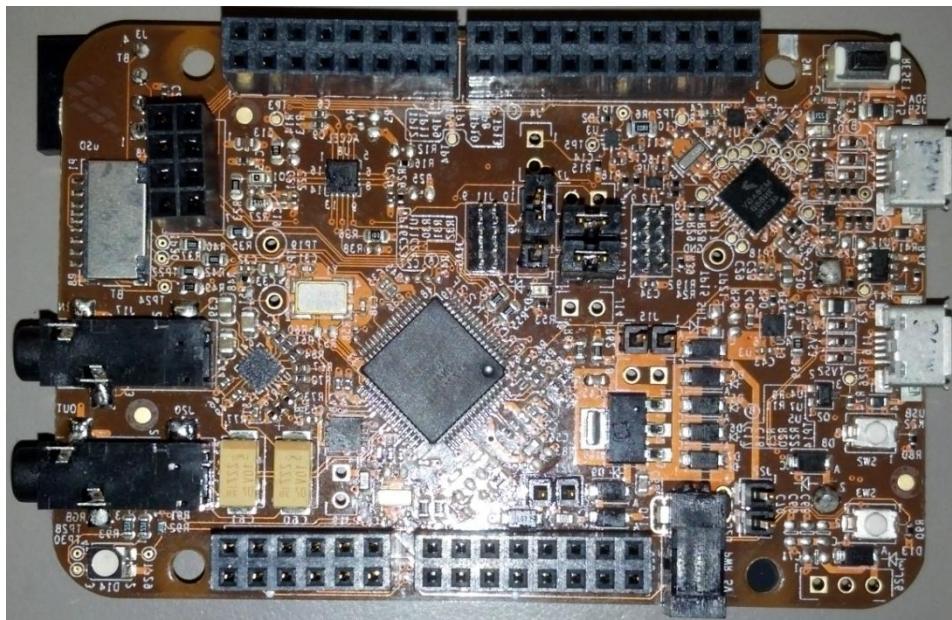


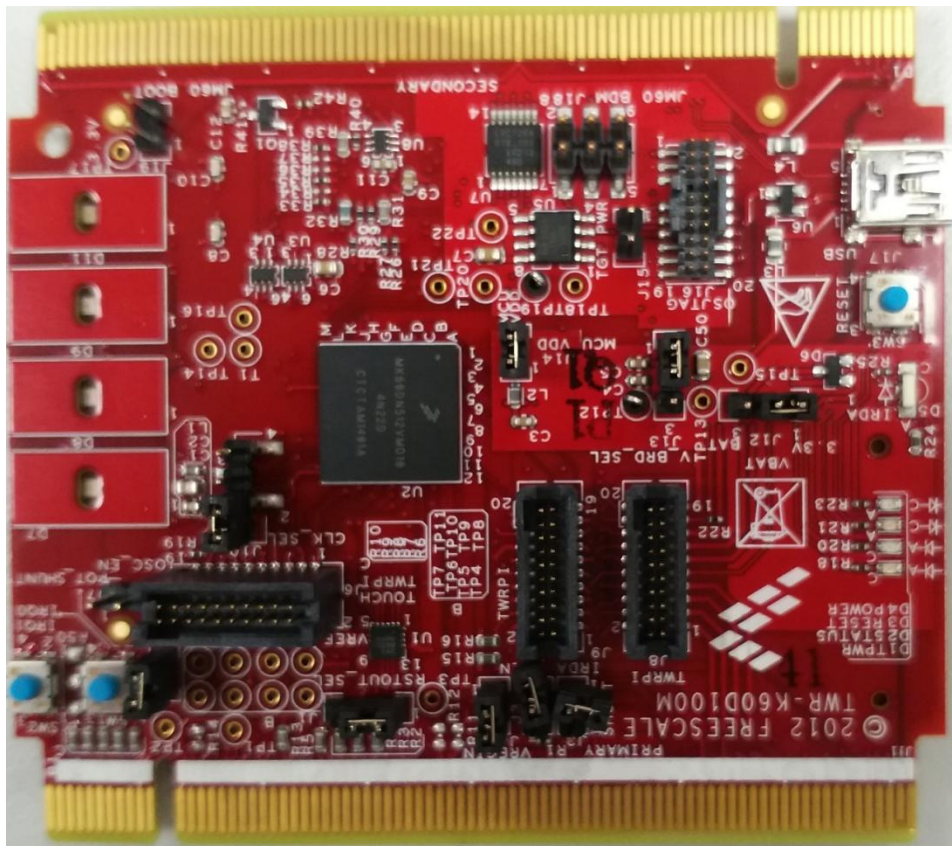
Figure-3 TWR-K22 Tower System module



J5 OpenSDA

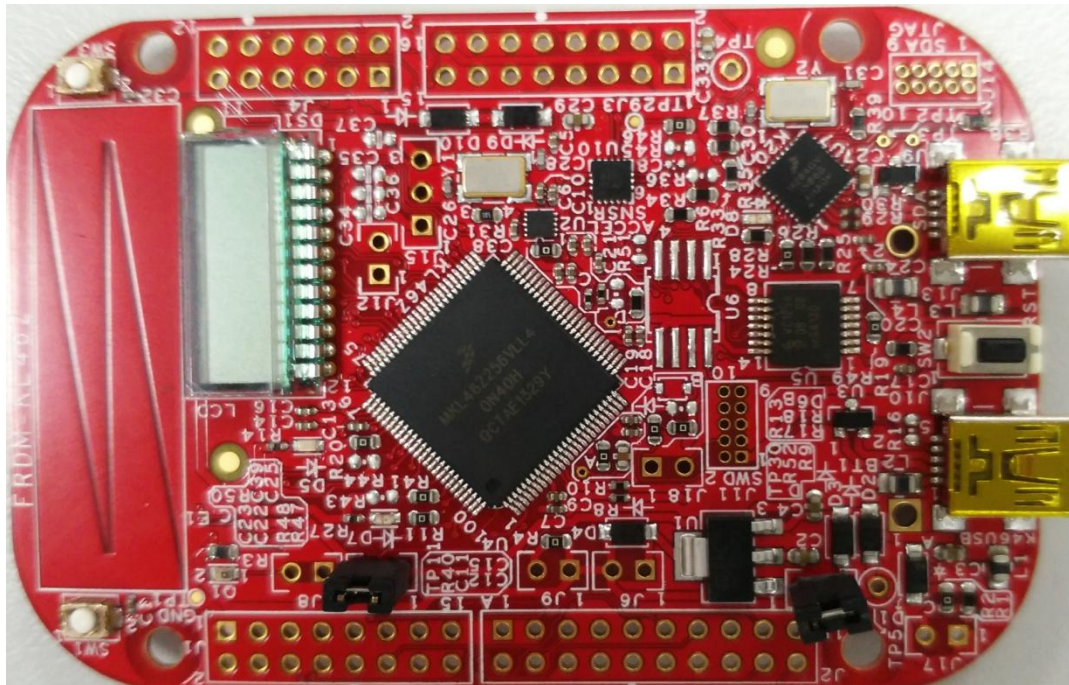
J16 Micro-USB

Figure-4 Freescale Freedom FRDM-K22 platform



J17 OpenSDA

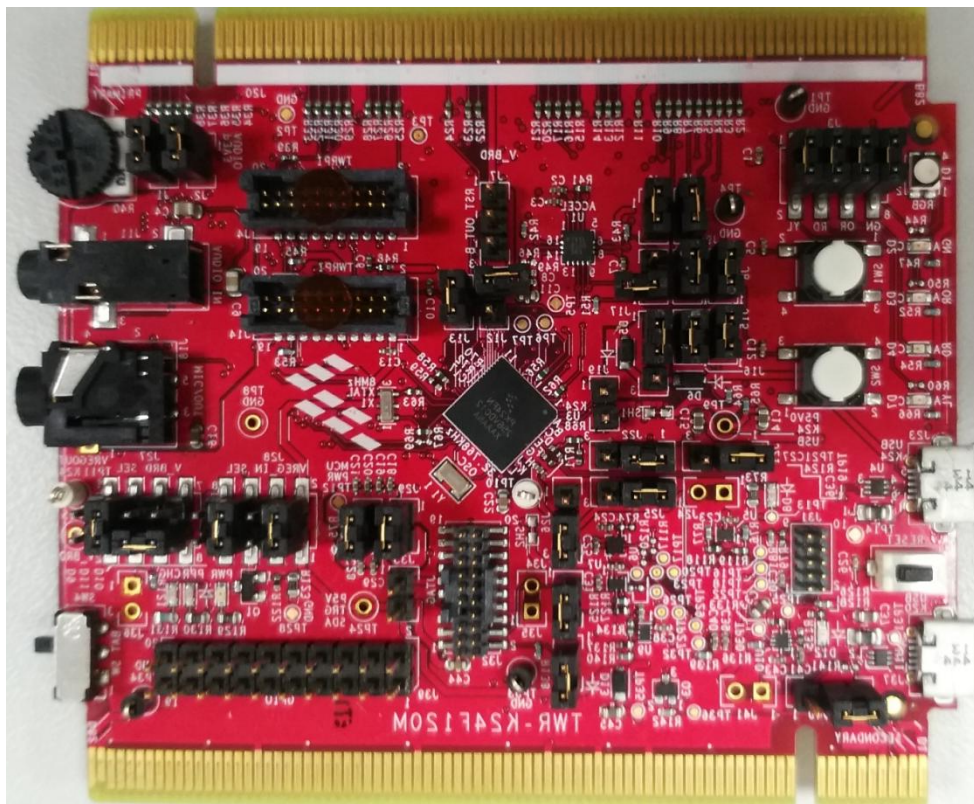
Figure-5 TWR-K60 Tower System module



J13 OpenSDA

J10 USB

Figure-6 Freescale Freedom FRDM-KL46 platform



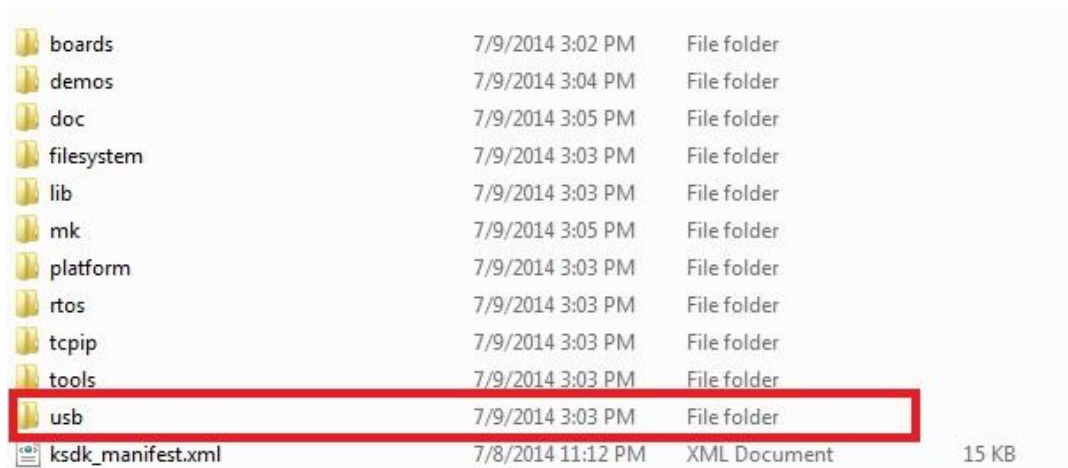
J23 Micro-USB

J37 OpenSDA

Figure-7 TWR-K24 Tower System module

4 USB Code Structure

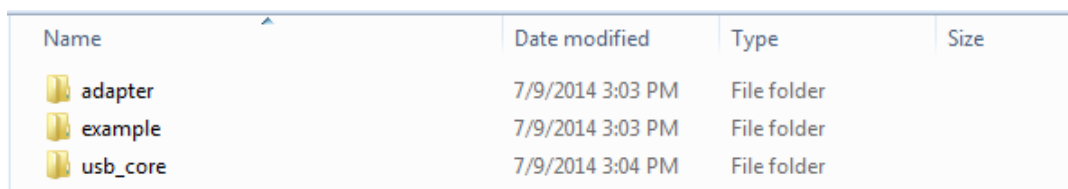
The USB code is located in the usb folder in the root level of the KSDK_1.1.0 folder.



| | | | |
|-------------------|-------------------|--------------|-------|
| boards | 7/9/2014 3:02 PM | File folder | |
| demos | 7/9/2014 3:04 PM | File folder | |
| doc | 7/9/2014 3:05 PM | File folder | |
| filesystem | 7/9/2014 3:03 PM | File folder | |
| lib | 7/9/2014 3:03 PM | File folder | |
| mk | 7/9/2014 3:05 PM | File folder | |
| platform | 7/9/2014 3:03 PM | File folder | |
| rtos | 7/9/2014 3:03 PM | File folder | |
| tcpip | 7/9/2014 3:03 PM | File folder | |
| tools | 7/9/2014 3:03 PM | File folder | |
| usb | 7/9/2014 3:03 PM | File folder | |
| ksdk_manifest.xml | 7/8/2014 11:12 PM | XML Document | 15 KB |

Figure-8 Kinetis SDK folder structure

The USB folder includes the source code, projects, and tools.



| Name | Date modified | Type | Size |
|----------|------------------|-------------|------|
| adapter | 7/9/2014 3:03 PM | File folder | |
| example | 7/9/2014 3:03 PM | File folder | |
| usb_core | 7/9/2014 3:04 PM | File folder | |

Figure-9 USB Folder Structure

The USB folder includes three subfolders:

- adapter

This subfolder includes the adapter files to support the USB stack running on a different RTOS with the same USB core code.

- example

This subfolder includes all source code and project files for the USB examples.

- usb_core

This subfolder includes the USB source files, such as HAL, controller driver, class drivers, and the USB library projects.

5 Compiling or Running the USB Stack and Examples

5.1 Step-by-step guide for IAR

This section shows how to use IAR. Open IAR as shown in this figure:

1. Open the workspace corresponding to different examples.

For example, the `host_hid_mouse_twrk22f120m_bm.eww` is located under the

`<Install_dir>/usb/example/host/hid/mouse/sdk/iar/host_hid_mouse_twrk22f120m_bm`.

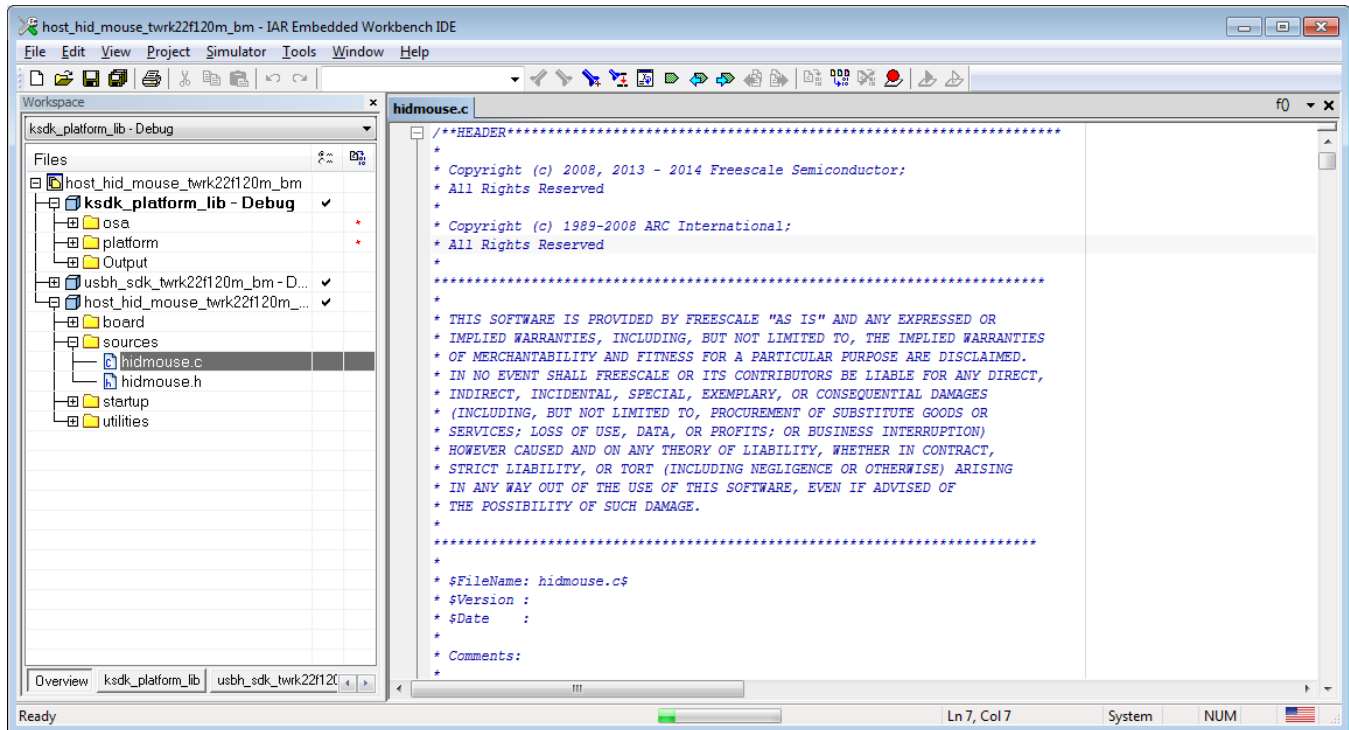


Figure-10 IAR workspace

2. Build the `ksdk_platform_lib` library.
3. Build the `usbh_sdk_twrk22f120m_bm` library.
4. Check the USB library build result.
5. After the USB library is built, the generated library binary file (`usbh.a`) is located in `output/twrk22f120m.iar/debug/usbh/sdk/`.
6. All USB-related public header files are copied to this folder.
7. Build the `host_hid_mouse_twrk22f120m_bm` example.
8. Connect the micro USB cable from a PC to the J25 of the TWR-K22F120M Tower System module to power on the board.
9. Click the "Download and Debug" button. Wait for the download to complete.

10. Click the “Go” button to run the example.

5.2 Step-by-step guide for KEIL

This section shows how to use KEIL. Open KEIL as shown in this figure:

1. Open the workspace corresponding to different examples.

For example, the `host_hid_mouse_twrk22f120m_bm.uvmpw` is located under the
<Install_dir>/usb/example/host/hid/mouse/sdk/mdk/host_hid_mouse_twrk22f120m_bm.

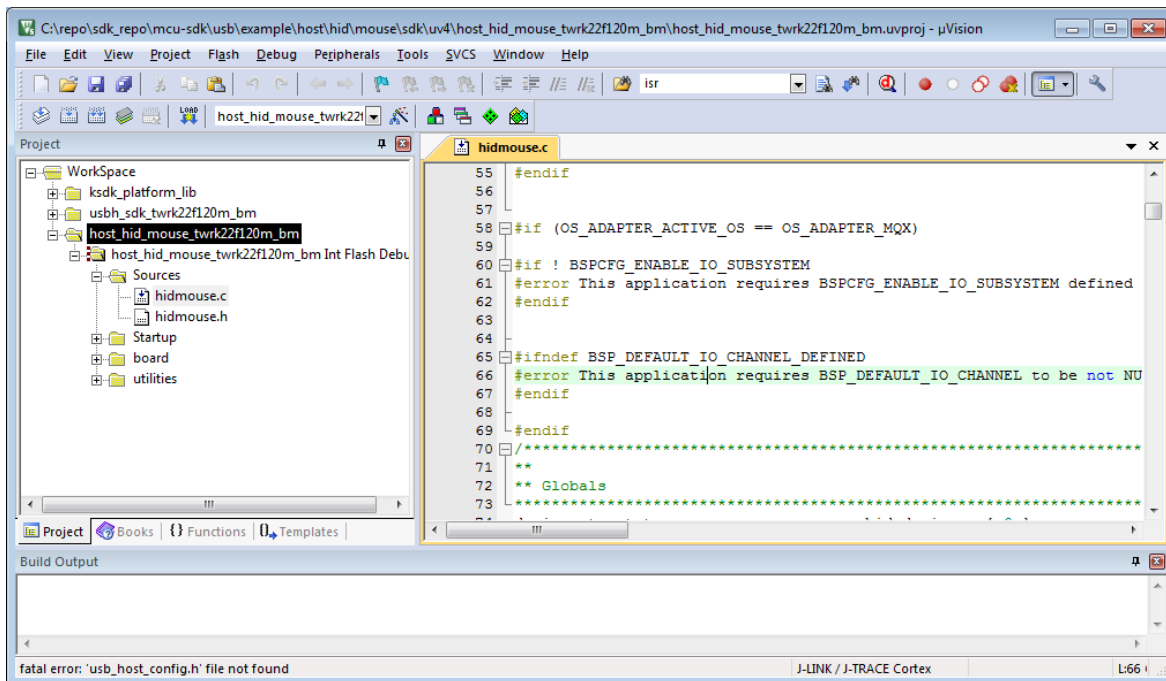


Figure-11 KEIL Workspace

2. Build the `ksdk_platform_lib` library.
3. Build the `usbh_sdk_twrk22f120m_bm` library.
4. Build the `host_hid_mouse_twrk22f120m_bm` example.
5. Click the “Start/Stop” debug session button. Wait for the download to complete.
6. Click the “Go” button to run the example.

5.3 Step-by-step guide for the Kinetis Design Studio

1. Unlike IAR or KEIL, the Kinetis Design Studio doesn't have a workspace. As a result, create a workspace and import Kinetis Design Studio USB examples, platform libraries, and the USB stack library.

2. Select “File” then “Import” from the KDS IDE Eclipse menu.
3. Expand the General folder and select “Existing Projects into Workspace”. Then, click the “Next” button.

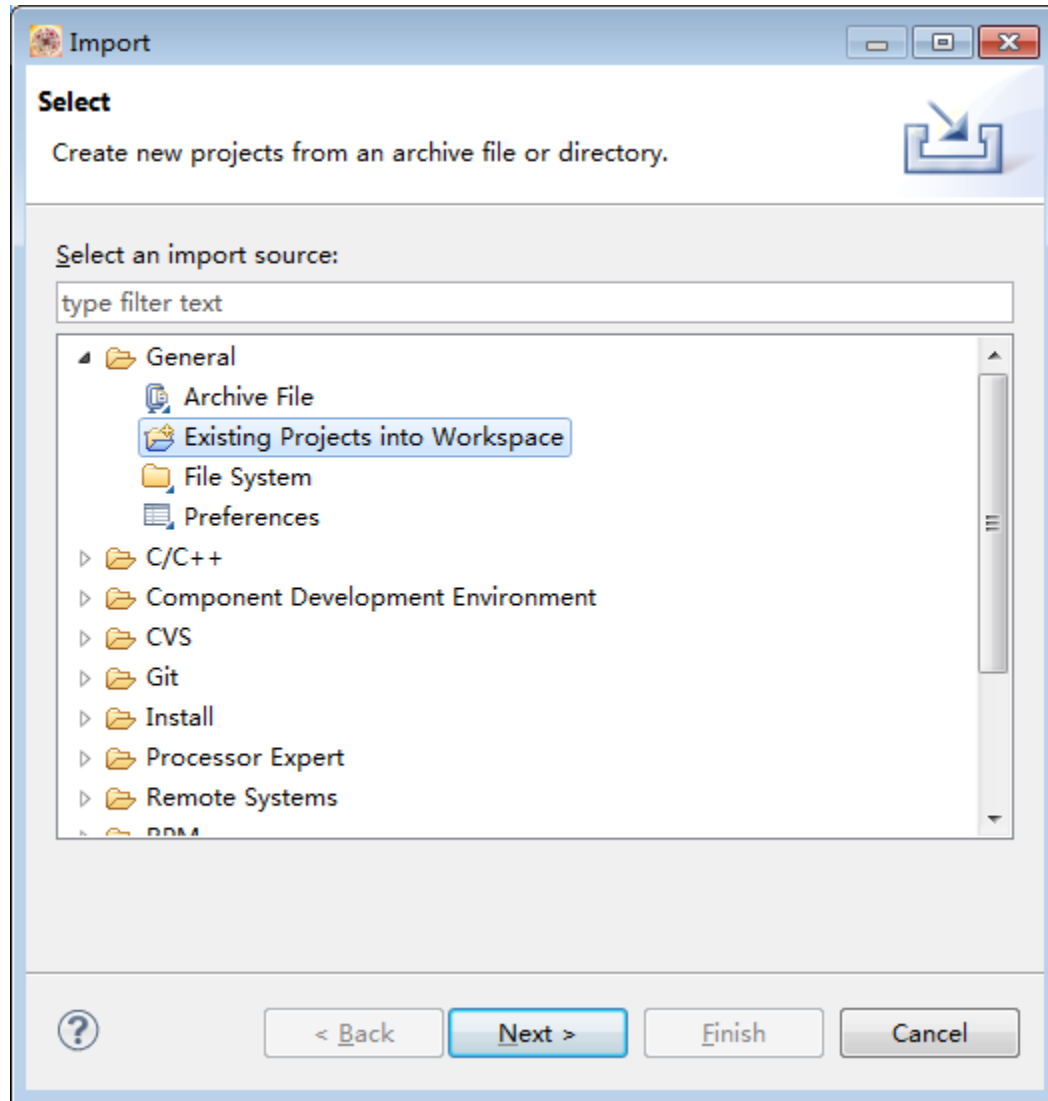


Figure-12 Selection of the correct import type in KDS IDE

4. Point the KDS IDE to the *ksdk_platform_lib* project in the K22. The import projects directory selection window should resemble this figure.

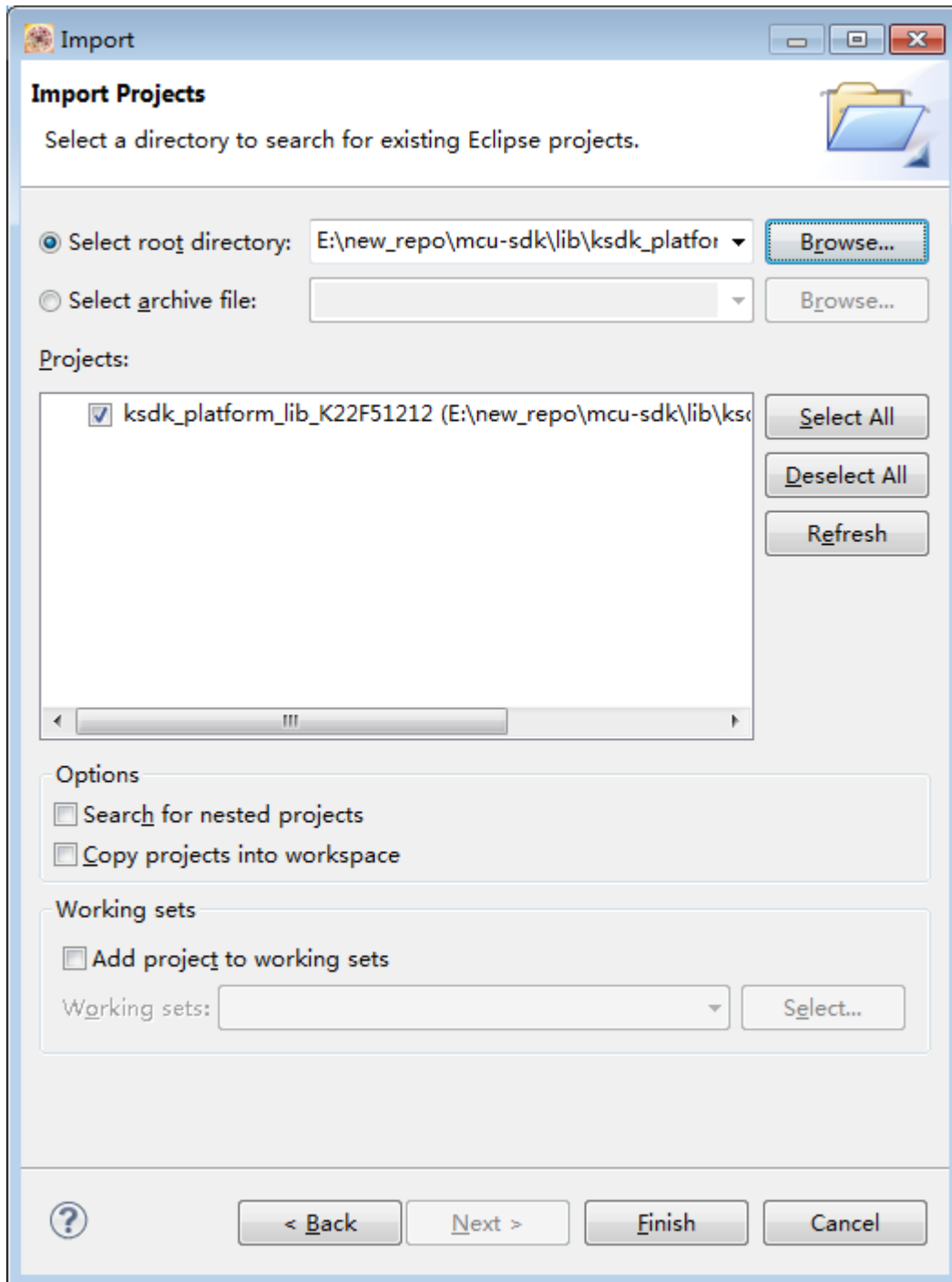


Figure-13 Selection of the K22 ksdk_platform_lib project

- Following the same step to import the USB device/host library and the USB example, after import them, the window should like this.

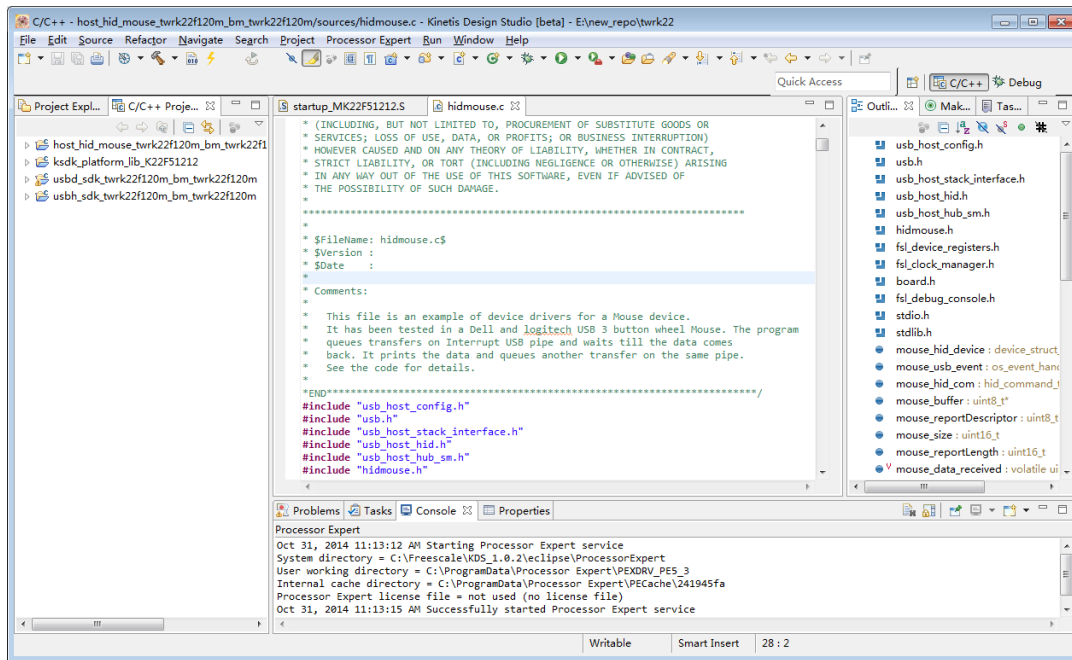


Figure-14 The USB projects workspace

- Choose the appropriate build target: “Debug” or “Release” by left-clicking the arrow next to the hammer icon as shown here.

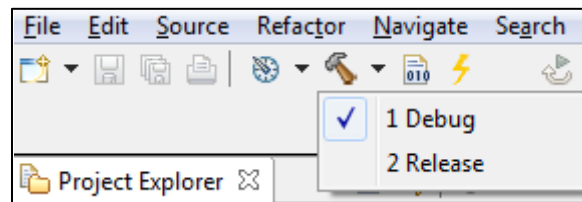


Figure-15 The hammer button

- If the library build does not begin after selecting the desired target, left-click the hammer icon to start the build.
- Following the same step to build the `ksdk_platform_lib` library, the `usbh_sdk_twrk22f120m_bm` library and the `host_hid_mouse_twrk22f120m_bm` example.

9. To check the debugger configurations, click the down arrow next to the green debug button and select “Debug Configurations”.

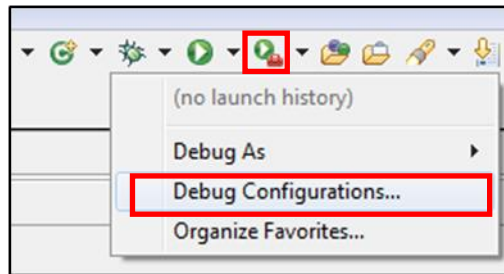


Figure-16 Debug configurations

10. After verifying that the debugger configurations are correct, click the “Debug” button.

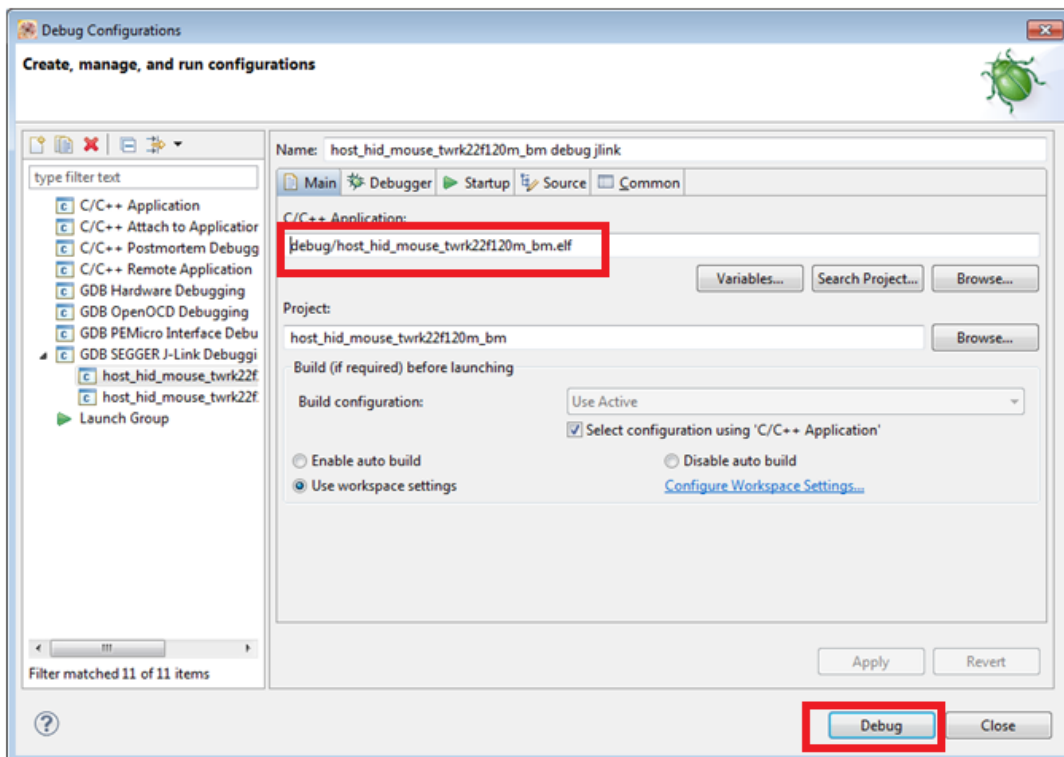


Figure-17 Kinetis Design Studio Debug configurations

11. The application is downloaded to the target and automatically run to main():
12. Run the code by clicking the “Resume” button to start the application:

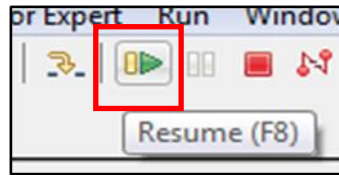


Figure-17 Resume button

5.4 Step-by-step guide for the Atollic TrueSTUDIO

1. Unlike IAR or KEIL, the Atollic TrueSTUDIO does not have a workspace. As a result, create a workspace and import Atollic TrueSTUDIO USB examples, platform libraries, and the USB stack library.
2. Select “File” then “Import...” from the Atollic TrueSTUDIO IDE Eclipse menu.

3. Expand the General folder and select “Existing Projects into Workspace”. Then, click the “Next” button.

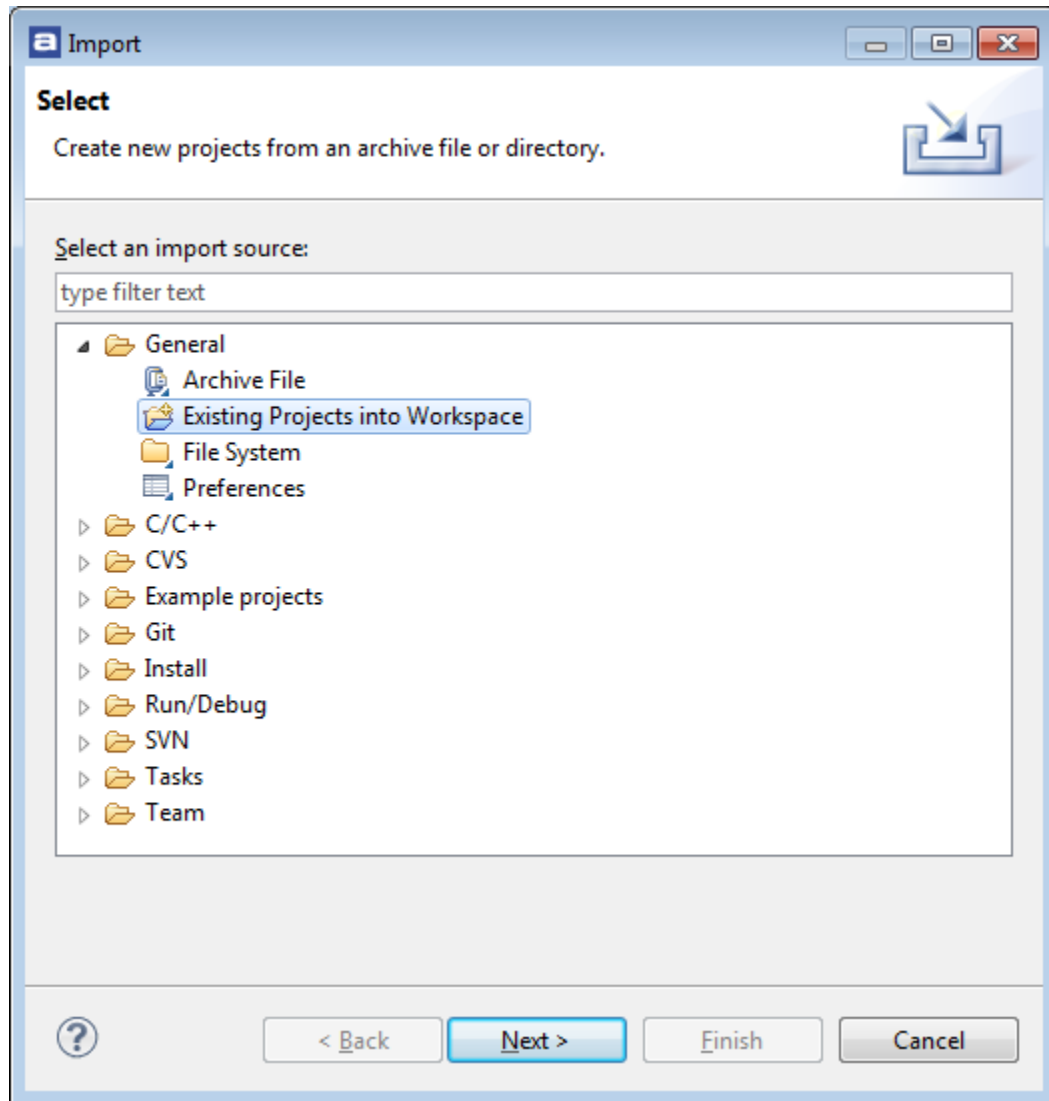


Figure-18 Selection of the correct import type in Atollic TrueSTUDIO IDE

4. Point the Atollic TrueSTUDIO IDE to the *ksdk_platform_lib* project in the K22. The import projects directory selection window should resemble this figure.

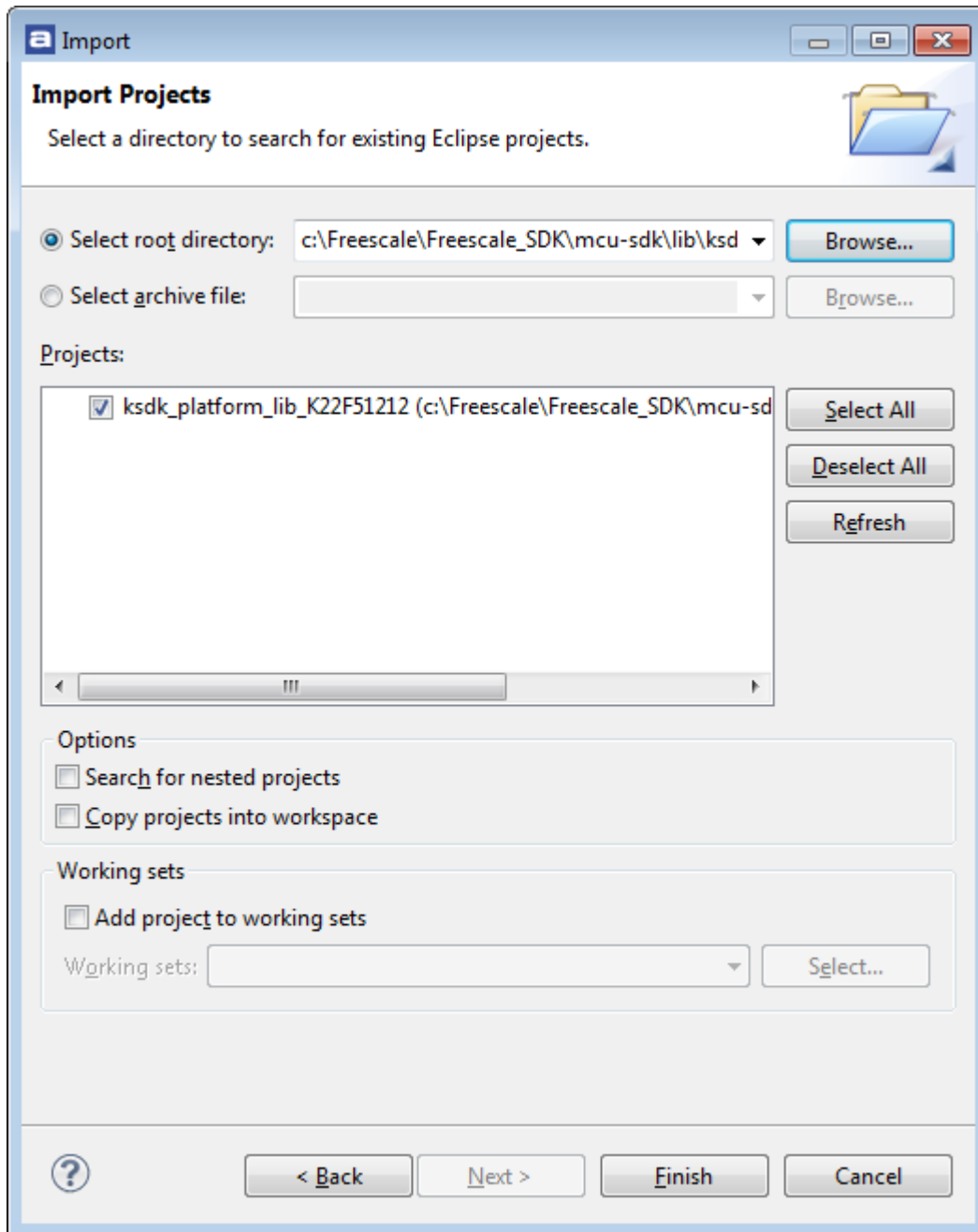


Figure-19 Selection of the K22 ksdk_platform_lib project

- Following the same step to import the USB device/host library and the USB example, after import them, the window should like this.

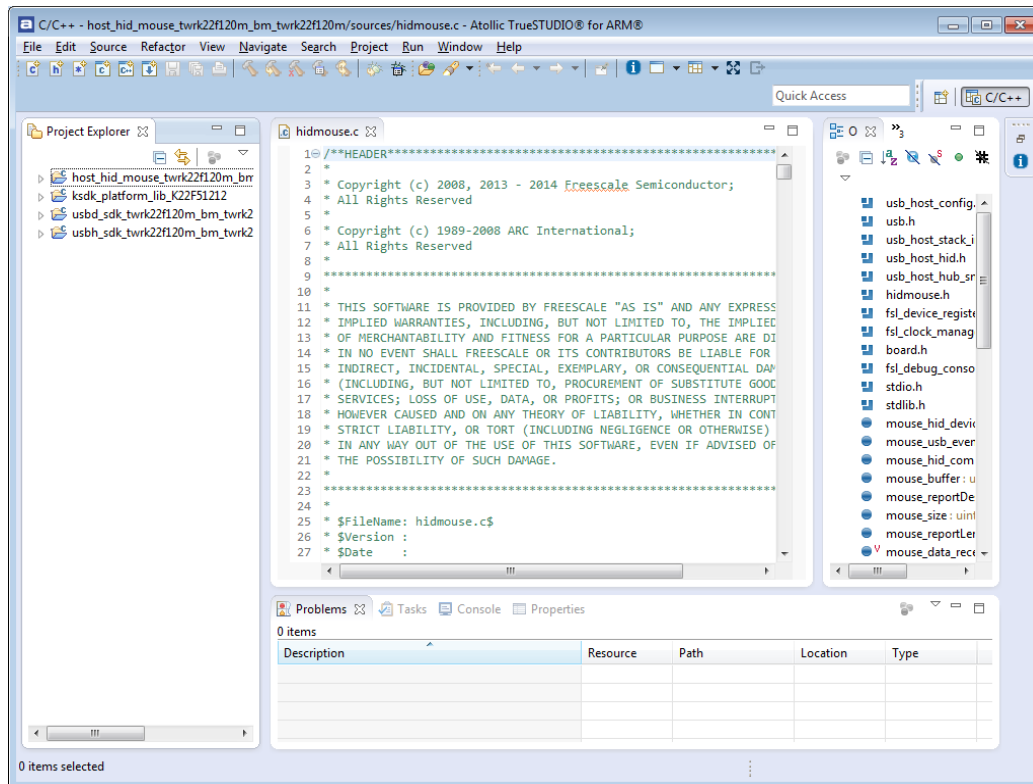


Figure-20 The USB projects workspace

- Choose the appropriate build target: “Debug” or “Release” by left-clicking the build configuration icon as shown here.

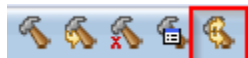


Figure-21 Manage build configuration button

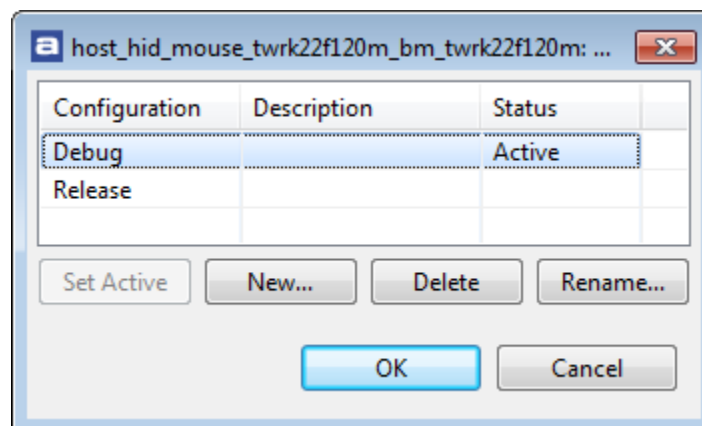


Figure-22 Set build configuration

7. If the library build does not begin after selecting the desired target, left-click the build icon to start the build.



Figure-22 Build project button

8. Following the same step to build the `ksdk_platform_lib` library, the `usbh_sdk_twrk22f120m_bm` library and the `host_hid_mouse_twrk22f120m_bm` example.
9. To check the debugger configurations, click the “Configure Debug” button.



Figure-23 Configure debug button

10. After verifying that the debugger configurations are correct, click the “Debug” button.

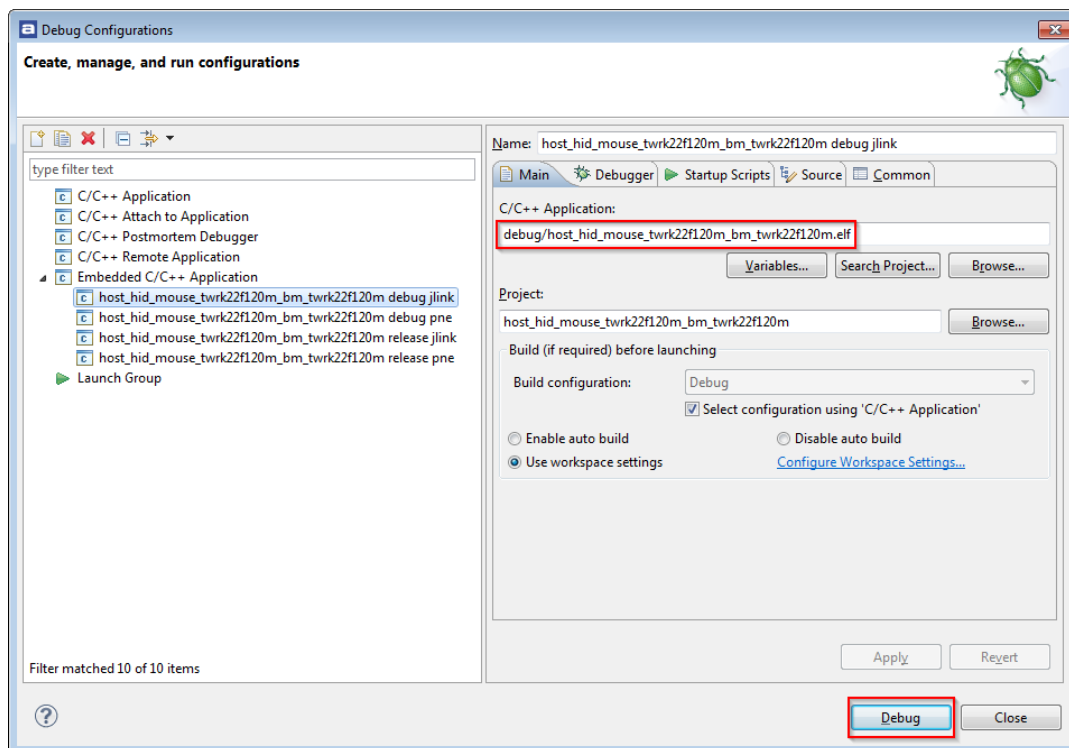


Figure-24 Atollic TrueSTUDIO Debug configurations

11. The application is downloaded to the target and automatically run to **main()**:
12. Run the code by clicking the “Resume” button to start the application:



Figure-25 The resume button

5.5 Step-by-step guide for the ARM GCC and KDS IDE GCC

5.5.1 Setup tool chains

5.5.1.1 Install GCC ARM Embedded tool chain

Download and install the installer from www.launchpad.net/gcc-arm-embedded.

5.5.1.2 Install MinGW

1. Download the latest mingw-get-setup.exe.
2. Install the GCC ARM Embedded toolchain. The recommended path is C:\MINGW, however, you may install to any location. Note that the installation path may not contain a space.
3. Ensure that the mingw32-base and msys-base are selected under Basic Setup.
4. Finally, click “Installation” and “Apply changes”.

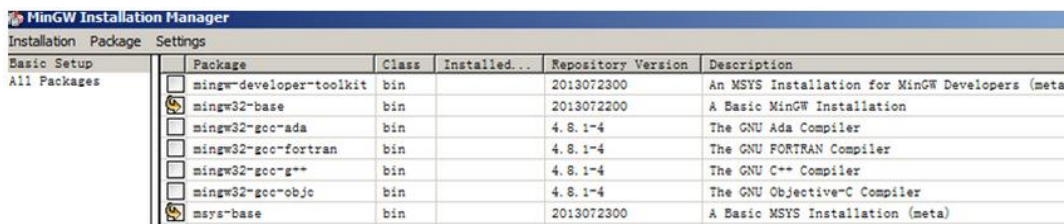


Figure 26: Setup MinGW and MSYS

5. Add paths C:\MINGW\msys\1.0\bin;C:\MINGW\bin to the system environment. Note that if the GCC aRM Embedded tool chain was installed somewhere other than the recommended location, the system paths added should reflect this change. An example using the recommended installation locations are shown below.

NOTE

There is a high chance that, if the paths are not set correctly, the tool chain will not work properly.

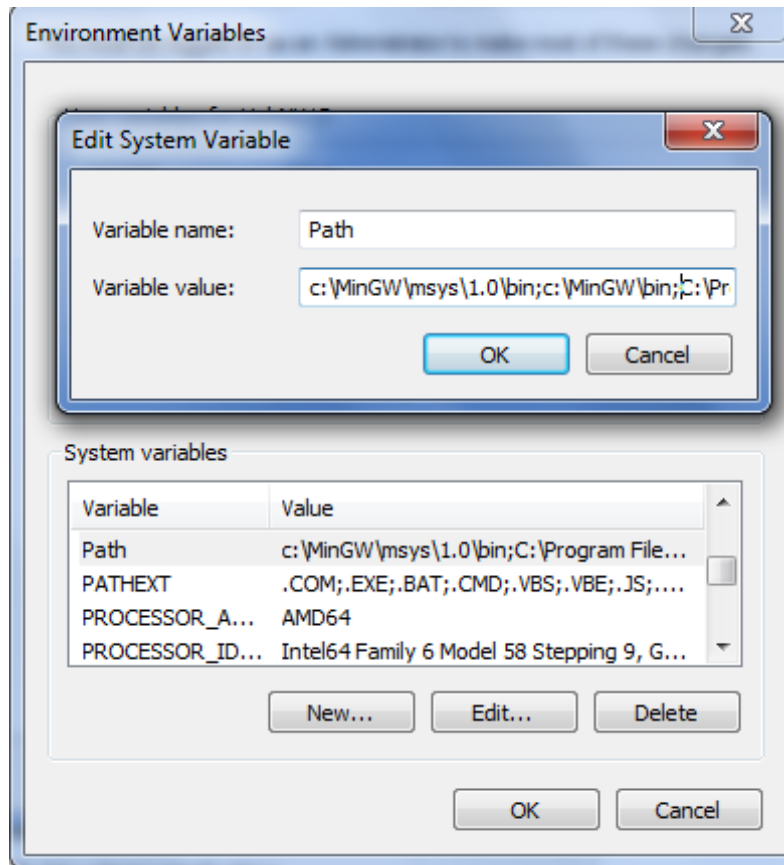


Figure 27: Add Path to systems environment

5.5.1.3 Add new system environment variable ARMGCC_DIR

Create a new system environment variable ARMGCC_DIR. The value of this variable should be the short name of the ARM GCC Embedded tool chain installation path.

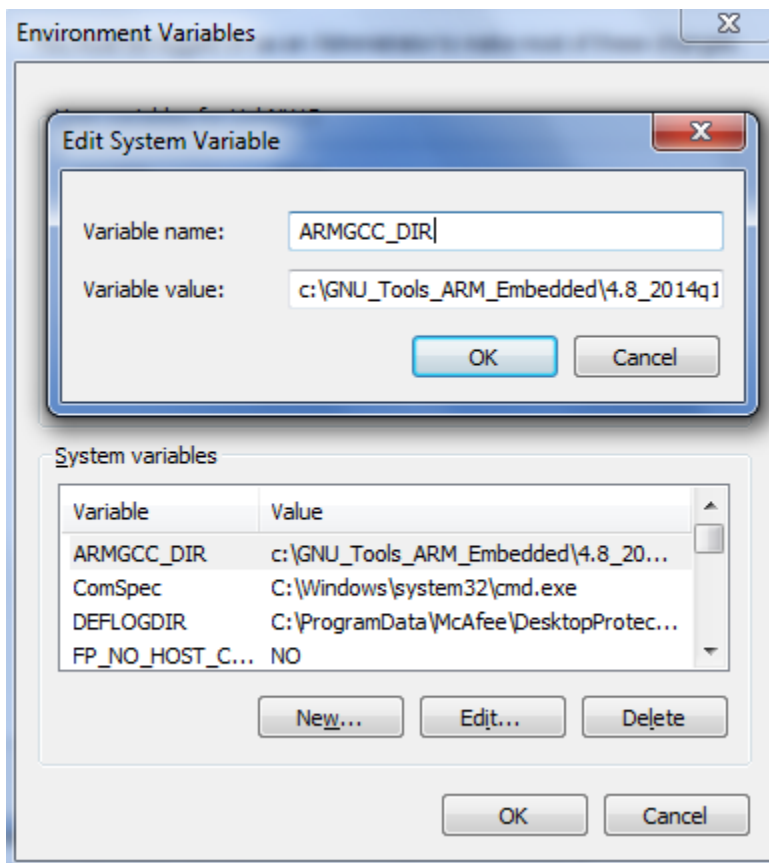


Figure 28: Add ARMGCC_DIR system variable

5.5.1.4 Add new system environment variable KDSGCC_DIR

Create a new system environment variable KDSGCC_DIR. The value for the variable is the name of the Kinetis Design Studio (KDS) IDE ARM GCC installation path. By default, KDS IDE is installed to the C:/Freescale/KDS_1.1.0/toolchain location.

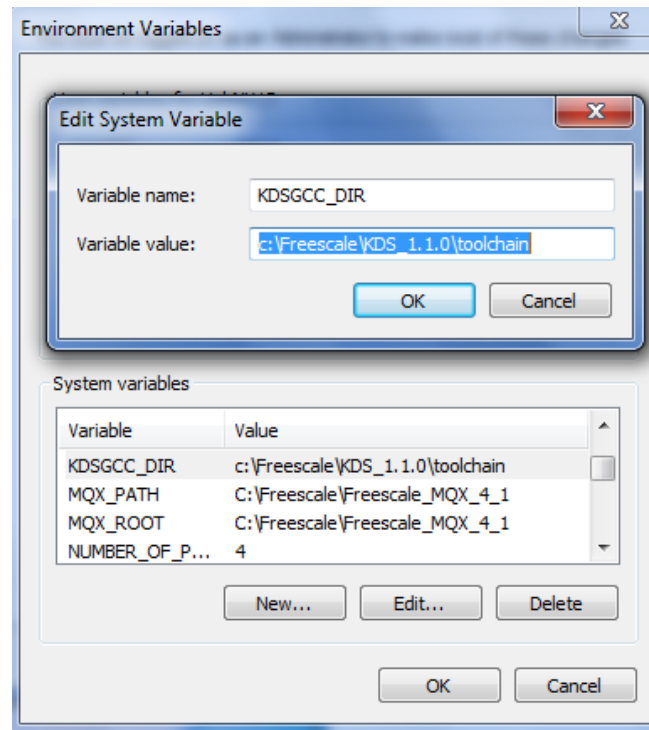


Figure 29: Add KDSGCC_DIR in system variable

5.5.1.5 Install CMake

1. Download CMake 3.0.0 from www.cmake.org/cmake/resources/software.html.
2. Install Cmake 3.0.0 (ensure that the option "Add CMake to system PATH" is selected when installing).

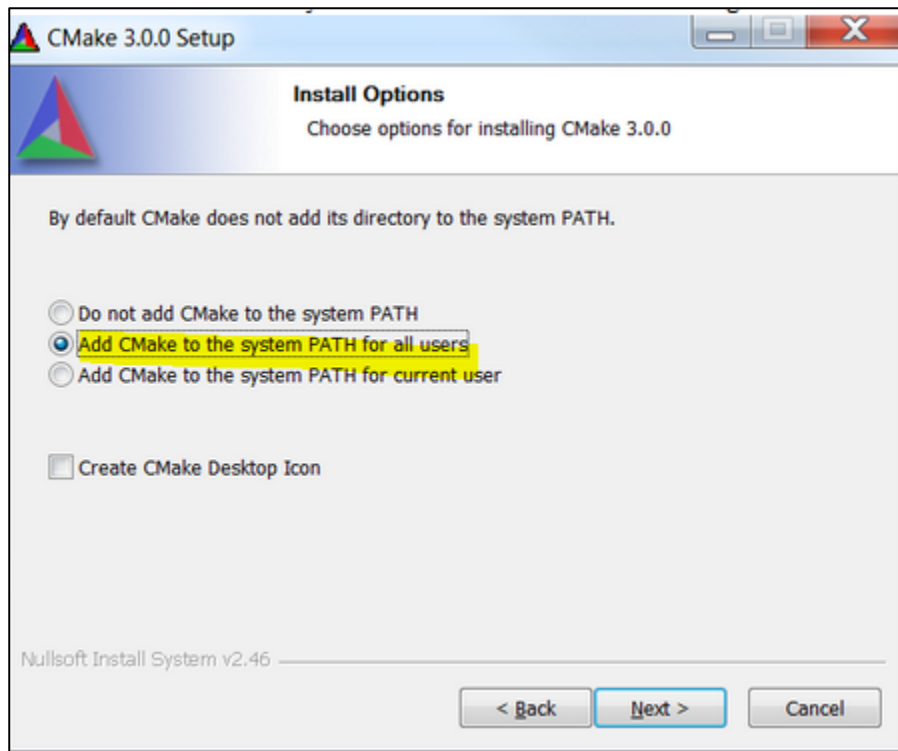


Figure 30: Install CMake

5.5.2 Build the platform driver library

To build the platform library, follow these instructions:

- a. Open a GCC ARM Embedded tool chain command window.
- b. Change the directory of the command window to the platform lib directory in the KSDK (one of these):

`<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/`

`<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/`

- c. Type “build_all”.

- d. Type “build_all”. CMake builds both the “debug” and the “release” target.

```
[ 95%] [ 96%] Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/uart/fsl_uart_driver.c.obj
Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/uart/fsl_uart_dma_driver.c.obj
[ 97%] [ 98%] Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/wdog/fsl_wdog_driver.c.obj
Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/platform/hal/src/wdog/fsl_wdog_hal.c.obj
[100%] Building C object CMakeFiles/KsdkPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/wdog/fsl_wdog_common.c.obj
Linking C static library release\libksdk_platform.a
[100%] Built target KsdkPlatformLib
c:\Freescale\Freescale_SDK\mcu-sdk\lib\ksdk_platform_lib\armgcc\K22F51212>pause
Press any key to continue . . .
```

Figure 31: ksdk_platform_lib build successfully

- e. The library (ksdk_platform_lib.a) is generated in these directories according to the build target:

```
<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/Debug
<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/Debug
<install_dir>/lib/ksdk_platform_lib/kdsgcc/<device_name>/Release
<install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/Release
```

5.5.3 Build the USB host/device library

1. Change the directory to the project directory:

```
<install_dir>/usb/usb_core/device/build/armgcc/usbd_sdk_twrk22f120m_bm
<install_dir>/usb/usb_core/device/build/kdsgcc/usbd_sdk_twrk22f120m_bm
<install_dir>/usb/usb_core/host/build/armgcc/usbd_sdk_twrk22f120m_bm
<install_dir>/usb/usb_core/host/build/kdsgcc/usbd_sdk_twrk22f120m_bm
```

2. Run the build_all.bat as described in steps 3 and 4 of section 5.5.2. The build output is shown in this figure:

```
[ 89%] [ 94%] Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/host/sources/controller/usb_host_dev_mng.c.obj
Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/host/sources/controller/khci/khci.c.obj
[100%] Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C:/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/host/sources/bsp/twrk22f120m/usb_host_bsp.c.obj
Linking C static library release\libusbh_bm.a
[100%] Built target KsdkUsbHostPlatformLib
c:\Freescale\Freescale_SDK\mcu-sdk\usb\usb_core\host\build\armgcc\usbh_sdk_twrk22f120m_bm>pause
Press any key to continue . . .
```

Figure 32: USB host library build successfully

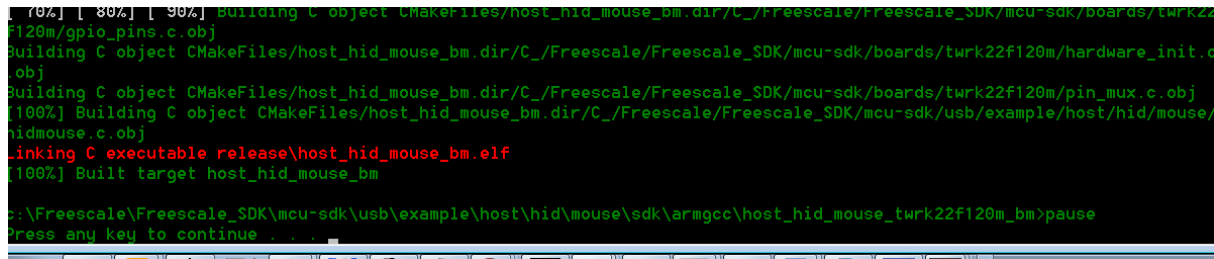
5.5.4 Build the USB demo

3. Change the directory to the project directory:

`<install_dir>/usb/example/host/hid/mouse/sdk/armgcc/host_hid_mouse_twrk22f120m_bm`

`<install_dir>/usb/example/host/hid/mouse/sdk/kdsgcc/host_hid_mouse_twrk22f120m_bm`

4. Run the build_all.bat as described in steps 3 and 4 of section 5.5.2. The build output is shown in this figure:



```
[ 70%] [ 80%] [ 90%] Building C object CMakeFiles/host_hid_mouse_bm.dir/C:/Freescale/Freescale_SDK/mcu-sdk/boards/twrk22f120m/gpio_pins.c.obj
Building C object CMakeFiles/host_hid_mouse_bm.dir/C:/Freescale/Freescale_SDK/mcu-sdk/boards/twrk22f120m/hardware_init.c.obj
Building C object CMakeFiles/host_hid_mouse_bm.dir/C:/Freescale/Freescale_SDK/mcu-sdk/boards/twrk22f120m/pin_mux.c.obj
[100%] Building C object CMakeFiles/host_hid_mouse_bm.dir/C:/Freescale/Freescale_SDK/mcu-sdk/usb/example/host/hid/mouse/hidmouse.c.obj
Linking C executable release\host_hid_mouse_bm.elf
[100%] Built target host_hid_mouse_bm

C:\Freescale\Freescale_SDK\mcu-sdk\usb\example\host\hid\mouse\sdk\armgcc\host_hid_mouse_twrk22f120m_bm>pause
Press any key to continue . . .
```

Figure 33: USB host demo build successfully

5.5.5 Run a demo application

This section describes steps to run a demo application using J-Link GDB Server application.

1. Connect the J-Link debug pod to the SWD/JTAG connector of the board.
2. Open the J-Link GDB Server application and modify your connection settings as shown in this figure.

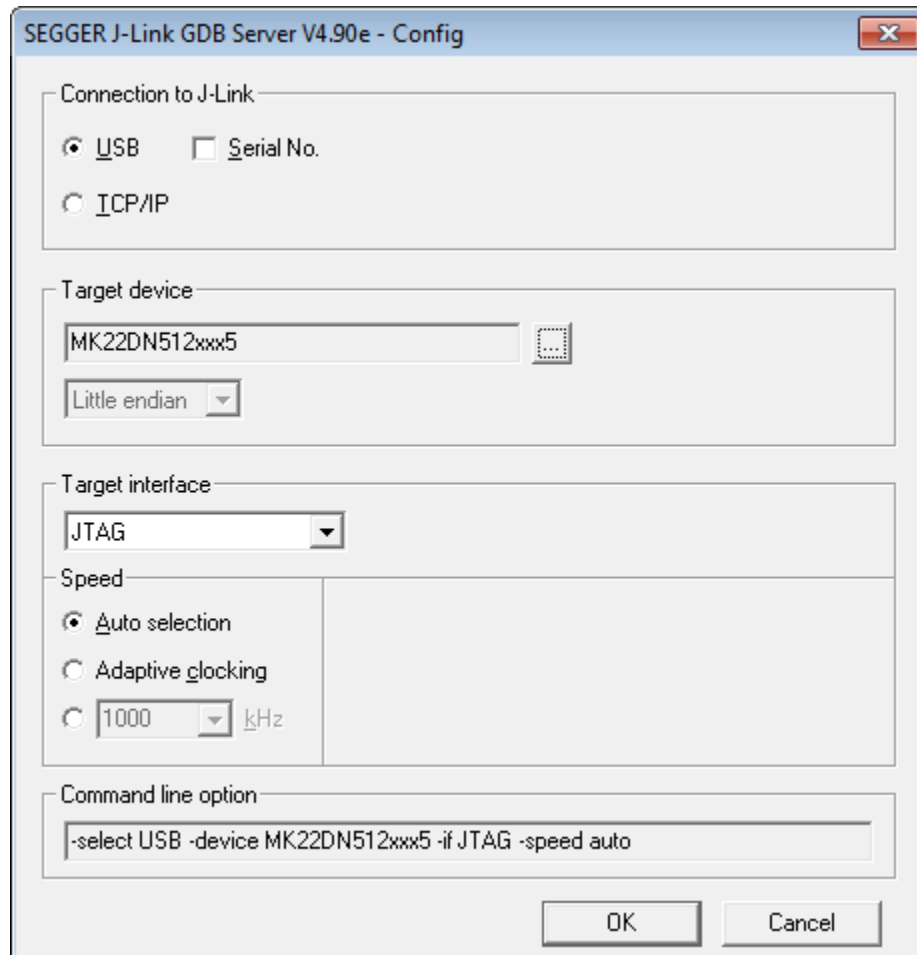


Figure 34: Segger J-Link GDB Server configuration

3. Once connected, the screen should resemble this figure:

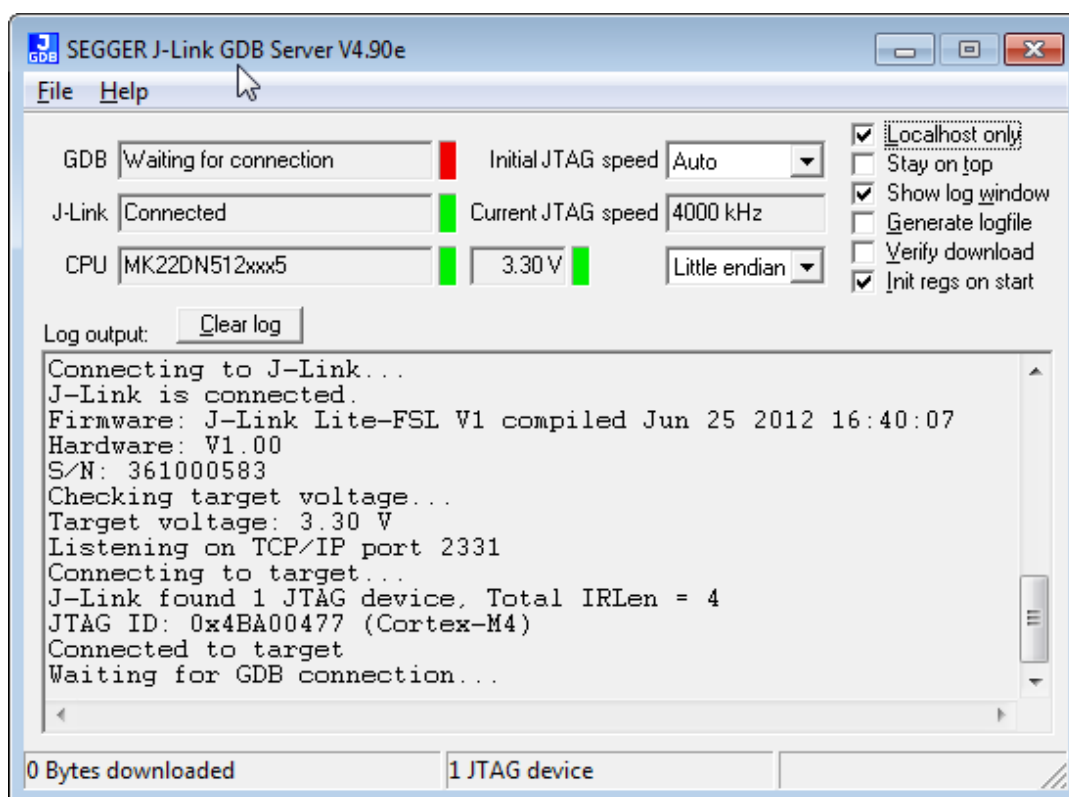


Figure 35: Segger J-Link GDB Server screen after successful connection

4. Open the ARM GCC command prompt and change the directory to the output directory of the desired demo. For this example, the directory is:

```
<install_dir>/usb/example/host/hid/mouse/sdk/armgcc/host_hid_mouse_twrk22f120m_bm/Debug
```


5. Run the command “arm-none-eabi-gdb.exe <DEMO_NAME>.elf”. For this example, it is “arm-none-eabi-gdb.exe host_hid_mouse_bm.elf”.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

c:\Freescale\Freescale_SDK\mcu-sdk\usb\example\host\hid\mouse\sdk\armgcc\host_hid_mouse_twrk22f120m_bm\debug>arm-none-eabi-gdb.exe host_hid_mouse_bm.elf
GNU gdb (GNU Tools for ARM Embedded Processors) 7.4.1.20130913-cus
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i586-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from c:\Freescale\Freescale_SDK\mcu-sdk\usb\example\host\hid\mouse\sdk\armgcc\host_hid_mouse_twrk22f120m_bm\debug\host_hid_mouse_bm.elf...done.
(gdb) _
```

Figure 36: Run arm-none-eabi-gdb

6. Run these commands:
 - a. “target remote localhost: 2331”
 - b. “monitor reset”
 - c. “monitor halt”
 - d. “load”
 - e. “monitor reset”
7. The application is downloaded and connected. Execute the “monitor go” command to start the demo application.

6 USB Stack Configuration

6.1 Device configuration

All device configurations are listed in this file:

```
usb_core/device/include/BOARD_NAME/usb_device_config.h
```

Replace BOARD_NAME with the name of the board.

This file is used to either enable or disable the USB class driver. The object number is configurable either to decrease the memory usage or to meet specific requirements.

If the device stack configuration is changed, rebuild both the USB library and the example projects.

NOTE

The composite device examples works only with this setting:

```
USBCFG_DEV_COMPOSITE      1
```

All the other non-composite device examples work only with this setting:

```
USBCFG_DEV_COMPOSITE      0
```

If incorrect settings are configured, a build error occurs.

6.2 Host configuration

All the host configurations are listed in this file:

```
usb_core/host/include/BOARD_NAME/usb_host_config.h
```

Replace BOARD_NAME with the name of the board.

This file is used to either enable or disable the USB class driver. The object number is configurable either to decrease the memory usage or to meet specific requirements.

If the device stack configuration is changed, rebuild both the USB library and the example projects.

NOTE

Micro and mini receptacles are available for the TWR-K22F120M Tower System module if the TWR-SER and elevator are used. Configure the software and hardware to switch between the two USB receptacles.

- To use the micro receptacle on the TWR-K22F120M Tower System module, the jumper settings should be (for both device and host):
 - J4 1-2
 - J27 2-3 (for rev. A)
 - J27 1-2 (for rev. B)

If the host stack is used, the additional configuration is needed:

```
USBCFG_HOST_PORT_NATIVE    1
```

- To use the mini receptacle on the TWR-SER Tower System module, the jumper settings should be (for both device and host):
 - J4 1-2
 - J27 1-2 (for rev. A)
 - J27 2-3 (for rev. B)
 - See the appropriate TWR-SER user's guide for the jumper settings on TWR-SER Tower System module.

If the host stack is used, the additional configuration is needed:

```
USBCFG_HOST_PORT_NATIVE    0
```

Additional configurations are not needed for the device because switching between the two USB receptacles doesn't require changing code in the device mode.

6.3 OTG configuration

All OTG configurations are listed in these files:

```
usb_core/device/include/twrk22f120m/usb_device_config.h
usb_core/host/include/twrk22f120m/usb_host_config.h
```

These files either enable or disable the USB class driver. The object number is configurable either to decrease the memory usage or to meet specific requirements.

If the OTG stack configuration is changed, rebuild both USB library and example projects.

NOTE

The OTG example for the TWR-K22F120M Tower System module requires the mini receptacle on the TWR-SER Tower System module. The jumper settings should be:

- J4 1-2
- J27 1-2 (for rev. A)
- J27 2-3 (for rev. B)
- See the appropriate TWR-SER user's guide for the jumper settings on the TWR-SER Tower System module.

The additional configuration is needed for the host mode:

```
USBCFG_HOST_PORT_NATIVE    0
```

The additional configuration is needed for the device mode:

```
USBCFG_DEV_COMPOSITE       0
```

NOTE

1. If the USB mini port (J14) on the serial board needs to be used as a device connector on the K64 Tower System module, J19 must be set to 2-3.
2. Because the K64_USB_DP and K64_USB_DN are not connected to the elevator micro USB port on the TWR-K64F120M Tower System module, the R522 and the R523 are not placed and only the micro USB port can be used.
3. If the TWRK-K64 Tower System module is a USB device when no OpenSDA power is supplied, the jumpers need to be set up like this:
J29, 5-6
J19, 2-3
J18, 2-3.
4. If the Freescale Freedom FRDM-K64F is a USB device when no OpenSDA power is supplied, add a 0-ohm resistor on R61 to power on the P3V3_SDA.

7 Revision History

This table summarizes revisions to this document.

| Revision History | | |
|------------------|---------|---------------------|
| Revision number | Date | Substantial changes |
| 0 | 12/2014 | Kinetis SDK 1.1.0 |
| 1.0.0 | 7/2014 | Initial release |

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

www.freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale 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.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM, Keil, μ Vision, and ARM powered logo are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

©2014 Freescale Semiconductor, Inc.

