User's Guide

# Kinetis SDK K22 User's Guide

## **1** Introduction

This document describes the hardware and software environment setup for the Kinetis SDK (KSDK) for first time Kinetis SDK users. It also explains how to build and run demo applications provided in the KSDK release package.

## 2 Overview

### 2.1 Kinetis SDK

Kinetis SDK (KSDK) is a Software Development Kit that provides comprehensive software support for the core and peripherals of all Freescale devices with Cortex®-M core. The KSDK includes a Hardware Abstraction Layer (HAL) for each peripheral and peripheral drivers built on the HAL. Example applications are provided to demonstrate driver and HAL usage to highlight the main features of targeted SoCs. It also contains the latest available RTOS kernels. Additionally, KSDK integrates the TCP/IP stack and File system. Therefore, it is necessary to add these integrated software solutions.

#### Contents

1	Introduction1
2	Overview1
3	Hardware Configurations2
4 IAR	Build and Run the KSDK Demo Applications using
5 ARM	Build and Run the KSDK Demo Applications using GCC
6 Keil I	Build and Run the KSDK Demo Applications using MDK25
7 Kinet	Build and Run the KSDK Demo Applications using is Design Studio
8	Revision history



### 2.2 Hardware requirement

- TWR-K22F120M Tower System module or Freescale Freedom FRDM-K22F platform
- USB A to micro AB cable
- Personal Computer

### 2.3 Toolchain requirement

- IAR embedded Workbench version 7.20.2 or later is required.
- ARM GCC 4.8.3 2014q1
- Keil MDK 5.11
- Kinetis Design Studio (KDS) v 1.0.1

## 3 Hardware Configurations

This section describes how to set up the TWR-K22F120M Tower System module and the Freescale Freedom FRDM-K22F platform for the KSDK application.

## 3.1 TWR-K22F120M Tower System module introduction

### 3.1.1 TWR-K22F120M Tower System module features

- K22FN512VDC12 MCU (120 MHz, 512 KB Flash, 128 KB RAM, 144 MAPBGA package)
- Dual-role USB interface with Micro-AB USB connector
- Onboard debug circuit: K20DX128VFM5 open (OpenSDA) with virtual serial port
- Three-axis accelerometer (MMA8451Q)
- Four (4) user-controllable LEDs
- One (1) user-controllable RGB LED
- Two (2) user-pushbutton switches for GPIO interrupts (SW1/SW3)
- One Potentiometer
- Independent, battery-operated power supply for Real Time Clock (RTC) and tamper detection modules

### 3.1.2 TWR-K22F120M Tower System module first look



Figure 1 Front side of TWR-K22F120M Tower System module



Figure 2 Back side of TWR-K22F120M Tower System module

## 3.1.3 TWR-K22F120M Tower System module jumper settings

Configure the jumper settings as indicated here. For detailed jumper options and headers, see the *TWR-K22F120M Tower System module User's Guide*.

Option	Jumper	Setting	Description
3.3 V Voltage Regulator Input Selector (VREG In	J3	5-6	Selects the K22 USB (P5V0_K22_USB) as the source voltage of the VREGIN pin on the K22 device.
Selector)		1-2	Selects the open SDA USB (P5V_TRG_SDA) as the source voltage of the System Voltage Regulator.
Board Power Selector	J4	3-5	Connect 3.3 V onboard regulator output (P3V3) to onboard supply (V_BRD)
MCU Power connection	J15	ON	Connect onboard 3.3 V or 1.8 V supply (V_BRD) to MCU VDD
MCU Power VDDA for current measurement	J12	ON	Connect MCU_PWR (3.3 V or 1.8 V) to VDDA and VREFH

#### Table 1 TWR-K22F120M Tower System module jumper settings

VBAT Power Source	J19	1-2	Connect VBAT to on-board 3.3 V or 1.8 V supply
	J2	ON	Connect V_BRD to LEDs
LED connections		1-2	Connect PTD4 to Yellow/Green LED (D7)
	J16	3-4	Connect PTD5 to Yellow LED (D5)
		5-6	Connect PTD6 to Orange LED (D4)
		7-8	Connect PTD7 to Blue LED (D3)
OpenSDA (K20) UART	J29	2-3	Connect the OpenSDA RX buffer to UART1_RX
Selection	J30	2-3	Connect the OpenSDA TX buffer to UART1_TX

## 3.2 Freescale Freedom FRDM-K22F platform introduction

### 3.2.1 Freescale Freedom FRDM-K22F platform features

- K22FN512VDC12 MCU (120 MHz, 512 KB Flash, 128 KB RAM, 144 MAPBGA package)
- Dual-role USB interface with Micro-AB USB connector
- Onboard debug circuit: K20DX128VFM5 open (OpenSDA) with virtual serial port
- Three-axis combined accelerometer and magnetometer (FXOS8700CQ)
- One (1) user-controllable RGB LED
- Two (2) user pushbutton switches for GPIO interrupts (SW2/SW3)
- One (1) audio code (SGTL5000) with line-in and audio output jacks
- One (1) visible light sensor
- One (1) micro SD card receptacle
- Connections for RF module support and Bluetooth support
- Independent, battery-operated power supply for Real Time Clock (RTC)



Figure 3 Front side of Freescale Freedom FRDM-K22F platform

Option	Jumper	Setting	Description
MCU power	J15	ON	Allows the connection or disconnection of power to the MK22FN512VLH12 device for current measurements. Note that a 0 Ohm and a 10 Ohm resistor are connected in parallel with the jumper (and each other) to allow for other power measurement options.
SWD CLK Isolation Jumper	J10	ON	Allows the OpenSDA circuitry to connect to the SWD CLK pin.
SWD DIO Isolation Jumper	J13	ON	Allows the OpenSDA circuitry to connect to the SWD DIO pin.
Reset Bypass Jumper	<b>J</b> 9	1-2	Allows the OpenSDA circuitry to control the reset pin of the target MCU. Connect 2-3 when OpenSDA is not powered.
VBAT Jumper	J21	OFF	Allows the VBAT pin to be powered by the same source as the main MCU power. (Note that there is a shorting trace on this jumper, so even though it is not populated by default, the VBAT pin is connected to the main power source of the MCU.)

#### Table 2 Freescale Freedom FRDM-K22F Default jumper settings

## 4 Build and Run the KSDK Demo Applications using IAR

This section describes the steps required to configure the IAR Embedded Workbench to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK. The hello\_world application targeted for the TWR-K22F120M Tower System hardware platform is used as an example.

## 4.1 Building the platform driver library in IAR

Before building and debugging demo applications in KSDK, the driver library project should be built to generate the library archive platform\_lib.a. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (platform.a). The platform library is prebuilt. It should not be necessary to build the library after initially downloading the KSDK. However, if it is necessary, follow these directions to rebuild the platform library.

Open the workspace file in IAR. The platform driver library project is located in this folder:

<Install\_dir>/lib/ksdk\_platform\_lib/iar/<device\_name>

The workspace file is named lib.eww:

<Install\_dir>/lib/ksdk\_platform\_lib/iar/<device\_name>lib.eww

The project file is named platform\_lib.ewp:

<Install\_dir>/lib/ksdk\_platform\_lib/iar/<device\_name>/platform\_lib.ewp

To build the platform driver library for the K22, open this workspace file in IAR: </br><Install dir>/lib/ksdk platform lib/iar/K22F51212/lib.eww

In the IAR Embedded Workbench project file, two compiler/linker configurations (build "targets") are supported:

- Debug The compiler optimization is set to low. The debug information is generated for the binary. This target should be used for developing and debugging.
- Release The complier optimization is set to high. The debug information is not generated. This target should be used for final application release.

Choose the appropriate build target: "Debug" or "Release", then click the "Make button" (highlighted by a red rectangle in this figure):

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

							_
ୡ platform_lib - IAR Embedded Wo	rkbench ID	E					
File Edit View Project Simulato	or Tools	Window Help					
D 🗲 E 🗗 🎒 X 🖻 🖻 ·	0 GI		 8	<b>X</b>	>	🗳 🎒 🖪	000 48
Workspace	×						
Debug	•						
Files	2: <b>B</b>						
🗆 🗇 platform_lib - Debug	¥						
├							
🛛 🛏 🗀 devices							
│							
📙 🖵 utilities							
🖵 🔁 Output							
🖵 🗋 platform_lib.a							

Figure 4 Platform driver library build

When the build is complete, the library (platform\_lib.a) is generated in this directory according to the build target:

- Debug <install\_dir>/lib/ksdk\_platform\_lib<toolchain>/<device\_name>/output/Debug
- Release <install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/output/Release

## 4.2 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. It is required that this library be built before compiling and downloading. To check if the library has been generated, verify that the platform\_lib.a file is located in

<Install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/<build> where build is the desired build and can be either Debug or Release. For example, if the desired project is the Debug version of the hello\_world demo application, the platform\_lib.a library should be in this folder:

<Install\_dir>/lib/ksd\_platform\_lib/iar/K22F51212/debug

Next, continue by opening the demo application project. Demo applications workspace files are located in this folder:

<install\_dir>/demos/<demo\_name>/<toolchain>/<board\_name>/<demo\_name>.eww

The hello\_world application is used as an example. The IAR workspace file is located in this folder:

<install\_dir>/demos/hello\_world/iar/twrk22f120m/hello\_world.eww

To build a demo application project, click the "Make button", which is highlighted by a red square in the this figure:

kello_world - IAR Embedded Workbench IDE       File     Edit       Yiew     Project       Yools     Window       Help       Image: State Sta		• <i>4</i> > <b>&gt; &gt; &gt; &gt; &gt; &gt; &gt; &gt; &gt; </b>
Workspace	×	hello world.c
Debug	•	int main (void)
Files  Files Files Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  Files  File		<pre>     /*********************************</pre>

Figure 5 Build the hello\_world demo application

When the build is complete, IAR shows this information in the Build window:

Messages		
hello_world.c		
pin_mux.c		
startup.c		
startup_MK64F	12.s	
system_MK64F	12.c	
Linking		
Total number o	f errors: 0	
Total number o	fwamings: 0	

Figure 6 Build hello\_world demo successfully

## 4.3 Run a demo application

To download and run the application, perform these actions:

- 1. Connect the K22 development platform to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
- 2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with these settings:
  - a. 115200 baud rate
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

🕵 PuTTY Configuration		? 🔀
PuTTY Configuration Category: 	Basic options for your PuTTY ses         Specify the destination you want to connect         Serial line         COM16         Connection type:         Raw       Telnet         Raw       Telnet         Saved Sessions         Debug         Default Settings         Debug         Close window on exit:         Always       Never         Only on clear	sion t to Speed 115200 Serial Load Save Delete
About <u>H</u> elp	Open	<u>C</u> ancel

Figure 7 Terminal (PuTTY) configurations

- 3. Ensure that the debugger configuration is correct in the project options.
  - a. The flash loader must be selected to support downloading the binary to internal Flash.

Options for node "hello_	world"
Options for node "hello_ Category: General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK Third-Party Driver XDS 100/200/ICDI	Setup       Download       Images       Extra Options       Multicore       Plugins         Attach to running targeti <ul> <li>Y Verify download</li> <li>Suppress download</li> <li>Qveride default board file</li> <li>\$TOOLKIT_DIR\$\config\flashloader\Freescale\Flash</li> <li>Edit</li> <li>Edit</li> </ul> <ul> <li>Edit</li> </ul>
	OK Cancel

Figure 8 Flash loader configurations

b. Select the appropriate debugger in the debugger setup.

General Options Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Category:	Factory Settin
Runtime Checking C/C ++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GOB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	General Options	
C/C++ Complier         Assembler         Output Converter         Custom Build         Build Actions         Linker         Debugger         Simulator         Angel         CMSIS DAP         GDB Server         IAR ROM-monitor         I-jet/JTAGjet         J-Link/J-Trace         TI Stellaris         Macraigor         PE micro         RDI         ST-LINK	Runtime Checking	
Nutration         Output Converter         Custom Build         Build Actions         Linker         Debugger         Simulator         Angel         CMSIS DAP         GDB Server         IAR ROM-monitor         I-jet/JTAGjet         J-Link/J-Trace         TI Stellaris         Macraigor         PE micro         RDI         ST-LINK	C/C++ Complier Assembler	Setup Download Images Extra Options Multicore Plugins
Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Output Converter	
Build Actions Linker Debugger Simulator Angel CMSIS DAP GOB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Custom Build	Driver 🛛 Run to
Linker  Debugger Simulator Angel CMSIS DAP GOB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Build Actions	PE micro main
Debugger       Setup macros         Simulator       Use macro file(s)         Angel          CMSIS DAP          GDB Server          IAR ROM-monitor          I-jet/JTAGjet          J-Link/J-Trace          TI Stellaris       Override default         Macraigor       \$TOOLKIT_DIR\$\CONFIG\debugger\Freescale\MK64FN1Mtb         PE micro       RDI         ST-LINK	Linker	In and
Simulator Angel CMSIS DAP GDB Server IJAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Debugger	Setup macros
Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Simulator	Use macro file(s)
CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Angel	
GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	CMSIS DAP	
IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	GDB Server	
I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	IAR ROM-monitor	
J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	I-jet/JTAGjet	Device description file
TI Stellaris Macraigor PE micro RDI ST-LINK	J-Link/J-Trace	
Macraigor PE micro RDI ST-LINK STOLKIT_DIR\$\CONFIG\debugger\Freescale\MK64FN1MQ	TI Stellaris	
PE micro RDI ST-LINK	Macraigor	\$TOOLKIT_DIR\$\CONFIG\debugger\Freescale\MK64FN1MQ
RDI ST-LINK	PE micro	
ST-LINK	RDI	
Construction of the second	ST-LINK	20

Figure 9 Debugger configurations for TWR-K22F120M Tower System module

Category:							Factory Settings
General Options							
Runtime Checking							
C/C++ Compiler	Setup	Download	Images	Extra Ontions	Multicore	Plugins	]
Output Converter	-	Dominoda	magoo	Data options	- Haldooro	1 logino	1
Custom Build	Driver			🔽 Run to	i.		
Build Actions	CMS	SDAP	-	main	2.		
Linker	Cimol	JUN		) main			
Debugger	Setu	ip macros		_			
Simulator	1	Use macro f	ile(s)				
Angel							
CMSIS DAP							
GDB Server							
IAR ROM-monitor							
I-jet/JTAGjet	Dev	ice descriptio	n file				
J-Link/J-Trace	1	Override def	ault				
Macraicar			DALCONE		1.510	(CATNEL	
PE micro	5	OOLKII_DI	R\$\CONF	IG \debugger \F	reescale \IVII	464FN IN	112
PDI							
RDI							

Figure 10 Debugger configurations for Freescale Freedom FRDM-K22F platform

c. SWD should be configured as the debugger interface in the debugger specific category.

Category:		Factory Settings
General Options Runtime Checking C/C++ Compiler Assembler Output Converter	Setup P&E Hardware interface type	Communication
Custom Build	OpenSDA - USB	USB
Linker	Reset delay	Device 1 💌
Debugger Simulator	ms 🔘 JTAG	Serial port
Angel	ITAG/SWD speed	COM1 +
GDB Server IAR ROM-monitor	5000 kHz	TCP/IP
J-Link/J-Trace		10.0.0.1
TI Stellaris Macraigor	Log communication	
PE micro	\$PROJ_DIR\$\cspycomm.log	
RDI		
Third-Party Driver		

Figure 11 Debugger configurations for TWR-K22F120M Tower System module (PE micro)

Category:			Factory Setting
General Options			
Runtime Checking			
C/C++ Compiler	Setup JTAG/SWI	Breakpoints	
Output Converter	Probe config	Probe configuration file	
Custom Build	a tobe coning		
Build Actions	<ul> <li>Auto</li> </ul>		
Linker	From file		
Debugger	C Explicit	CPU Select	
Simulator			
Angel	Interface	Explicit probe configuration	
CMSIS DAP	) JTAG	Multi-target debug system	
GDB Server	© SWD	Target number (TAP or Multidrop ID):	
IAR ROM-monitor	SWD	Terrent with an diale CDUs	
1-jet/JIAGjet			
TT Stellaris		CPU number on target:	
Macraigor	JTAG/SWD speed		
PE micro	Auto detect 🔻		
RDI			
ST J INK		3	

Figure 12 Debugger configurations for Freescale Freedom FRDM-K22F platform (CMSIS-DAP)

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

4. When the application is built, press the "Download and Debug button" to download the application to the target.



Figure 13 Download and Debug button

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

Workspace		×	hello	world.c		
Debug		•]	- /************************************			
Files	8.a.	D:		* Code		
<b>□                                    </b>	*			*******		
→			•	int main (void)		
Left Cutput				<pre>/************************************</pre>		
				<pre>/*! @param receiveBuff Buffer used t uint8_t receiveBuff[19] = {0};</pre>		

Figure 14 Stop at main() when run debugging

Now run the code by clicking on the "go button" to start the application:



Figure 15 Go button

6. The hello\_world application should now be running and this banner should be displayed on the terminal. If this is not the case, check the terminal settings and terminal connections.



Figure 16 Main prompt of the hello\_world demo

## 5 Build and Run the KSDK Demo Applications using ARM GCC

This section describes the steps required to configure the ARM GCC toolchain to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK. The hello\_world demo application targeted for the TWR-K22F120M Tower System module hardware platform is used as an example.

### 5.1 Environment setup

### 5.1.1 Install GCC ARM v4.8.3 2014q1 embedded toolchain

- 1. Download the Windows installer.
- 2. Install the toolchain in the /Program Files/ location, which is usually on your "C:\" drive.

GNU Tools for ARM Embedded Processors 4.8 2014q1 Setup
Choose Destination Location
Where should GNU Tools for ARM Embedded Processors 4.8 2014q1 be installed?
Setup will install GNU Tools for ARM Embedded Processors in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
C:\Program Files\GNU Tools ARM Embedded\4.8 2014q1 Browse
InstallJammer
< Back Next > Cancel

Figure 17 Install GCC ARM embedded toolchain

### 5.1.2 Install MinGW and MSYS

- 1. Download the MinGW installer, which is located here.
- 2. Run mingw-get-setup.exe and select the installation path, such as: C:/MINGW.
- 3. Select the mingw32-base and the msys-base under the Basic Setup as shown in Figure 18 MinGW Installer.

Installation Package	Settings				
sic Setup	Package	Class	Installed	Repository Version	Description
1 Packages	mingw-developer-toolki	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
	sys-base	bin		2013072300	A Basic MSYS Installation (meta)
	15 17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1				

Figure 18 MinGW Installer

4. Click "Apply Changes" from the "Installation" menu to install packages, as shown here.

Installation Package	Settings					
Update Catalogue		kage	Class	Installed	Repository Version	Description
Mark All Upgrades		gw-developer-toolkit	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
Apply Changes		gw32-base	bin		2013072200	A Basic MinGW Installation
Quit Alt+F4		gw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
		gw32-gcc-fortran	bin		4.8.1-4	The GNU FORTRAN Compiler
	mi mi	ngw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
i mi		ngw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
	S ===	ys-base	bin		2013072300	A Basic MSYS Installation (meta)

Figure 19 MinGW Installer apply change

### 5.1.3 Configure system environment

1. Update the system environment variable "Path" to include the MINGW installation folder, such as the <drive>\MINGW\msys\1.0\bin;<drive>:\MINGW\bin.

System Properties	X
Computer Name Hardware	Advanced System Protection Remote
Environment Variables	X
Edit User Variable	X
Variable same:	LIT I
variable name; P	AIN
Variable value: C	C:\MinGW\msys\1.0\bin;C:\MinGW\bin
	OK Cancel

Figure 20 Environment variable update

2. Run the GCC Command prompt in Start->All Programs-> GNU Tools ARM Embedded 4.8.3 2014q1.



Figure 21 PATH environment

3. When using a Windows command line, add the environment variable, ARMGCC\_DIR, which is also the short name of the ARM GCC installation path.

Environment Variable	25	×
User variables for B4	5603	
Variable	Value	<b>_</b>
ARMGCC_DIR	C:\PROGRA~1\GNUTOO~1\43F2B~1.720	
Edit User Variable	-	×
Variable name:	ARMGCC_DIR	
Variable value:	C:\PROGRA~1\GNUTOO~1\43F2B~1,720	-
Г	OK Cancel	┛┢
Variable	Value	ᠴ
CLASSPATH	.;C:\Program Files\Java\jre6\lib\ext\QT	
ComSpec	C:\Windows\system32\cmd.exe	
CYGWIN	ntsec tty	
FP_NO_HOST_C	NO	-
	New Edit Delete	
	OK Cance	1

#### Figure 22 Add environment variable

**4.** When using GIT Bash or Cygwin, set the environment variable to "export ARMGCC\_DIR=C:/PROGRA~/GNUTOO~1/4298B~1.820".

#### Note

Use a forward slash  $^{\prime\prime}$  as a separator.

When using the KDS GCC toolchain, add the system environment variable, KDSGCC\_DIR, which is also the path of KDS GCC toolchain. Use a short name and run the "mingw32-make toolchain=kdsgcc".

>/mk/common.mk

## 5.2 Building the platform driver library in ARM GCC

Before building and debugging any demo applications in the SDK, the driver library project must be built to generate the necessary library archives (platform\_lib.a). This library contains all binary codes for the HAL and peripheral drivers specific to the chip and each device has its own library archive.

To build the platform library, change the current directory in the ARM GCC command prompt to: <install\_dir>/lib/ksdk\_platform\_lib/gcc/<device\_name>/

Using K22 as an example, the correct path would be: <install\_dir>/lib/ksdk\_platform\_lib/gcc/K22F51212.

Once the command prompt path has been correctly set, run the command "mingw32-make build=debug" to build the debug library or "mingw32-make build=release" to build the release library.

Once the build has successfully completed, the platform\_lib.a file (library archive) is located in these directories:

Debug - <install\_dir>/lib/ksdk\_platform\_lib/gcc/<device\_name>/Debug

Release - <install\_dir>/lib/ksdk\_platform\_lib/gcc/<device\_name>/Release

## 5.3 Build a demo application

The KSDK demo applications use a prebuilt linkable library to compile in the necessary functions. Therefore, it is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the platform\_lib.a file resides in <Install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/<build>

where the build is the desired build, which is either Debug or Release. For example, if the desired project to run is the Debug version of the hello\_world demo application, it is required that the platform\_lib.a library exists in this folder:

<Install\_dir>/lib/ksd\_platform\_lib/gcc/K22F51212/debug

Continue by changing the directory of your ARM GCC compiler to the directory where the application makefile resides. This location is <install dir>/demos/<toolchain>/<board name>/Makefile

The hello\_world application is used as an example. The GCC Makefile is located in: <install\_dir>/demos/hello\_world/gcc/twrk22f120m/

To build a demo application project, open an ARM GCC command prompt and change the directory to the location where the desired Makefile is stored (listed above). Then execute the command "mingw32-make build=<target\_build> target=<target\_mem\_location>, where build and target can be one of these:

#### Table 3 Build and Target Options

Build options	Target options
Debug	Flash
Release	SRAM

When the build is complete, the GCC command prompt appears as shown:



Figure 23 hello\_world demo successfully built

## 5.4 Run a demo application

To download and run the application, perform these actions:

- 1. If you do not already have the Segger J-Link software and documentation installed, do this first. Go to the segger.com/jlink-software.html to download the package. Otherwise, continue to step 2.
- 2. For Freedom platform, remove the OpenSDA isolation jumpers (jumpers J10 and J13 for the Freedom platform, and jumpers J36 and J37 for the Tower System) if they are populated.
- 3. Connect your J-Link debug pod to the MK22F512 SWD connector (J31) of the Freedom platform.
- 4. Open the J-Link GDB Server application. After opening this application, modify your connection settings as shown in this figure:

SEGGER J-Link GDB Server V4.80h - Config
Connection to J-Link
C ICP/IP
_ Taraet device
MK22FN512xxx12
Target interface
Speed
<ul> <li>⊙ Auto selection</li> </ul>
C Adaptive <u>c</u> locking
C 1000 🖃 kHz
Command line option
-select USB -device MK22FN512xxx12 -if SWD -speed auto
OK Cancel

Figure 24 Segger J-Link GDB server configurations

5. Once connected, the screen appears as shown:



Figure 25 Segger J-Link GDB Server screen after successful connection

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

<SDK root>\demos\hello\_world\gcc\twrk22f120m\Flash\_Debug.

- 7. Run the command "arm-none-eabi-gdb <DEMO\_NAME>.elf". For this example, "arm-none-eabi-gdb hello\_world.elf".
- 8. Run these commands:
- "target remote localhost: 2331"
  - a. "monitor reset"
  - b. "monitor halt"
  - c. "load"
  - d. "monitor reset"

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

- 9. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with these settings:
  - e. 115200 baud rate
  - f. No parity
  - g. 8 data bits
  - h. 1 stop bit

🕵 PuTTY Configuration	? 🔀		
Category:			
Session	Basic options for your PuTTY session		
Logging	Specify the destination you want to connect to		
	Serial li <u>n</u> e Speed		
Bell	COM16 115200		
Features	Connection type:		
⊡ · Window	○ Raw ○ Telnet ○ Rlogin ○ SSH		
Appearance     Appearance     Selection     Colours     Connection     Proxy     Froxy     Telnet     Rlogin     SSH	Load, save or delete a stored session Sav <u>e</u> d Sessions Debug Default Settings Debug Save Delete		
Serial	Close window on e <u>xi</u> t: Always Never Only on clean exit		
About <u>H</u> elp	<u>Open</u> <u>C</u> ancel		

Figure 26 Terminal (PuTTY) configurations

10. The application is now downloaded and connected. You may execute the "monitor go" command to begin the demo application.

11. The hello\_world demo application should now be running and this banner should be displayed on the terminal. If this is not the case, check the terminal settings and terminal connections.



Figure 27 Main prompt of the hello\_world demo

## 6 Build and Run the KSDK Demo Applications using Keil MDK

This section describes the steps required to build, run, and debug demo applications and necessary driver libraries provided in the Freescale KSDK using ARM MDK v5 (Keil version 5). The Kinetis K20 Series Device Support and examples DFP 1.1.0 must be updated. The hello\_world demo application is used as an example (TWR-K22F120M Tower System module hardware platform).

#### NOTE

Before opening the K22 KSDK demos and/or the libraries, the KEIL Device Family Pack for the K22 series should be installed. This software pack can be downloaded from the Freescale MK22FN512xxx12 webpage on KEIL's website at http://www.keil.com/dd2/freescale/mk22fn512xxx12/.

## 6.1 Building the platform driver library in Keil MDK

Before building and debugging demo applications in the KSDK, the driver library for the target device should be built. The library archive for the Keil demo applications is named **ksdk\_platform\_lib.lib**. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (ksdk\_platform\_lib.lib). The platform library is not prebuilt. Therefore, it is necessary to build the library after initially downloading the KSDK.

Open the hello\_world multiproject workspace file (hello\_world.uvmpw) in Keil (Project->Open project...). The demo application multiproject workspace is located in this folder:

<install\_dir>/demos/<demo\_name>/<toolchain>/<board\_name>/<demo\_name>.uvmpw

Using the hello\_world as an example, the path should be:

<Install\_dir>/demos/hello\_world/uv4/twrk22f120m/hello\_world.uvmpw

To build the platform driver library for the K22, first set the ksdk\_platform\_lib as the active project, by right-clicking on the ksdk\_platform\_lib and selecting "Set as Active Project".



Figure 28 Selection of the ksdk\_platform\_lib as the active project

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

In the ksdk\_platform\_lib project, two compiler/linker configurations (build "targets") are supported:

- Debug The compiler optimization is set to low. The debug information is generated for the binary. This target should be used for developing and debugging.
- Release The complier optimization is set to high. The debug information is not generated. This target should be used for final application release.

Choose the appropriate build target: "Debug" or "Release" from the drop-down box.



Figure 29 Drop-down selection of the Debug/Release target

Rebuild the project files by left-clicking the "Rebuild button".



Figure 30 Rebuild all button

When the build is complete, the library (ksdk\_platform\_lib.lib) is generated in this directory according to the build target:

- Debug <install\_dir>/lib/ksdk\_platform\_lib<toolchain>/<device\_name>/ Debug
- Release <install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/ Release

## 6.2 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. It is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the ksdk\_platform\_lib.lib file is in

<Install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/<build> location, where the build is the desired build and can be either Debug or Release. For example, if the desired project is the Debug version of the hello\_world demo application, the platform\_lib.a library should be in this folder:

 $<\!Install\_dir\!>\!/lib/ksdk\_platform\_lib/uv4/K22F51212/debug$ 

Continue by opening the demo application project. Demo applications workspace files are located in this folder:

<install\_dir>/demos/<demo\_name>/<toolchain>/<board\_name>/<demo\_name>.uvmpw

The hello\_world application is used as an example. The Keil multi-project workspace file is located in this folder:

<install\_dir>/demos/hello\_world/uv4/twrk22f120m/hello\_world.uvmpw

To build a demo application project, click the "Rebuild button", which is highlighted by a red square in this figure:



Figure 31 Build the demo application

When the build is complete, Keil shows this information in the Build Output window:

Build Output
Rebuild Project 'hello_world' - Target 'hello_world Debug'
compiling fsl_misc_utilities.c
compiling fsl_debug_console.c
assembling startup_MK22F51212.s
compiling system_MK22F51212.c
compiling startup.c
compiling hello_world.c
compiling fsl_uart_irq.c
compiling gpio_pins.c
compiling pin_mux.c
compiling hardware_init.c
linking
Program Size: Code=16436 RO-data=1228 RW-data=36 ZI-data=65716
"debug\hello_world.out" - 0 Error(s), 0 Warning(s).

Figure 32 Build Output window upon successful hello\_world build

## 6.3 Run a demo application

To download and run the application, perform these actions:

- 1. Connect the K22 development board to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
- 2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with these settings:
  - a. 115200 baud rate
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

🔀 PuTTY Configuration		? 🗙
Category: 	Basic options for your PuTTY sess         Specify the destination you want to connect         Serial line         COM16         Connection type:         Raw       Telnet         Rlogin       SSH         Load, save or delete a stored session         Saved Sessions         Debug         Default Settings         Debug         Close window on exit:         Always       Never         Only on cle	ion to Speed 115200 Serjal Load Save Delete an exit
About <u>H</u> elp	<u>Open</u>	<u>C</u> ancel

Figure 33 Terminal (PuTTY) configurations

- 3. Ensure that the debugger configuration is correct in the project options.
  - a. To verify the debugger configurations, first open Options for the Target using the Options for Target dialog button



Figure 34 Options for Target dialog button

b. In the "Options for Target …" dialog box, select the Debug tab and ensure that the simulator is not selected and the correct debug driver is selected. If the target is a TWR-K22F120M Tower System development card, the PEMicro Debugger should be selected. If the target is a Freescale Freedom FRDM-K22F platform development system, the CMSIS-DAP should be selected.

Options for Target 'hello_world Debug'				
Device   Target   Output   Listing   User   C/C++   Asm	Linker Debug Utilities			
C Use <u>Si</u> mulator Settings	€ Use: PEMicro Debugger     ✓ Settings			
Limit Speed to Real-Time				
Initialization File: Initialization File:	C Load Application at Startup Run to main()			
Edit	Edit			
Restore Debug Session Settings	Restore Debug Session Settings			
Breakpoints Toolbox	Breakpoints Toolbox			
Watch Windows & Performance Analyzer	Vatch Windows			
CPU DLL: Parameter:	Driver DLL: Parameter:			
SARMCM3.DLL	SARMCM3.DLL			
Dialog DI L · Parameter·	Dialog DI L · Parameter·			
DCM.DLL -pCM4	TCM.DLL pCM4			
, ,	., ,			
ОК Са	ncel Defaults Help			

Figure 35 Debug driver selection for TWR-K22F120M Tower System module

Options for Target 'hello_world Debug'			
Device   Target   Output   Listing   User   C/C++   Asm	Linker Debug Utilities		
C Use Simulator Settings	⊡se: CMSIS-DAP Debugger     Settings     Settings     Settings		
Limit Speed to Real-Time	- <u> </u>		
Load Application at Startup     Run to main()     Initialization File:	I Load Application at Startup I Run to main() Initialization File:		
Edit	Edit		
Restore Debug Session Settings	Restore Debug Session Settings		
Breakpoints     Toolbox     Windows & Porformance Analyzor	Breakpoints     Toolbox     Windown		
Memory Display System Viewer	Memory Display Vistem Viewer		
CPU DLL: Parameter:	Driver DLL: Parameter:		
Dialog DI L. Parameter	Dialog DI Li Parameter:		
DCM.DLL PCM4	TCM.DLL Parameter.		
	., ,		
ОК Са	ncel Defaults Help		

Figure 36 Debug driver selection for Freescale Freedom FRDM-K22F platform

c. Next, click the "Settings button" next to the debug driver selection drop-down box. Ensure that the settings are correct for the appropriate development platform.

P&E Connection Manager - v1.28.00.00
Please select connection interface, port, and settings for connection.
Connection port and Interface Tupe
Add LPT Port
Refresh List
Port USB1 : OpenSDA (76B4CE17)
Interface Detected : Firmware Version :
Target CPU Information
CPU: K22FN512M12
BDM Communication Speed
PC Parallel Port wait states : IO_DELAY_CNT = 0
Debug Shift Speed = (0) : Shift Frequency = 10.000Mhz
BDM_SPEED = 0
CMCI Unternal Bus Erequency (For programming)
Auto-Detect
C MCU Internal Bus frequency (FREQ) in Hz = 0 (Decimal)
Reset Options
Delay after Reset and before communicating to target for     U milliseconds (decimal).
Power Control for Cyclone / TraceLink / Multilink Universal EX
Provide power to target     Regulator Output Voltage     Power Down Delay     250 mS
Power off target upon software exit 3V Power Up Delay 250 mS
<u>O</u> k <u>C</u> ancel
□ Show this dialog before attempting to contact target (Otherwise only display on Error)

Figure 37 Debugger configurations for TWR-K22F120M Tower System module

CMSIS-DAP - JTAG/SW Adapter	-SW Dev SWDIO	IDCODE	Device Name ARM CoreSight SW-	DP	Move
Firmware Version: 1.0	ာ Aut က Ma	iomatic Detection nual Configuration	ID CODE:		Down
Max Clock:       IMHz       Add       Delete       Update       AP:       0x00         Debug       Connect & Reset Options       Cache Options       Download Options         Connect:       Nomal       Reset:       Autodetect       Image: Cache Options       Download Options         Image: Connect:       Nomal       Reset:       Autodetect       Image: Cache Options       Download Options         Image: Connect:       Reset:       Autodetect       Image: Cache Options       Image: Cache Options       Image: Cache Options         Image: Connect:       Reset:       Autodetect       Image: Cache Options       Image: Cache Options       Image: Cache Options         Image: Connect:       Image: Cache Options       Image: Cache Options       Image: Cache Options       Image: Cache Options         Image: Connect:       Image: Cache Options         Image: Connect       Image: Cache Options         Image: Connect       Image: Cache Options         Image: Cache Options       Image: Cach					

Figure 38 Debugger configurations for Freescale Freedom FRDM-K22F platform

d. If the target debugger is the CMSIS-DAP, the Flash loader must also be configured properly. To verify that the flash loader is configured properly, select the "Flash Download" in the Cortex-M Target Driver Setup dialog box and verify that the flash loader settings are configured as shown in this figure.

Cortex-M Target Driver Setup	<ul> <li>)00-(4</li> </ul>	a luna la	* [ mm ]	×
Debug Flash Download				
Download Function C Erase Full Chip Erase Sectors C Do not Erase	<ul> <li>Program</li> <li>Verify</li> <li>Reset and Ru</li> </ul>	RAM for Alg Start: 0x2	orithm 20000000 Size: 0x1000	
Programming Algorithm				
Description	Device Size	Device Type	Address Range	
		Start:	Size:	
	Add	Remove		
	ОК	Cancel		Help

Figure 39 Flash loader configurations for Freescale Freedom FRDM-K22F platform

4. Once the application has been properly built and the debugger configurations have been verified, press the "Download button" to download the application to the target.



#### Figure 40 Download Button

5. After clicking the "Download button", the application has been downloaded to the target and should be running. To debug the application as well, click the "Start/Stop Debug Session button".



Figure 41 Start/Stop Debug Session Button

After clicking the "Start/Stop Debug Session button", the debugger resets the target device and stop at main():



Figure 42 Stop at main() when run debugging

Now you can run the code by clicking on the "Run button" to start the application:

RST		∞	<b>{</b> ⁴}}	0	{}	*{}
Regis	ters 📑	Ru	n (F5)		1 🕅	Disa
Regi	ster	Start code execution			on <sup>0:</sup>	

Figure 43 Go Button

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

6. The hello\_world application should now be running and the following banner should be displayed on the terminal. If this is not the case, check the terminal settings and terminal connections.



Figure 44 Main prompt of the hello\_world demo

## 7 Build and Run the KSDK Demo Applications using Kinetis Design Studio

This section describes the steps required to build, run, and debug demo applications and necessary driver libraries provided in the KSDK using Kinetis Design Studio. The hello\_world demo application is used as an example (TWR-K22F120M Tower System module hardware platform).

## 7.1 Installing KSDK Eclipse update

Before using any Eclipse-based IDE with KSDK, apply the SDK Eclipse update, titled KSDK\_<version>\_Update\_for\_Eclipse. Without the update, Eclipse cannot generate KSDK-compatible projects. To install the update, follow these instructions:

1. Select Help > Install New Software.



Figure 45 Install new software

2. In the "Install New Software" dialog box, select the "Add..." button in the upper right corner.

3. In the "Add Repository" dialog, select "Archive..."

Select a site	or enter the location of a	site.			
Work with:	type or select a site				· Add
Charles to			Find more softwar	e by working with the <u>"Availal</u>	ole Software Sites" preference
Name	ĸt		Version		
🗐 🚺 T	here is no site selected.	Add Repository			
		Location: http://		Archive	

Figure 46 Add repository

- 4. In the dialog box that appears, browse your KSDK install directory.
- 5. From the top-level, enter the tools/eclipse\_update folder and select the Eclipse update .zip file. The file name depends on the version of SDK that you are using. For example, the update file in the SDK 1.0.0 version is SDK\_1.0.0-GA\_Update\_for\_Eclipse.zip.
- 6. Click "Open", then "OK" in the "Add Repository" dialog box.
- 7. The KSDK update now shows up in the list of the original Install dialog.
- 8. Check the box to the left of the KSDK Eclipse update and click "Next" in the lower right corner.
- 9. Follow the remaining instructions to finish the installation of the update.
- 10. After the update is applied, restart the KDS/Eclipse for the changes to take effect.

## 7.2 Building the platform driver library in Kinetis Design Studio

Before building and debugging demo applications in the KSDK, the driver library for the target device should be built. The library archive for the KDS demo applications is named ksdk\_platform\_lib.a. This library contains all HAL and peripheral driver functions which are device-specific. Therefore, each device has its own library (ksdk\_platform\_lib.a). The platform library is not prebuilt. Therefore, it is necessary to build the library after initially downloading the KSDK.

## 7.2.1 Open the library project in KDS

1) Select File->Import... from the KDS Eclipse menu.



Figure 47 Selection of the correct import type in KDS

2) Select the "Select root directory:" option. Then click "Browse..." to point KDS to the correct library.

🧱 Import	
Import Projects Select a directory to search for existing Eclipse projects.	
Select root directory:     Select archive file:     Projects:	Browse           v         Browse
	Select All Deselect All Refresh
Options Search for nested projects Copy projects into workspace Working sets	
Add project to working sets	* S <u>e</u> lect
< Back Next >	Einish Cancel

Figure 48 Import Projects directory selection window

3) Point KDS to the ksdk\_platform\_lib project in the K22F51212 (the library project is located at <Install\_dir>/lib/ksdk\_platform\_lib/kds/<device\_name>). Once you have done this, your Import Projects directory selection window should look like the following.

🏽 Import		- <b>O</b> X
Import Projects Select a directory to searc	ch for existing Eclipse projects.	
<ul> <li>Select root directory:</li> <li>Select archive file:</li> </ul>	mcu-sdk\lib\ksdk_platform_lib\kds\K22F51212 +	B <u>r</u> owse
Projects:	_K22F51212 (C:\Users\b38350\Documents\CodeRep	<u>Select All</u> Deselect All Refresh
Options     Search for nested pro	jects	
<u>Copy projects into we</u> Working sets     Add projec <u>t</u> to work     Working sets:	ing sets	S <u>e</u> lect
?	< <u>B</u> ack Next > Finish	Cancel

Figure 49 Import Projects directory selection window after selecting the K22F51212 ksdk\_platform\_lib project.

4) Click Finish.

## 7.2.2 Build the library project

In the Kinetis Design Studio platform library project, two compiler/linker configurations (build "targets") are supported:

- Debug The compiler optimization is set to low. The debug information is generated for the binary. This target should be used for developing and debugging.
- Release The complier optimization is set to high. The debug information is not generated. This target should be used for final application release.

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

sign Studio [beta]				
Refac <u>t</u>	or	<u>N</u> avigate	Se <u>a</u> rch	Pro
	4	- 🗟 🗲	ð	_
	✓	1 Debug		
		2 Release		Co

Figure 50 Selection of build target in KDS

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

If the library build does not begin after selecting the desired target, left-click the hammer icon to begin the build. When the build is complete, the library (ksdk\_platform\_lib.a) is generated in this directory according to the build target:

Debug - <install\_dir>/lib/ksdk\_platform\_lib<toolchain>/<device\_name>/Debug

Release - <install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/Release

## 7.3 Build a demo application

The KSDK demo applications utilize a prebuilt linkable library to compile the necessary functions. Therefore, it is required that this library be built before compiling and downloading. To check that this library has been generated, verify that the ksdk\_platform\_lib.a file is in the

<Install\_dir>/lib/ksdk\_platform\_lib/<toolchain>/<device\_name>/<build> location, where build is the desired build and can be either Debug or Release. For example, if the desired project is the Debug version of the hello\_world demo application, the ksdk\_platform\_lib.a library should be in this folder:

<Install\_dir>/lib/ksdk\_platform\_lib/kds/K22F51212/debug

Continue by opening the demo application project. Demo applications workspace files are located in the following folder:

<install\_dir>/demos/<demo\_name>/<toolchain>/<board\_name>/<demo\_name>.eww

Follow the steps in the above section 7.2.1. Open the library project to open the hello\_world demo application project. The project is located in the following folder:

<install\_dir>/demos/hello\_world/kds/twrk22f120m/

To build the demo application project, select the target to build by left-clicking the arrow next to the hammer icon.



#### Figure 51 Selection of build target in KDS for the demo application

If your target application does not begin building immediately, left-click the hammer icon. When the build is complete, the KDS console window appears with the following information:



Figure 52 Console window output upon successful build

## 7.4 Run a demo application

To download and run the application, perform these actions:

### NOTE

Before continuing with the following steps, ensure you have copied the Segger J-Link OpenSDA debug firmware to your OpenSDA hardware. If you are using the TWR-K22F120M, the original OpenSDA J-Link firmware is needed. If you are using the FRDM-K22F platform, the OpenSDA v2 J-Link firmware is needed. Both versions of the firmware can be found at https://segger.com/opensda.html.

For more details on changing your OpenSDA firmware, consult the Quick Start documentation for your Freescale Development Platform.

- 1. Connect the K22 development board to your PC via USB cable between the OpenSDA USB connector and the PC USB connector.
- 2. Open the terminal application on the PC, such as PuTTY or TeraTerm, and connect to the OpenSDA serial port number. Configure the terminal with the following settings:
  - a. 115200 baud rate
  - b. No parity
  - c. 8 data bits
  - d. 1 stop bit

🕵 PuTTY Configuration	? <mark>- × -</mark>
Category:	
Category: 	Basic options for your PuTTY session         Specify the destination you want to connect to         Serial line       Speed         COM16       115200         Connection type:       Rlogin       SSH         Raw       Telnet       Rlogin       SSH         Load, save or delete a stored session       Saved Sessions       Load         Debug       Load       Load       Save         Debug       Load       Save       Delete         Close window on exit:       Only on clean exit       Only on clean exit
About <u>H</u> elp	Open <u>C</u> ancel

Figure 53 Terminal (PuTTY) configurations

- 3. Ensure that the debugger configuration is correct in the project options.
  - a. To check the debugger configurations, click the down arrow next to the green debug button and select "Debug Configurations".



Figure 54 Debug Configurations dialog button

b. In the Debug Configurations dialog box, select debug configuration from the GDB SEGGER J-Link Debugger in the groups on the left-hand side of the dialog box.

Bebug Configurations		
Create, manage, and run configurations		
Image: Second secon	Name: hello_world_twrk22f120r Main * Debugger b Sta C/C++ Application: debug/hello_world_twrk22f120	m debug jlink
C GDB Hardware Debugging C GDB OpenOCD Debugging C GDB PEMicro Interface Debugging C GDB SEGGER J-Link Debugging	Project: hello_world_twrk22f120m	Variables Searc <u>n</u> Project Browse Browse
hello_world_twrk22f120m debug jlink      hello_world_twrk22f120m release jlink     Launch Group	Build (if required) before laun Build configuration:	Crung Debug ▼ ▼ Select configuration using 'C/C++ Application'
	<ul> <li>Enable auto build</li> <li>Use workspace settings</li> </ul>	Disable auto build <u>Configure Workspace Settings</u>
Filter matched 12 of 12 items		Apply Revert
?		Debug Close

Figure 55 Selection of the debug configuration in the Debug Configurations dialog box.

c. In the debugger setup, verify that the C/C++ Application and Project are set to the correct path (the C/C++ Application path should be debug/<application name>.elf and the Project should be the target project name).

🛞 Debug Configurations		×
Create, manage, and run configurations		Ť.
Ype filter text         C/C++ Application         C/C++ Attach to Application         C/C++ Postmortem Debugger         C/C++ Remote Application         GDB Hardware Debugging         GDB SopenOCD Debugging         GDB SEGGEP LLink Debugging         C hello world twrk22f120m debug link         E hello world twrk22f120m release jlink         ► Launch Group	Name:       hello_world_twrk22f120m debug jlink         Main       梦 Debugger       > Startup       Source       Common         C/C++ Application:       debug/hello_world_twrk22f120m.elf         Project:       hello_world_twrk22f120m         Build (if required) before launching       Build configuration:       Debug         Ø Select configuration using 'C/C++ Application         © Enable auto build       © Disable auto build         @ Use workspace settings       Configure Workspace Settings	Browse Browse
Filter matched 12 of 12 items	Apply	Revert
0	Debug	Close

Figure 56 KDS debugger configurations for TWR-K22F120M Tower System module

d. Once the debugger configurations are correct, click the "Debug button".

😹 Debug Configurations					
Create, manage, and run configurations					
Image: Second secon	Name: hello_world_twrk22f120	Om debug jlink			
C/C++ Application C/C++ Attach to Application	C/C++ Application:				
<ul> <li>C/C++ Postmortem Debugger</li> <li>C/C++ Remote Application</li> <li>GDB Hardware Debugging</li> <li>GDB ODE ODE Debugging</li> <li>GDB SEGGER J-Link Debugging</li> <li>C hello_world_twrk22f120m debug jink</li> <li>C hello_world_twrk22f120m release jink</li> <li>Launch Group</li> </ul>	debug/hello_world_twrk22f12	00m.elf <u>V</u> ariables Searc <u>h</u> Project B <u>r</u> owse			
	hello_world_twrk22f120m	Browse			
	Build configuration:	Debug v			
	<ul> <li>Enable auto build</li> <li>Use workspace settings</li> </ul>	Select configuration using 'C/C++ Application'  Configure Workspace Settings			
Filter matched 12 of 12 items		Apply Reyert			
0		Debug Close			

Figure 57 Debug button in Debug Configurations dialog box

4. The application is downloaded to the target and automatically run to main():



Figure 58 Stop at main() when run debugging

Now you can run the code by clicking on the "Resume button" to start the application:



Figure 59 Resume Button

5. The hello\_world application should now be running and the following banner should be displayed on the terminal. If this is not the case, check the terminal settings and terminal connections.



Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

## 8 Revision history

This table summarizes revisions to this document.

Revision History				
Revision number	Date	Substantial changes		
1.0.0	7/2014	Initial release		

Kinetis SDK K22 User's Guide, Rev. 1.0.0, 07/2014

How to Reach Us:

Home Page: freescale.com

Web Support: 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 and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2014 Freescale Semiconductor, Inc.





