Freescale Semiconductor, Inc Document Number: USBKSDKUG
User's Guide Rev. 1, 04/2015

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

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

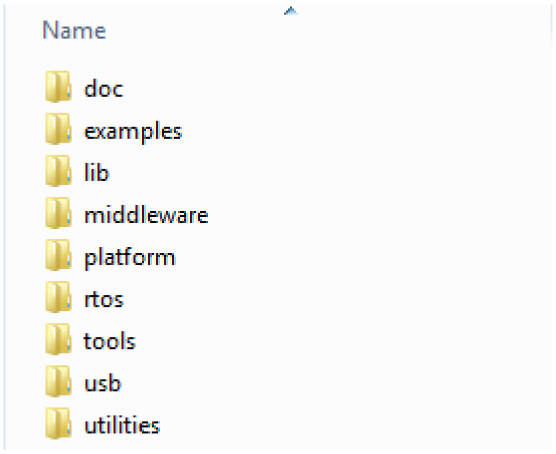
2.2 Software

- KSDK release package
- IAR Embedded Workbench for ARM® Version 7.40.2
- Keil µVision5 Integrated Development Environment Version 5.14, available for Kinetis ARM Cortex®-M4 devices
- Kinetis Design Studio IDE v3.0
- Atollic® TrueSTUDIO® v5.3
- Makefiles support with GCC revision 4.8-2014-q3-update from ARM Embedded

3 USB Code Structure

The USB code is located in the folder:

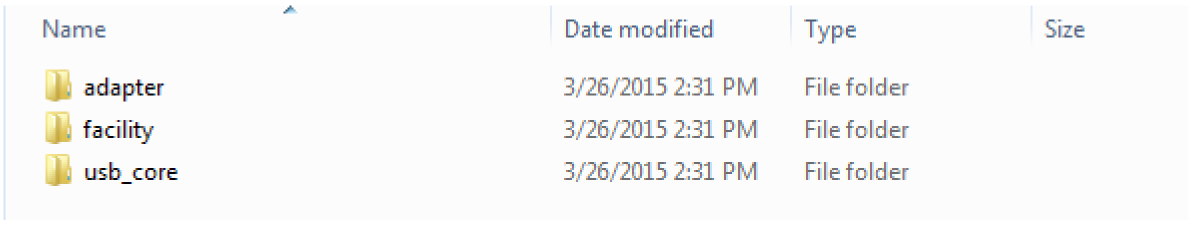
<install_dir>/usb



Name
doc
examples
lib
middleware
platform
rtos
tools
usb
utilities

Figure-1 Kinetis SDK folder structure

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



Name	Date modified	Type	Size
adapter	3/26/2015 2:31 PM	File folder	
facility	3/26/2015 2:31 PM	File folder	
usb_core	3/26/2015 2:31 PM	File folder	

Figure-2 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.

- usb_core

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

- facility

  This subfolder includes the interfaces to accommodate various peripherals to which the USB refers to by, such as UART.

4 Compiling or Running the USB Stack and Examples

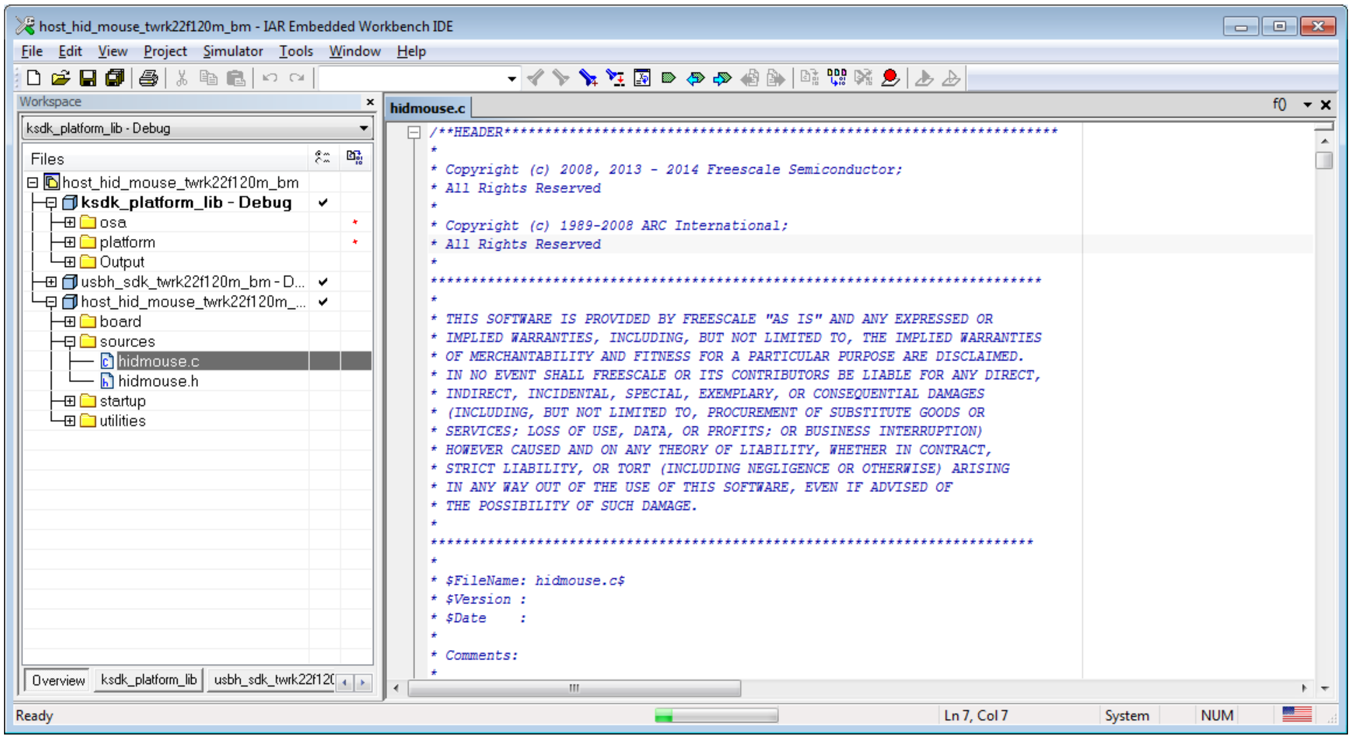
4.1 Step-by-step guide for IAR

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 workspace file is located as

   <install_dir>/examples/twrk22f120m/demo_apps/usb/host/hid/mouse/bm/iar/
host_hid_mouse_twrk22f120m_bm.eww .


host_hid_mouse_twrk22f120m_bm - IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
ksdk_platform_lib - Debug
Files
host_hid_mouse_twrk22f120m_bm
ksdk_platform_lib - Debug
osa
platform
Output
usbh_sdk_twrk22f120m_bm - D...
host_hid_mouse_twrk22f120m_...
board
sources
hidmouse.c
hidmouse.h
startup
utilities
Overview
ksdk_platform_lib
usbh_sdk_twrk22f120
hidmouse.c
f()
Ready
Ln 7, Col 7
System
NUM

```
/**HEADER*********************************************************************
*
* Copyright (c) 2008, 2013 - 2014 Freescale Semiconductor;
* All Rights Reserved
*
* Copyright (c) 1989-2008 ARC International;
* All Rights Reserved
*
****************************************************************************
*
* THIS SOFTWARE IS PROVIDED BY FREESCALE "AS IS" AND ANY EXPRESSED OR
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
* IN NO EVENT SHALL FREESCALE OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
* THE POSSIBILITY OF SUCH DAMAGE.
*
***************************************************************************
*
* $FileName: hidmouse.c$
* $Version :
* $Date    :
*
* Comments:
*
```

Figure-3 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 is located in <install_dir>/usb/usb_core/host/build/iar/usbh_sdk_twrk22f120m_bm/debug/libusbh_bm.a .
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.
11. See the example-specific readme.pdf for more test information.

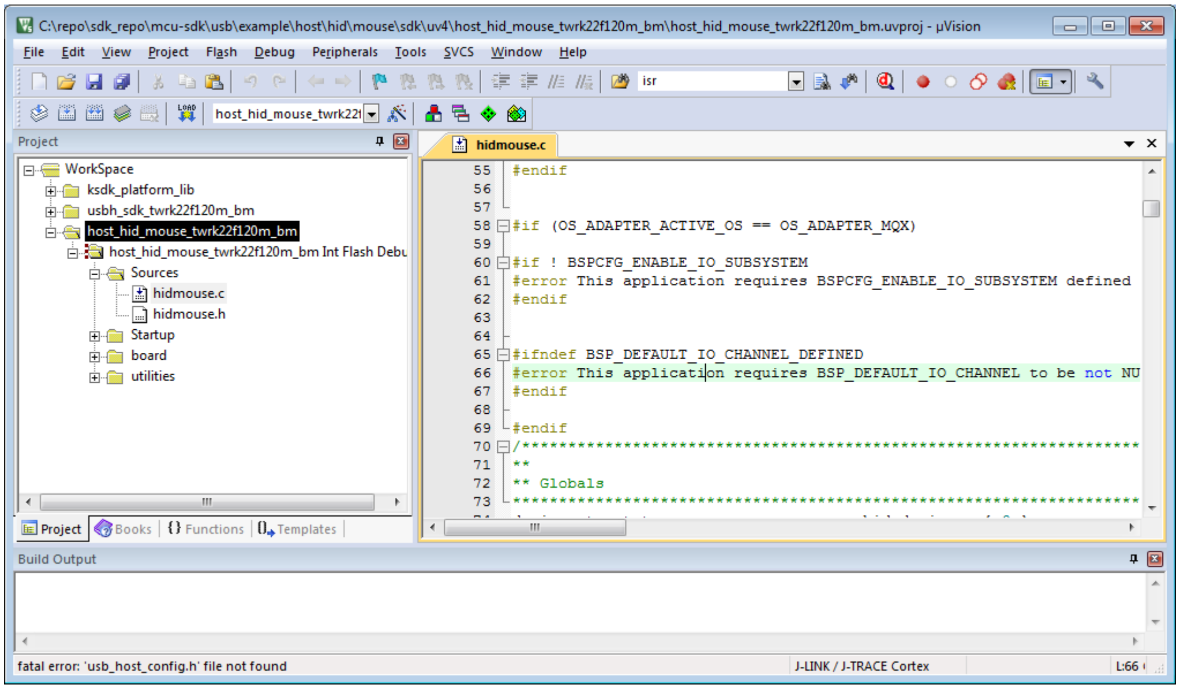
4.2 Step-by-step guide for Keil µVision5

This section shows how to use Keil µVision5. Open Keil µVision5 as shown in this figure:

1. Open the workspace corresponding to different examples.

   For example, the workspace file is located in

   <install_dir>/examples/twrk22f120m/demo_apps/usb/host/hid/mouse/bm/mdk/host_hid_mouse_twrk22f120m_bm.uvmpw


C:\repo\sdk_repo\mcu-sdk\usb\example\host\hid\mouse\sdk\uv4\host_hid_mouse_twrk22f120m_bm\host_hid_mouse_twrk22f120m_bm.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
isr
host_hid_mouse_twrk22f
Project
WorkSpace
ksdk_platform_lib
usbh_sdk_twrk22f120m_bm
host_hid_mouse_twrk22f120m_bm
host_hid_mouse_twrk22f120m_bm Int Flash Debu
Sources
hidmouse.c
hidmouse.h
Startup
board
utilities
Project
Books
Functions
Templates
hidmouse.c
Build Output
J-LINK / J-TRACE Cortex
L:66

```
#endif


#if (OS_ADAPTER_ACTIVE_OS == OS_ADAPTER_MQX)

#if ! BSPCFG_ENABLE_IO_SUBSYSTEM
#error This application requires BSPCFG_ENABLE_IO_SUBSYSTEM defined
#endif


#ifndef BSP_DEFAULT_IO_CHANNEL_DEFINED
#error This application requires BSP_DEFAULT_IO_CHANNEL to be not NU
#endif

#endif
/*******************************************************************
**
** Globals
********************************************************************
```

```
fatal error: 'usb_host_config.h' file not found
```

Figure-4 Keil µVision5 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.
7. See the example-specific readme.pdf for more test information.

4.3 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.

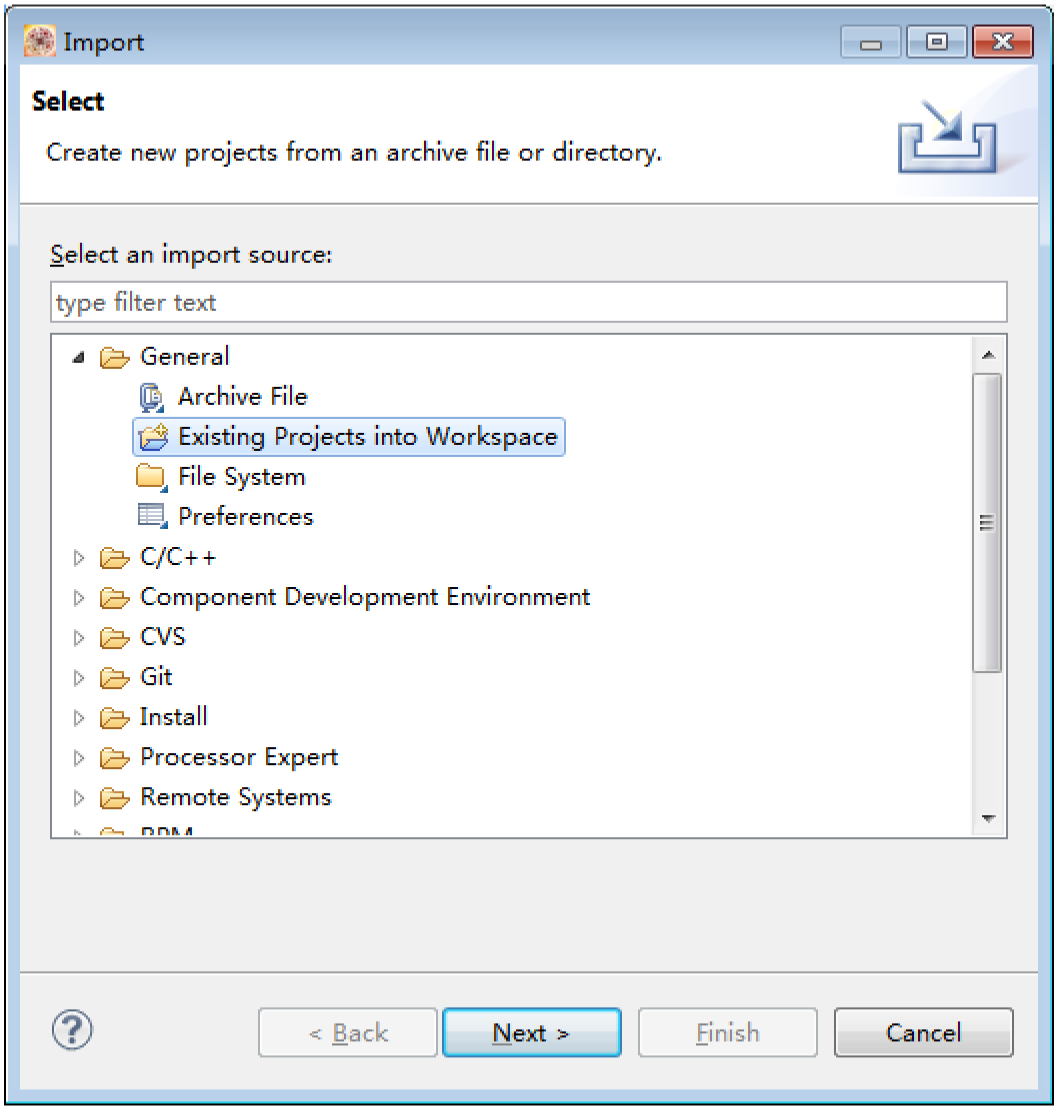
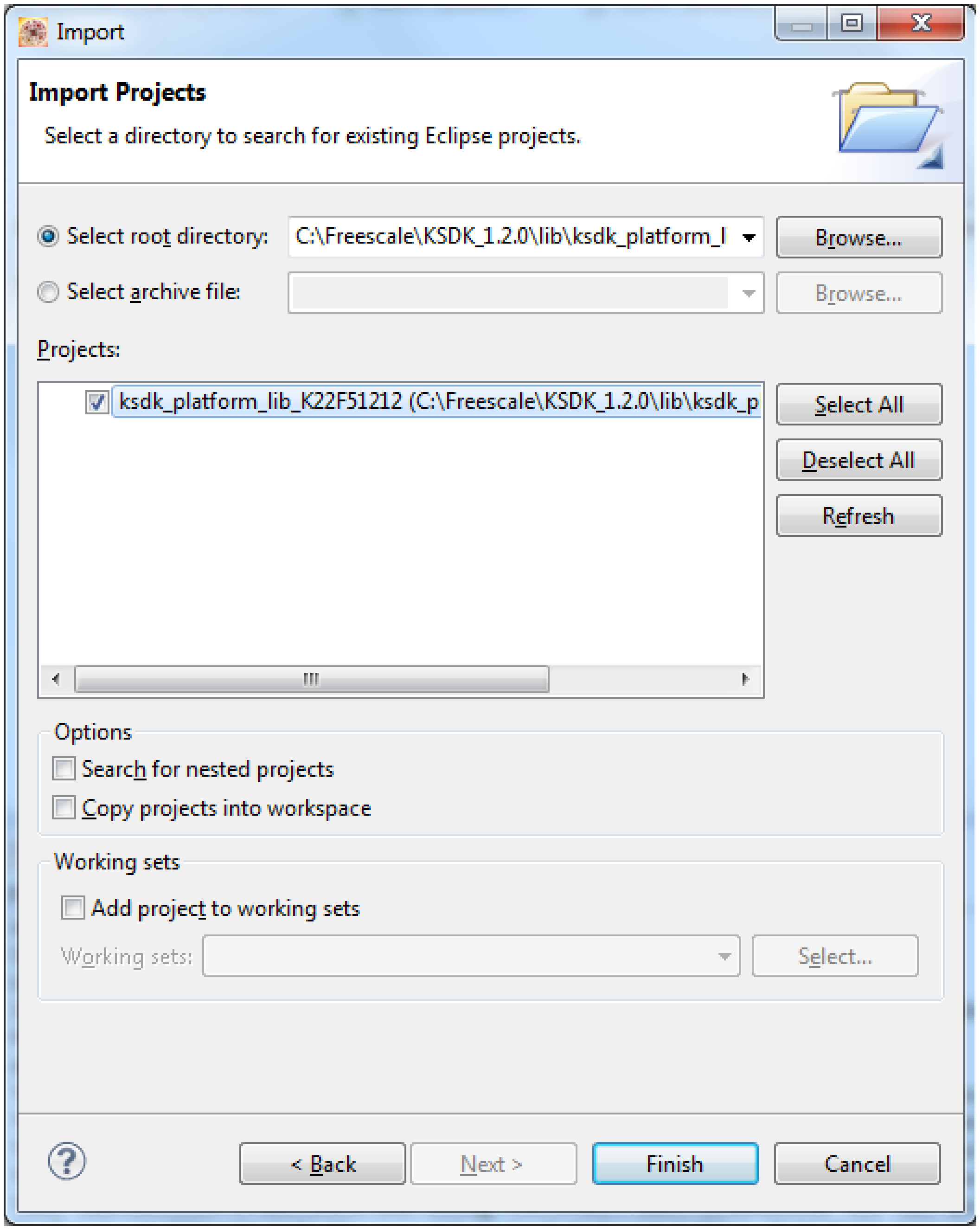


Figure-5 Selection of the correct import type in KDS IDE

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Freescale\KSDK_1.2.0\lib\ksdk_platform_l

( ) Select archive file:

Projects:

[✓] ksdk_platform_lib_K22F51212 (C:\Freescale\KSDK_1.2.0\lib\ksdk_p

Options

[ ] Search for nested projects

[ ] Copy projects into workspace

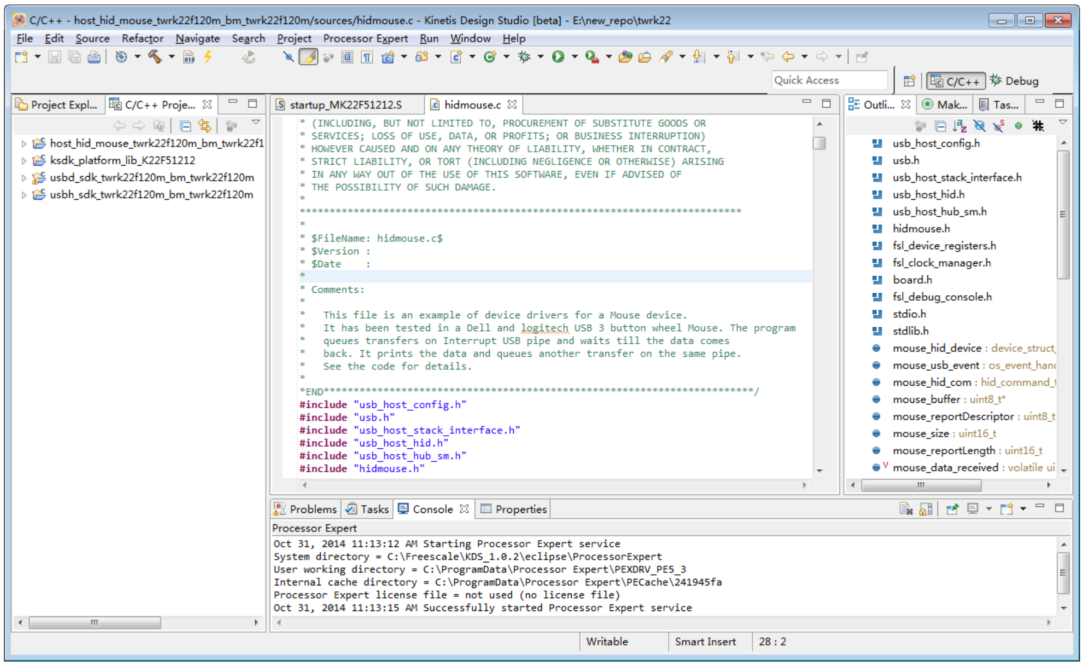
Working sets

[ ] Add project to working sets

Working sets:

Figure-6 Selection of the K22 ksdk_platform_lib project

5. Follow the same steps to import the USB device/host library and the USB example. After importing, the window should like this.


C/C++ - host_hid_mouse_twrk22f120m_bm_twrk22f120m/sources/hidmouse.c - Kinetis Design Studio [beta] - E:\new_repo\twrk22
File Edit Source Refactor Navigate Search Project Processor Expert Run Window Help
Quick Access
C/C++
Debug
Project Expl...
C/C++ Proje...
host_hid_mouse_twrk22f120m_bm_twrk22f1
ksdk_platform_lib_K22F51212
usbd_sdk_twrk22f120m_bm_twrk22f120m
usbh_sdk_twrk22f120m_bm_twrk22f120m
startup_MK22F51212.S
hidmouse.c
Outli...
Mak...
Tas...
usb_host_config.h
usb.h
usb_host_stack_interface.h
usb_host_hid.h
usb_host_hub_sm.h
hidmouse.h
fsl_device_registers.h
fsl_clock_manager.h
board.h
fsl_debug_console.h
stdio.h
stdlib.h
mouse_hid_device : device_struct
mouse_usb_event : os_event_hanc
mouse_hid_com : hid_command_t
mouse_buffer : uint8_t*
mouse_reportDescriptor : uint8_t
mouse_size : uint16_t
mouse_reportLength : uint16_t
mouse_data_received : volatile ui
Problems
Tasks
Console
Properties
Writable
Smart Insert
28 : 2

```
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
* THE POSSIBILITY OF SUCH DAMAGE.
*
**************************************************************************
*
* $FileName: hidmouse.c$
* $Version :
* $Date    :
*
* Comments:
*
*   This file is an example of device drivers for a Mouse device.
*   It has been tested in a Dell and logitech USB 3 button wheel Mouse. The program
*   queues transfers on Interrupt USB pipe and waits till the data comes
*   back. It prints the data and queues another transfer on the same pipe.
*   See the code for details.
*
*END**********************************************************************/
#include "usb_host_config.h"
#include "usb.h"
#include "usb_host_stack_interface.h"
#include "usb_host_hid.h"
#include "usb_host_hub_sm.h"
#include "hidmouse.h"
```

Processor Expert

```
Oct 31, 2014 11:13:12 AM Starting Processor Expert service
System directory = C:\Freescale\KDS_1.0.2\eclipse\ProcessorExpert
User working directory = C:\ProgramData\Processor Expert\PEXDRV_PE5_3
Internal cache directory = C:\ProgramData\Processor Expert\PECache\241945fa
Processor Expert license file = not used (no license file)
Oct 31, 2014 11:13:15 AM Successfully started Processor Expert service
```

Figure-7 The USB projects workspace

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

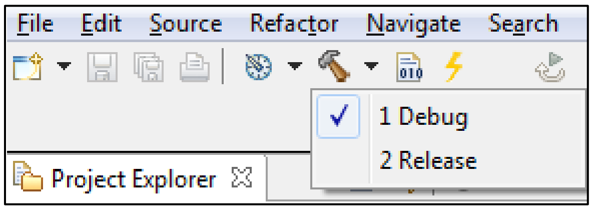


Figure-8 The hammer button

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

8. Follow the same steps 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”.

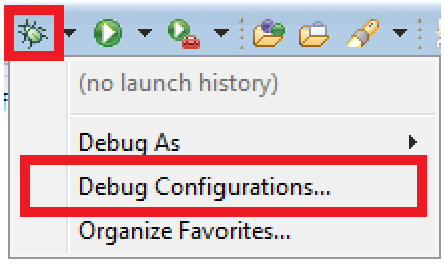
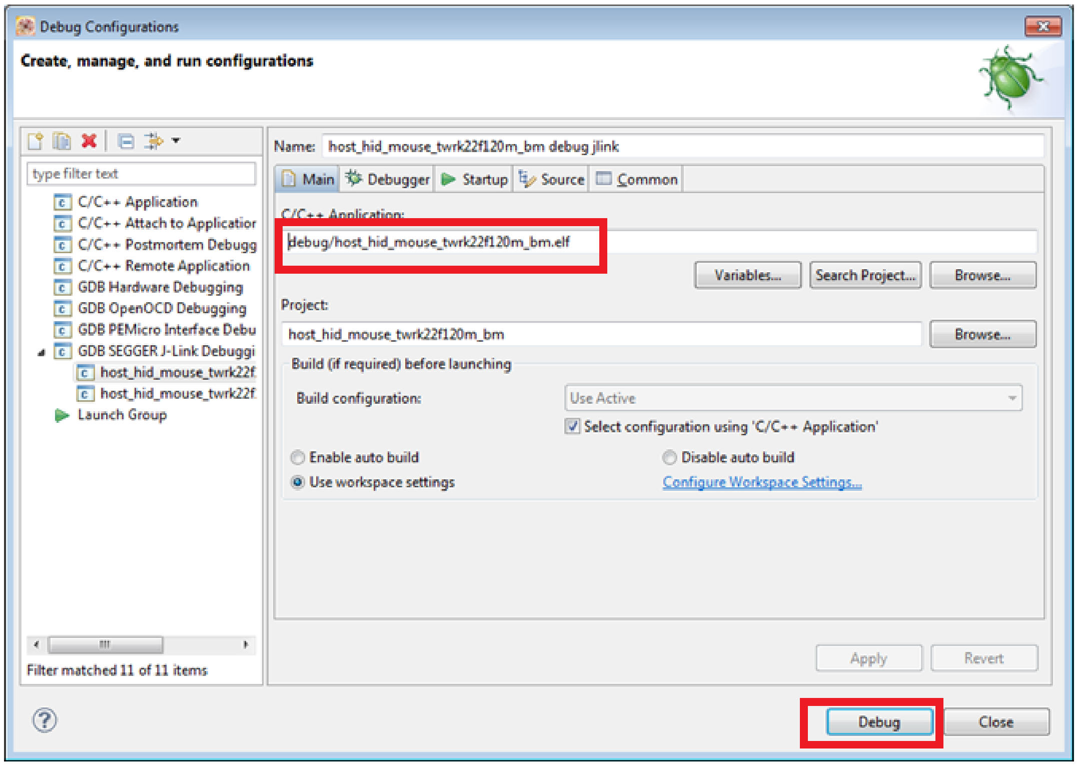


Figure-9 Debug configurations

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



Name: host_hid_mouse_twrk22f120m_bm debug jlink

C/C++ Application: debug/host_hid_mouse_twrk22f120m_bm.elf

Project: host_hid_mouse_twrk22f120m_bm

Build (if required) before launching

Build configuration: Use Active

[✓] Select configuration using 'C/C++ Application'

( ) Enable auto build ( ) Disable auto build

(●) Use workspace settings Configure Workspace Settings...

Figure-10 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:

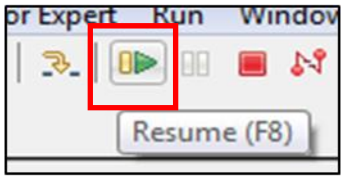


Figure-11 Resume button

13. See the example-specific readme.pdf for more test information.

4.4 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.

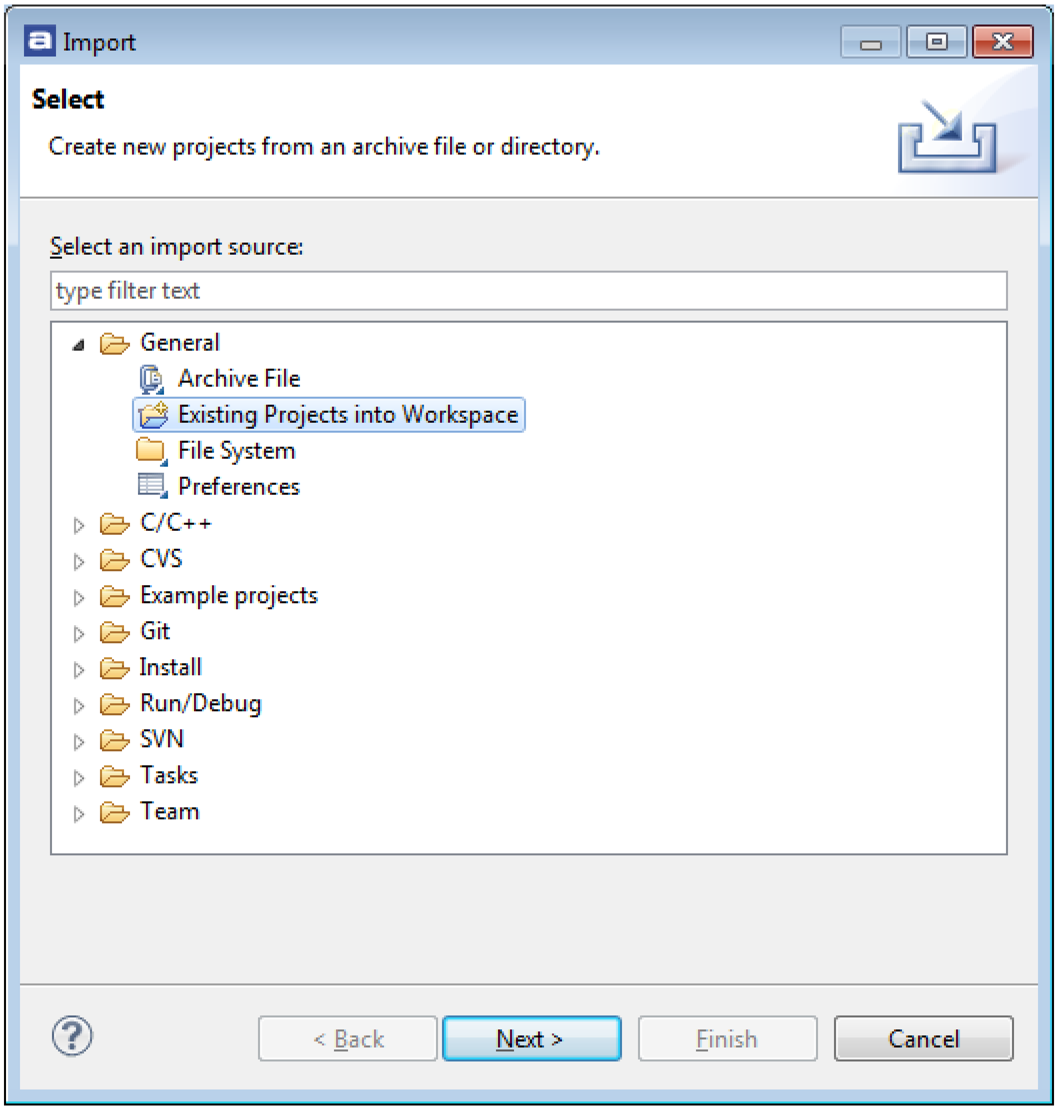
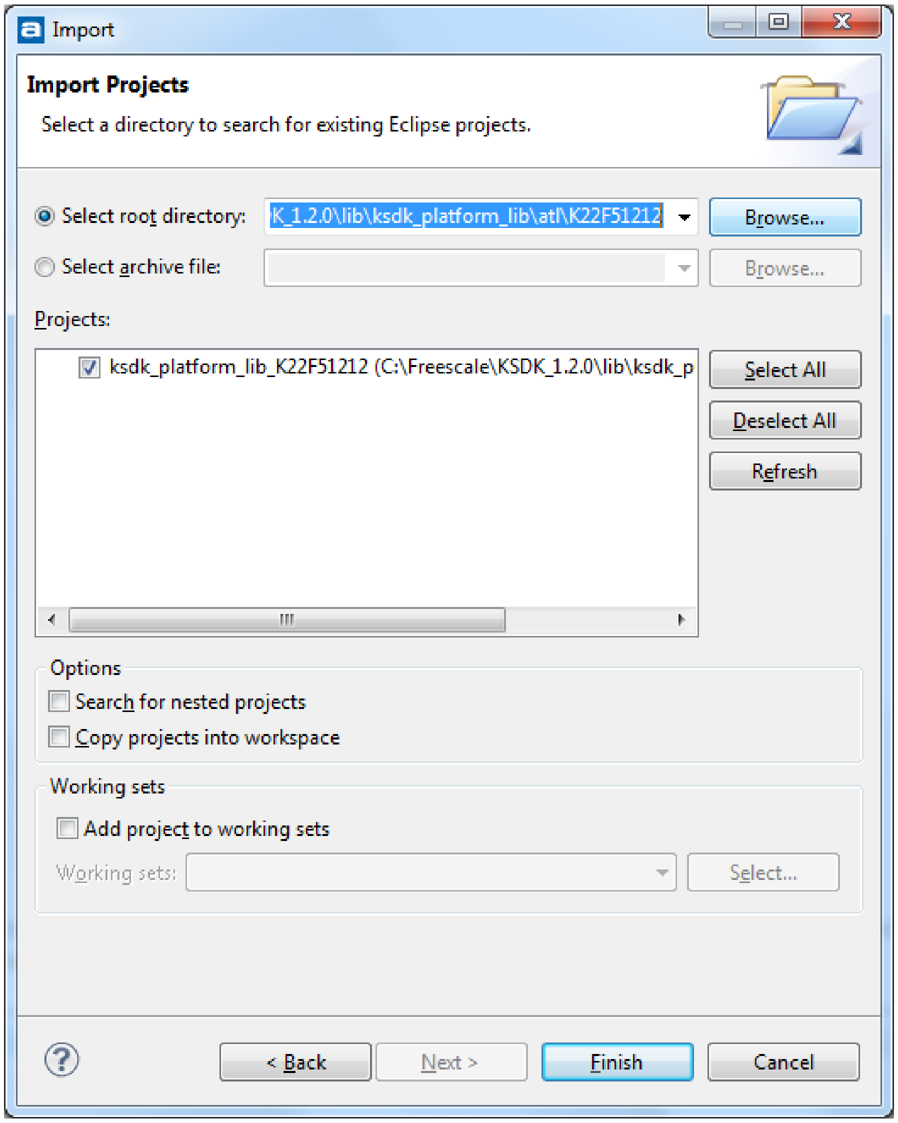


Figure-12 Selection of the correct import type in Atollic TrueSTUDIO IDE

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.


Import
Import Projects
Select a directory to search for existing Eclipse projects.

Select root directory: K_1.2.0\lib\ksdk_platform_lib\atl\K22F51212

Select archive file:

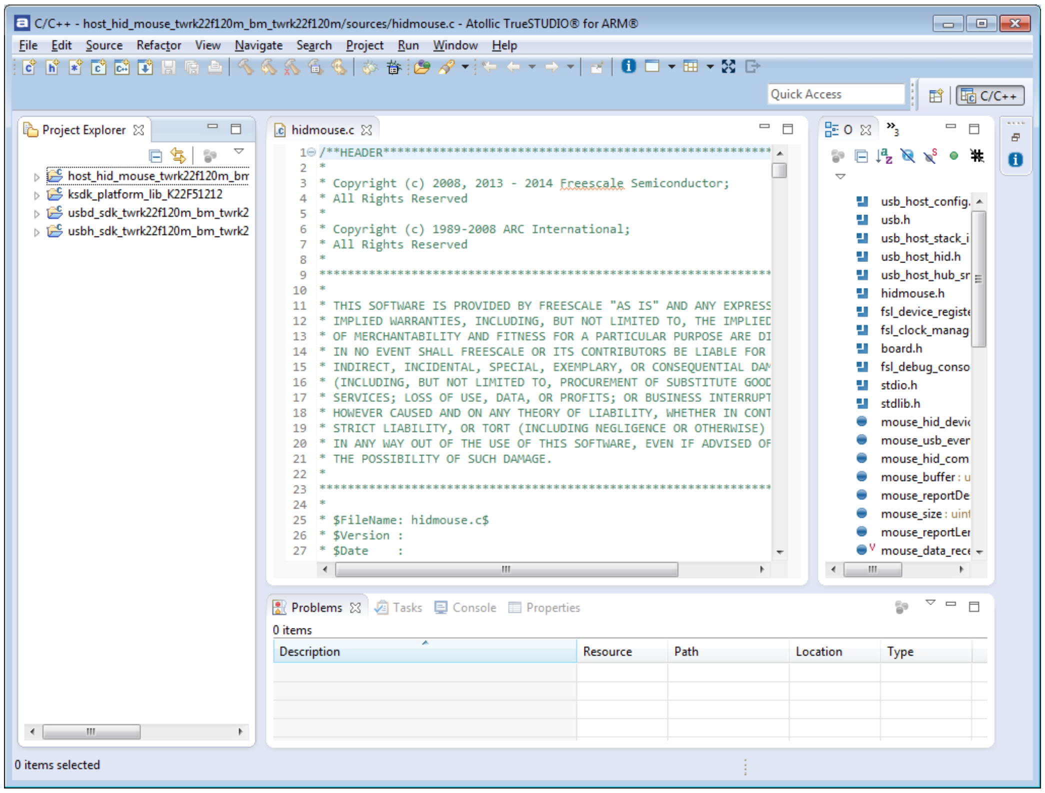
Projects:

ksdk_platform_lib_K22F51212 (C:\Freescale\KSDK_1.2.0\lib\ksdk_p

Browse...
Browse...
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

Figure-13 Selection of the K22 ksdk_platform_lib project

5. Follow the same steps to import the USB device/host library and the USB example. After importing, the window should like this.



C/C++ - host_hid_mouse_twrk22f120m_bm_twrk22f120m/sources/hidmouse.c - Atollic TrueSTUDIO® for ARM®

Project Explorer

- host_hid_mouse_twrk22f120m_bm
- ksdk_platform_lib_K22F51212
- usbd_sdk_twrk22f120m_bm_twrk2
- usbh_sdk_twrk22f120m_bm_twrk2

hidmouse.c

```
/**HEADER***********************************************************
*
* Copyright (c) 2008, 2013 - 2014 Freescale Semiconductor;
* All Rights Reserved
*
* Copyright (c) 1989-2008 ARC International;
* All Rights Reserved
*
*******************************************************************
*
* THIS SOFTWARE IS PROVIDED BY FREESCALE "AS IS" AND ANY EXPRESS
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DI
* IN NO EVENT SHALL FREESCALE OR ITS CONTRIBUTORS BE LIABLE FOR
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAM
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOOD
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPT
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONT
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
* THE POSSIBILITY OF SUCH DAMAGE.
*
*******************************************************************
*
* $FileName: hidmouse.c$
* $Version :
* $Date    :
```

Outline:

- usb_host_config.
- usb.h
- usb_host_stack_i
- usb_host_hid.h
- usb_host_hub_sr
- hidmouse.h
- fsl_device_registe
- fsl_clock_manag
- board.h
- fsl_debug_conso
- stdio.h
- stdlib.h
- mouse_hid_devic
- mouse_usb_ever
- mouse_hid_com
- mouse_buffer : u
- mouse_reportDe
- mouse_size : uint
- mouse_reportLer
- mouse_data_rece

Problems | Tasks | Console | Properties

0 items

Description	Resource	Path	Location	Type

0 items selected

Figure-14 The USB projects workspace

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

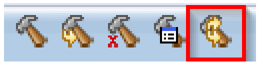
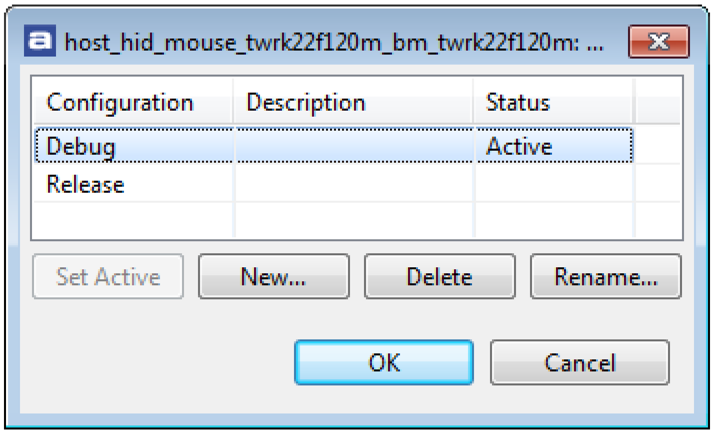


Figure-15 Manage build configuration button



host_hid_mouse_twrk22f120m_bm_twrk22f120m: ...

Configuration	Description	Status
Debug		Active
Release		

Figure-16 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.

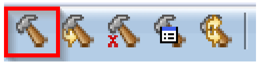


Figure-17 Build project button

8. Follow the same steps 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.

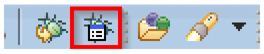
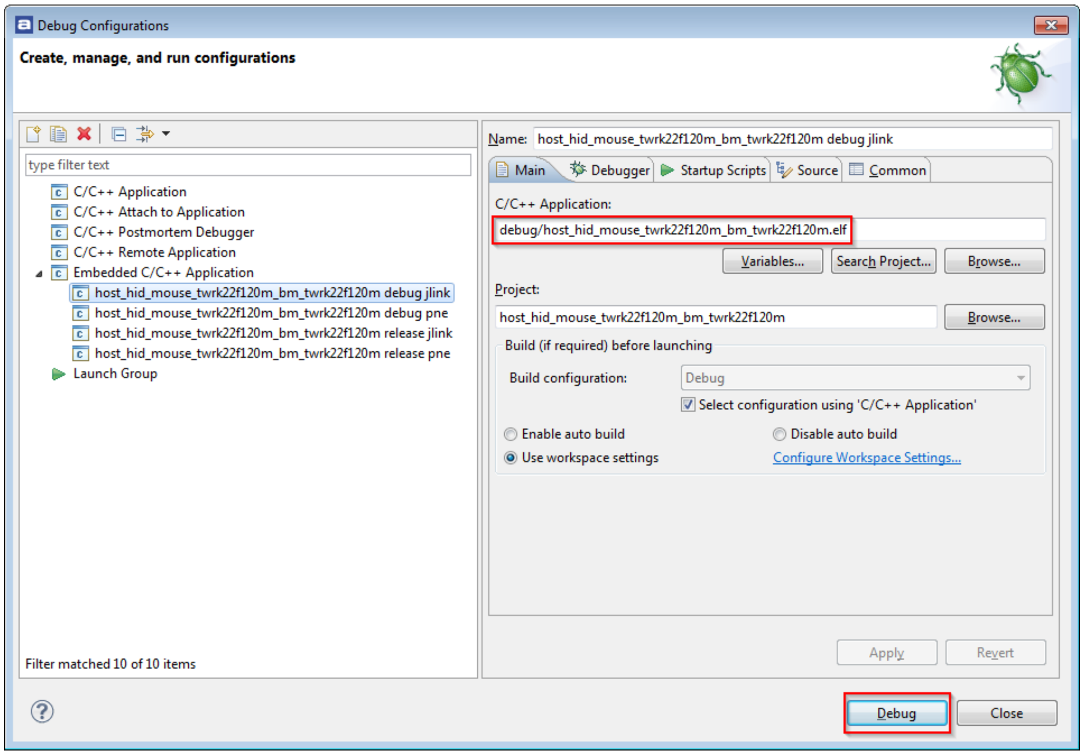


Figure-18 Configure debug button

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


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- Embedded C/C++ Application
  - host_hid_mouse_twrk22f120m_bm_twrk22f120m debug jlink
  - host_hid_mouse_twrk22f120m_bm_twrk22f120m debug pne
  - host_hid_mouse_twrk22f120m_bm_twrk22f120m release jlink
  - host_hid_mouse_twrk22f120m_bm_twrk22f120m release pne
- Launch Group

Filter matched 10 of 10 items

Name: host_hid_mouse_twrk22f120m_bm_twrk22f120m debug jlink

Main | Debugger | Startup Scripts | Source | Common

C/C++ Application:
debug/host_hid_mouse_twrk22f120m_bm_twrk22f120m.elf

Variables... Search Project... Browse...

Project:
host_hid_mouse_twrk22f120m_bm_twrk22f120m

Browse...

Build (if required) before launching

Build configuration: Debug

[✓] Select configuration using 'C/C++ Application'

Enable auto build | Disable auto build
[✓ Use workspace settings] | Configure Workspace Settings...

Apply | Revert

Debug | Close

Figure-19 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:

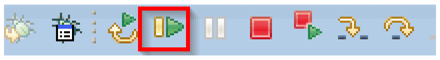


Figure-20 The resume button

13. See the example-specific readme.pdf for more test information.

4.5 Step-by-step guide for the ARM GCC

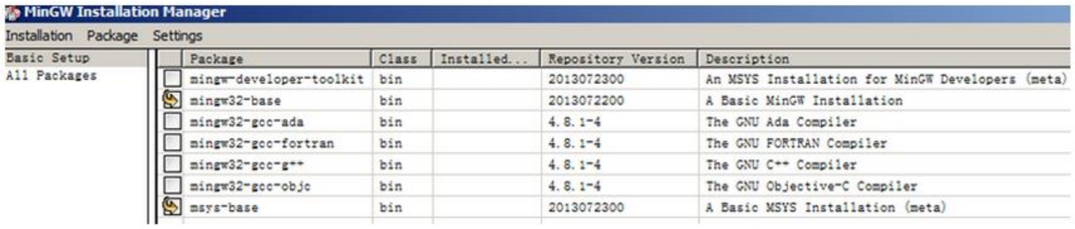
4.5.1 Setup tool chains

4.5.1.1 Install GCC ARM Embedded tool chain

Download and install the installer from [www.launchpad.net/gcc-arm-embedded.](www.launchpad.net/gcc-arm-embedded)

4.5.1.2 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”.



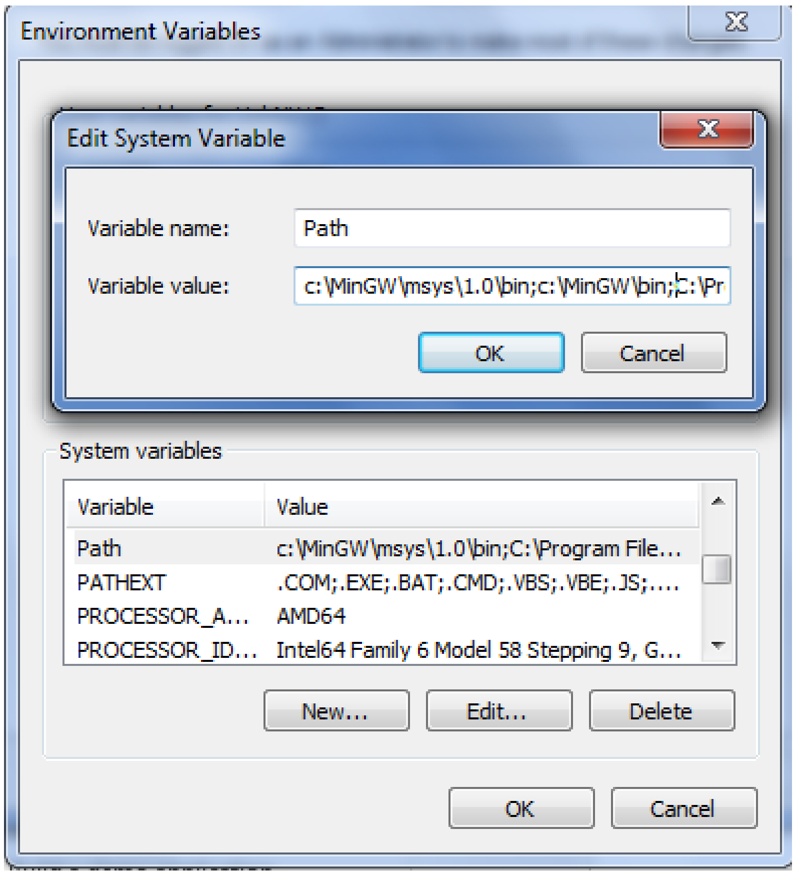
	Package	Class	Installed...	Repository Version	Description
☐	mingw-developer-toolkit	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
✓	mingw32-base	bin		2013072200	A Basic MinGW Installation
☐	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
☐	mingw32-gcc-fortran	bin		4.8.1-4	The GNU FORTRAN Compiler
☐	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
☐	mingw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
✓	msys-base	bin		2013072300	A Basic MSYS Installation (meta)

Figure 21: 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.



Variable name: Path

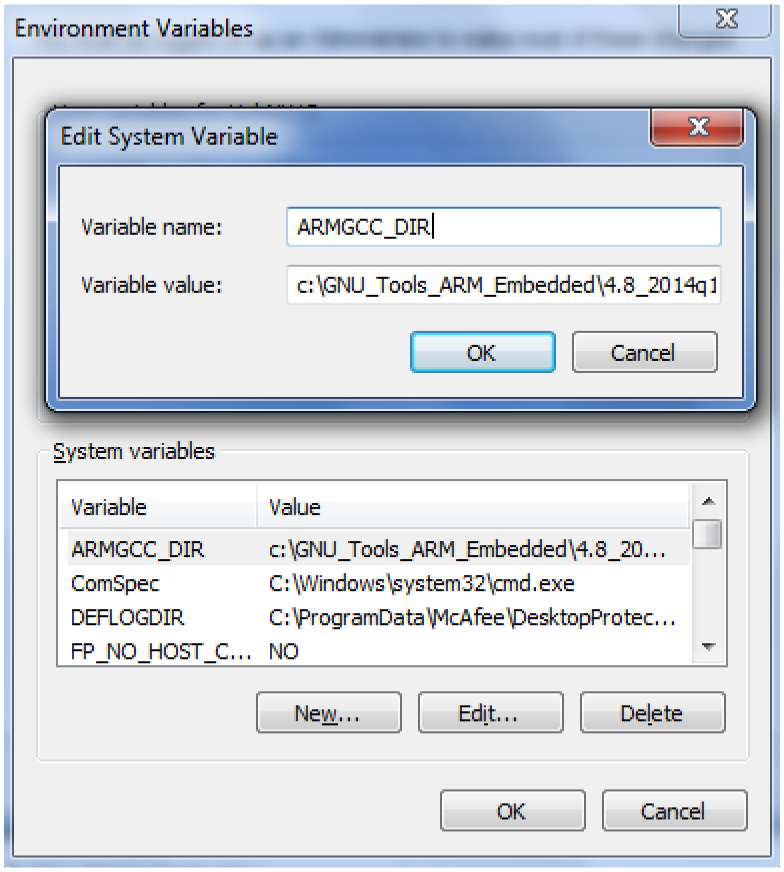
Variable value: c:\MinGW\msys\1.0\bin;c:\MinGW\bin;C:\Pr

Variable	Value
Path	c:\MinGW\msys\1.0\bin;C:\Program File...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64
PROCESSOR_ID...	Intel64 Family 6 Model 58 Stepping 9, G...

Figure 22: Add Path to systems environment

4.5.1.3 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.



Variable name: ARMGCC_DIR

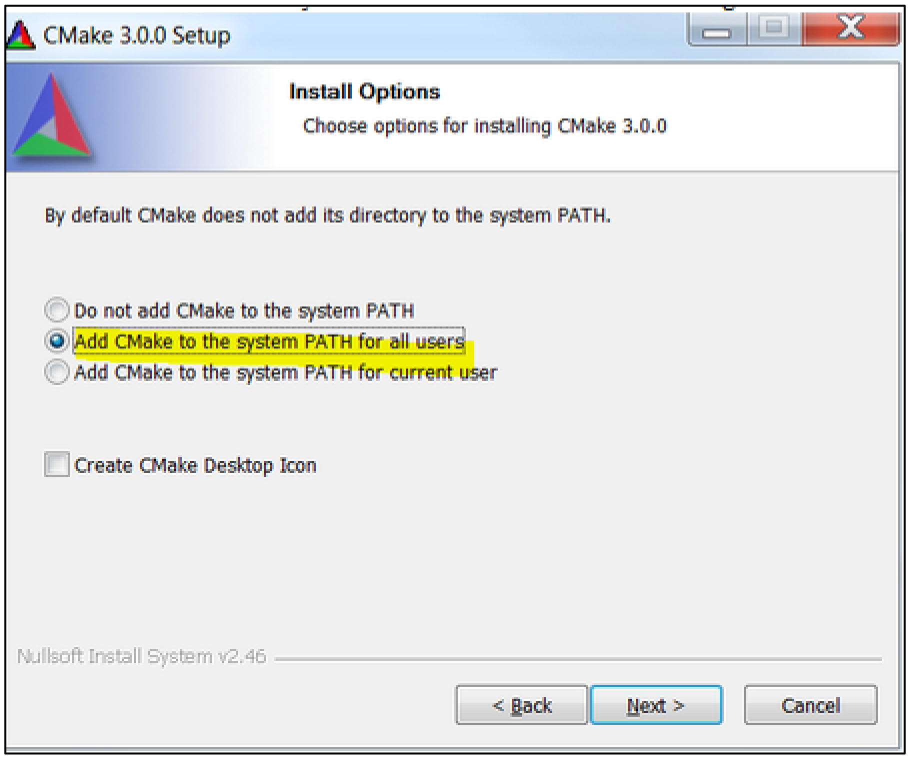
Variable value: c:\GNU_Tools_ARM_Embedded\4.8_2014q1

Variable	Value
ARMGCC_DIR	c:\GNU_Tools_ARM_Embedded\4.8_20...
ComSpec	C:\Windows\system32\cmd.exe
DEFLOGDIR	C:\ProgramData\McAfee\DesktopProtec...
FP_NO_HOST_C...	NO

Figure 23: Add ARMGCC_DIR system variable

4.5.1.4 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).


CMake 3.0.0 Setup

Install Options

Choose options for installing CMake 3.0.0

By default CMake does not add its directory to the system PATH.

○ Do not add CMake to the system PATH
● Add CMake to the system PATH for all users
○ Add CMake to the system PATH for current user

☐ Create CMake Desktop Icon

Nullsoft Install System v2.46

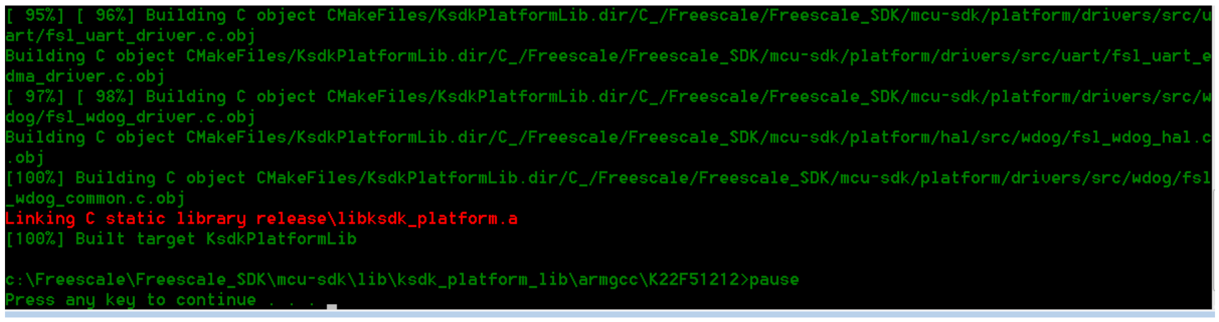
Figure 24: Install CMake

4.5.2 Build the platform driver library

To build the platform library, follow these instructions:

1. Open a GCC ARM Embedded tool chain command window.
2. Change the directory of the command window to the platform lib directory in the KSDK:

   <install_dir>/lib/ksdk_platform_lib/armgcc/<device_name>/
3. Type “build_all”.



```
[ 95%] [ 96%] Building C object CMakeFiles/KsdkPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/u
art/fsl_uart_driver.c.obj
Building C object CMakeFiles/KsdkPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/uart/fsl_uart_e
dma_driver.c.obj
[ 97%] [ 98%] Building C object CMakeFiles/KsdkPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/platform/drivers/src/w
dog/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 25: ksdk_platform_lib build successfully

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

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

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

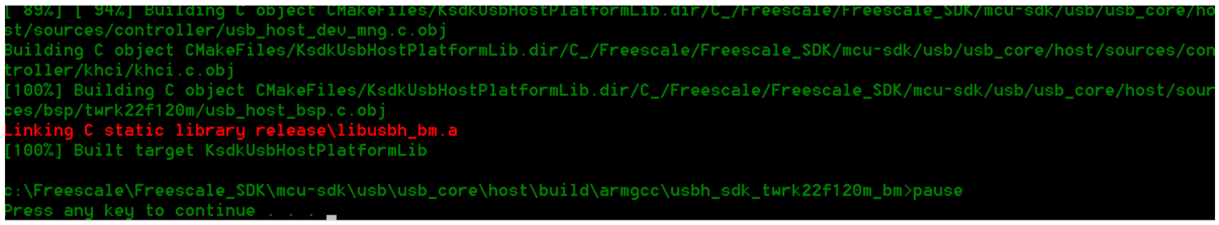
4.5.3 Build the USB host/device library

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/host/build/armgcc/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:

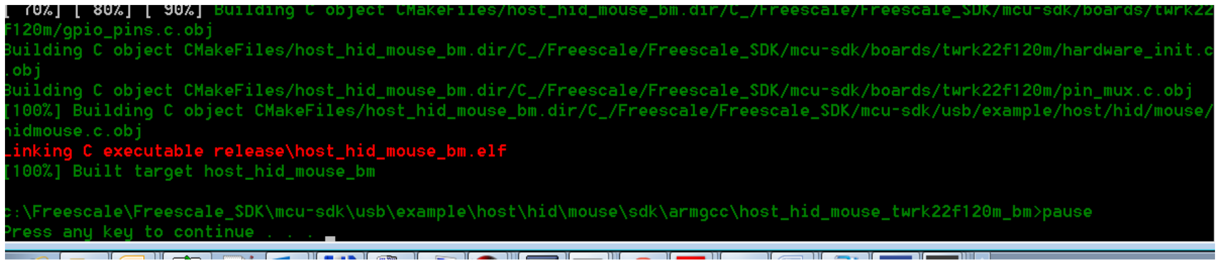


```
[ 89%] [ 94%] Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/ho
st/sources/controller/usb_host_dev_mng.c.obj
Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/host/sources/con
troller/khci/khci.c.obj
[100%] Building C object CMakeFiles/KsdkUsbHostPlatformLib.dir/C_/Freescale/Freescale_SDK/mcu-sdk/usb/usb_core/host/sour
ces/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 26: USB host library build successfully

4.5.4 Build the USB demo

1. Change the directory to the project directory:
2. *<install_dir>/examples/twrk22f120m/demo_apps/usb/host/hid/mouse/bm/armgcc* 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:



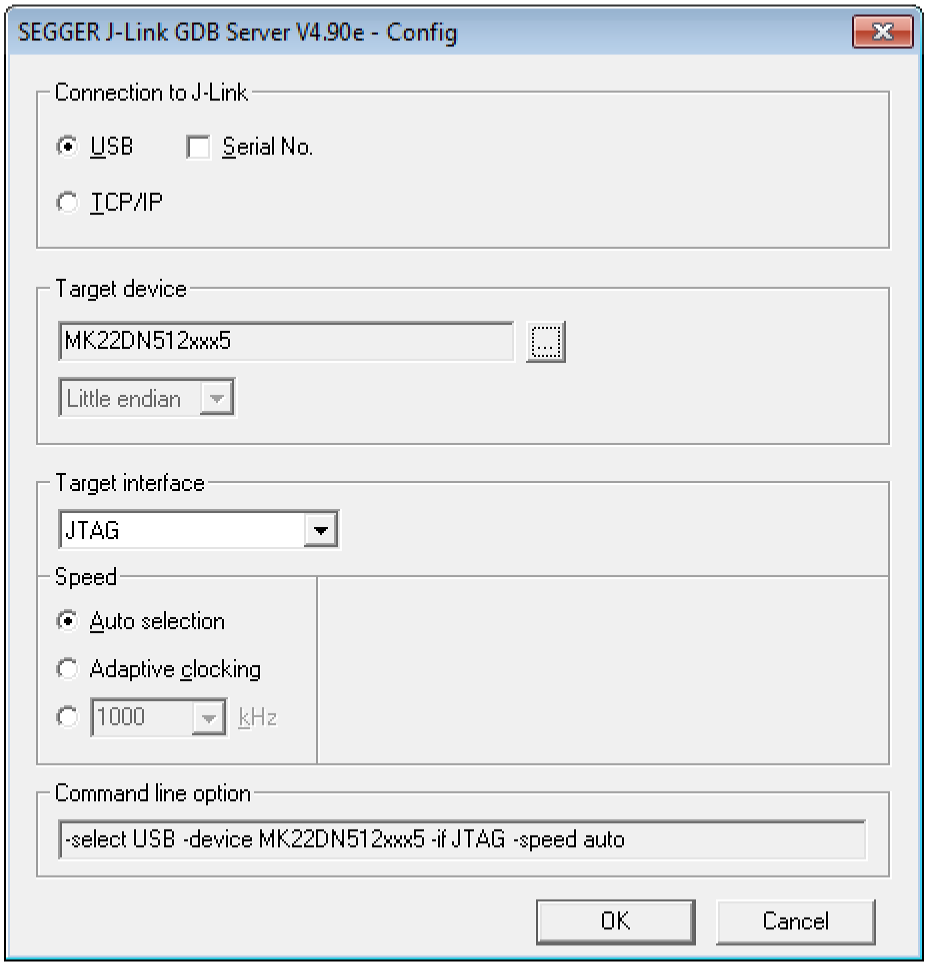
```
[ 70%] [ 80%] [ 90%] Building C object CMakeFiles/host_hid_mouse_bm.dir/C_/Freescale/Freescale_SDK/mcu-sdk/boards/twrk22
F120m/gpio_pins.c.obj
Building C object CMakeFiles/host_hid_mouse_bm.dir/C_/Freescale/Freescale_SDK/mcu-sdk/boards/twrk22f120m/hardware_init.c
.obj
Building C object CMakeFiles/host_hid_mouse_bm.dir/C_/Freescale/Freescale_SDK/mcu-sdk/boards/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 27: USB host demo build successfully

4.5.5 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 port 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.



SEGGER J-Link GDB Server V4.90e - Config

Connection to J-Link: [✓ USB] Serial No. TCP/IP

Target device: MK22DN512xxx5

Little endian

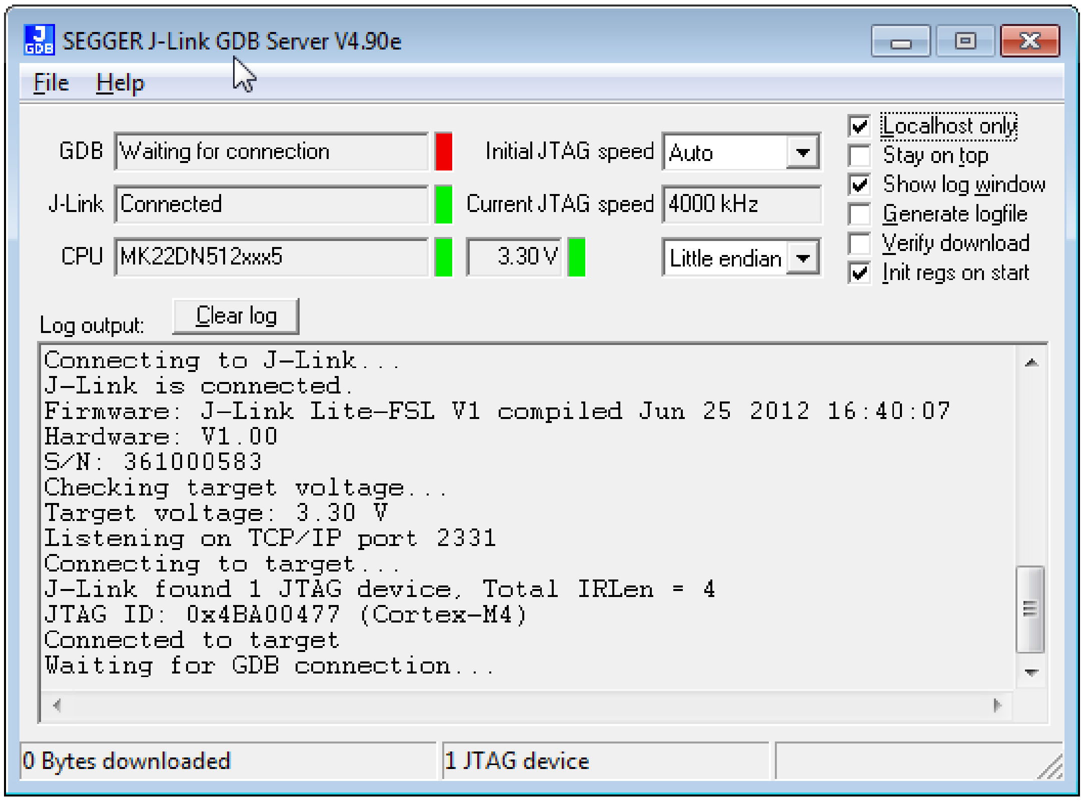
Target interface: JTAG

Speed: [✓ Auto selection] Adaptive clocking 1000 kHz

Command line option: -select USB -device MK22DN512xxx5 -if JTAG -speed auto

Figure 28: SEGGER J-Link GDB Server configuration

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



GDB: Waiting for connection 🟥
J-Link: Connected 🟩
CPU: MK22DN512xxx5 🟩
3.30 V 🟩
Initial JTAG speed: Auto
Current JTAG speed: 4000 kHz
Little endian
☑ Localhost only
☐ Stay on top
☑ Show log window
☐ Generate logfile
☐ Verify download
☑ Init regs on start

Log output:

```
Connecting to J-Link...
J-Link is connected.
Firmware: J-Link Lite-FSL V1 compiled Jun 25 2012 16:40:07
Hardware: V1.00
S/N: 361000583
Checking target voltage...
Target voltage: 3.30 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 4
JTAG ID: 0x4BA00477 (Cortex-M4)
Connected to target
Waiting for GDB connection...
```

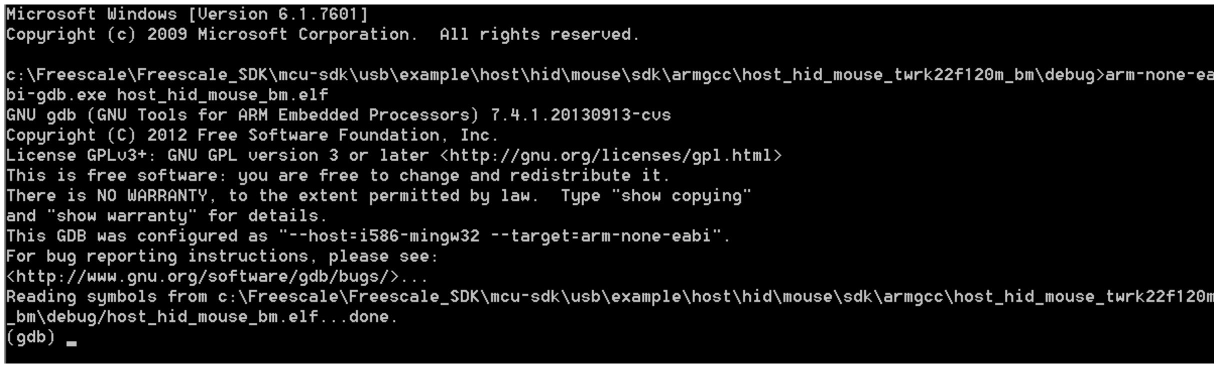
0 Bytes downloaded | 1 JTAG device

Figure 29: 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>/examples/twrk22f120m/demo_apps/usb/host/hid/mouse/bm/armgcc/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-ea
bi-gdb.exe host_hid_mouse_bm.elf
GNU gdb (GNU Tools for ARM Embedded Processors) 7.4.1.20130913-cvs
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 30: 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.
8. See the example-specific readme.pdf for more test information.

5 USB Stack Configuration

5.1 Device configuration

All device configurations are listed in this file:

```
<install_dir>/usb/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 device 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.

5.2 Host configuration

All the host configurations are listed in this file:

```
<install_dir>/usb/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 Host stack configuration is changed, rebuild both the USB Host 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 1-2

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
  - 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.

5.3 OTG configuration

All OTG configurations are listed in these files:

```
usb_core/device/include/<BOARD_NAME>/usb_device_config.h
usb_core/host/include/<BOARD_NAME>/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 OTG 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
- 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 TWR-SER 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.

6 Revision History

This table summarizes revisions to this document.

Table 1 Revision History

Revision number	Date	Substantial changes
1	04/2015	Kinetis SDK 1.2.0 release
0	12/2014	Kinetis SDK 1.1.0 release

Document Number: USBKSDKUG
Rev. 1.
04/2015