Freescale Semiconductor, Inc.

KSDKK64UG
User's Guide Rev. 1.0.0-beta, 03/2014

Kinetis SDK K64 User's Guide

1 Introduction

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

2 Overview

2.1 Kinetis SDK

Kinetis SDK is a Software Development Kit that provides software support for the core and peripherals for Freescale devices with the ARM 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 peripheral driver and HAL usage to highlight the main features of the targeted SoCs. Also, the KSDK contains the latest available RTOS kernels, USB stacks, and other software components used on the supported evaluation boards.

Contents

2.2 Hardware requirement

TWR-K64F120M Tower System module or FRDM-K64F board is required.

2.3 Toolchain requirement

IAR embedded Workbench version 6.70.3 or GCC ARM Embedded 4.7.4 Tool is required. To use the IAR 6.70.3 for K64 development, you need to apply a flash loader patch from IAR. Download this patch here, and unzip it under the IAR install folder like: C:\Program Files\IAR Systems\Embedded Workbench 6.5

3 Hardware Configurations

This section describes how to set up the TWR-K64F120M Tower System module and FRDM-K64F for the KSDK application.

3.1 TWR-K64F120M Introduction

3.1.1 TWR-K64F120M features

- K64FN1M0VMD12 MCU (120 MHz, 1024 KB Flash, 260 KB RAM, low power, 144 MAPBGA package)
- Dual-role USB interface with Micro-AB USB connector
- On-board debug circuit: K20DX128VFM5 open (OpenSDA) with virtual serial port
- Three-axis accelerometer (MMA8451Q)
- Four (4) user-controllable LEDs
- Two (2) user push-button switches for GPIO interrupts (SW1/SW3)
- One potentiometer
- Independent, battery-operated power supply for the Real Time Clock (RTC)

3.1.2 TWR-K64F120M first look

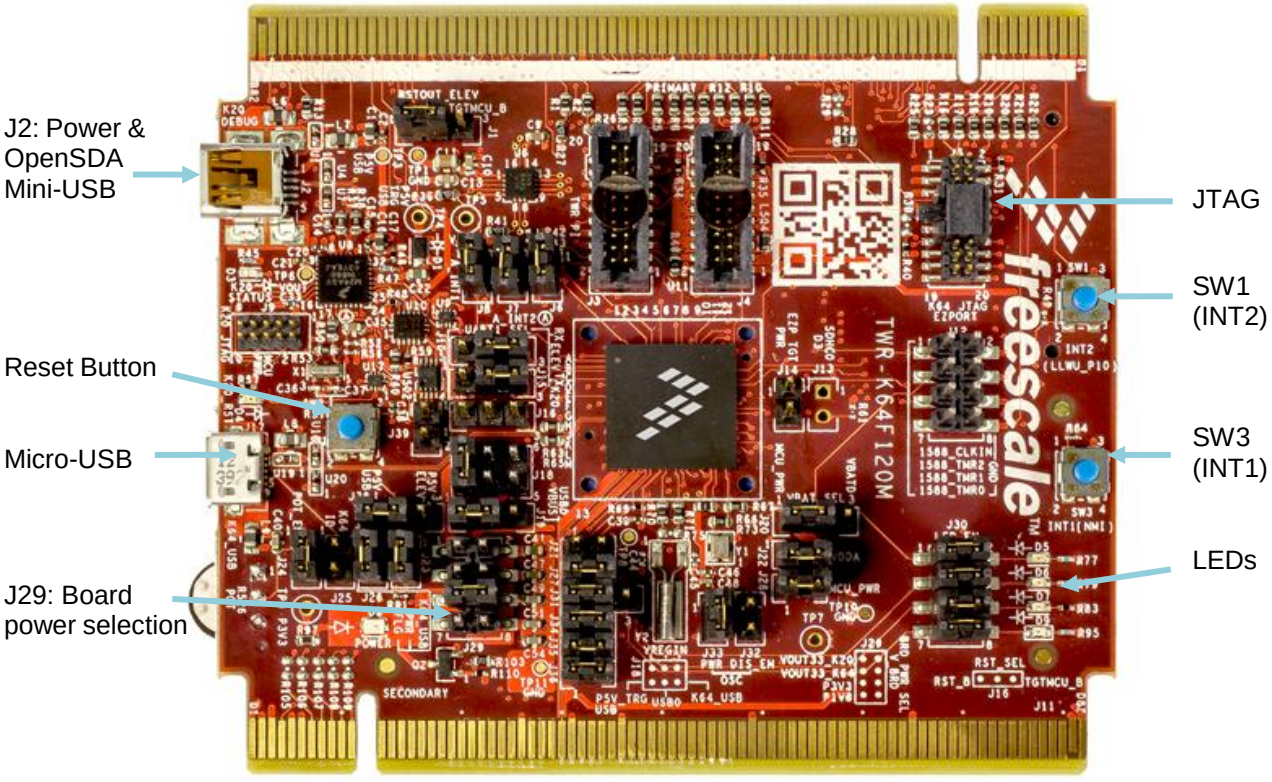


Figure-1 Front side of TWR-K64F120M

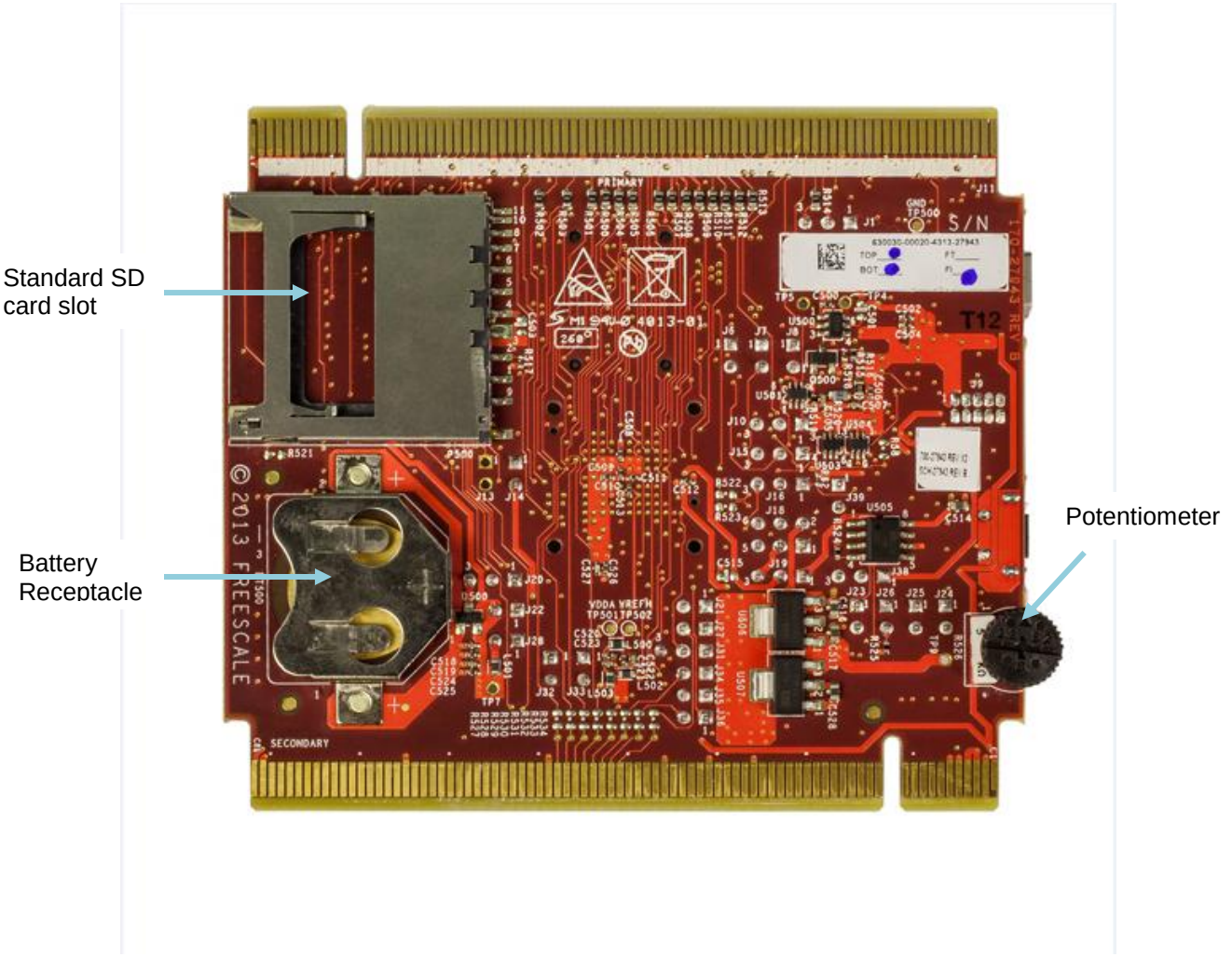


Figure-2 Back side of TWR-K64F120M

3.1.3 TWR-K64F120M jumper settings

Ensure that the jumpers are set as shown in this table. For more details, see the TWR-K64F120M User Manual.

Table-1 TWR-K64F120M jumper settings

Option	Jumper	Setting	Description
50 MHz Clock OSC power	J33	1-2	Enable V_BRD power supply to 50 MHz OSC
JTAG Board Power Selection	J14	OFF	Disconnect OSJTAG 5V output (P5V_TRG_USB) from JTAG port
3.3 V Voltage Regulator Input Selector	J18	1-2	Output of USB power switch controlled by the VTRG_EN signal from the K20 MCU. Provides input to 3.3 V regulator.
Board Power Selector	J29	1-2	Connect K20 USB regulator output (VOUT_3V3) to on-board supply (V_BRD)
MCU Power connection	J28	ON	Connect on-board 3.3 V or 1.8 V supply (V_BRD) to MCU VDD

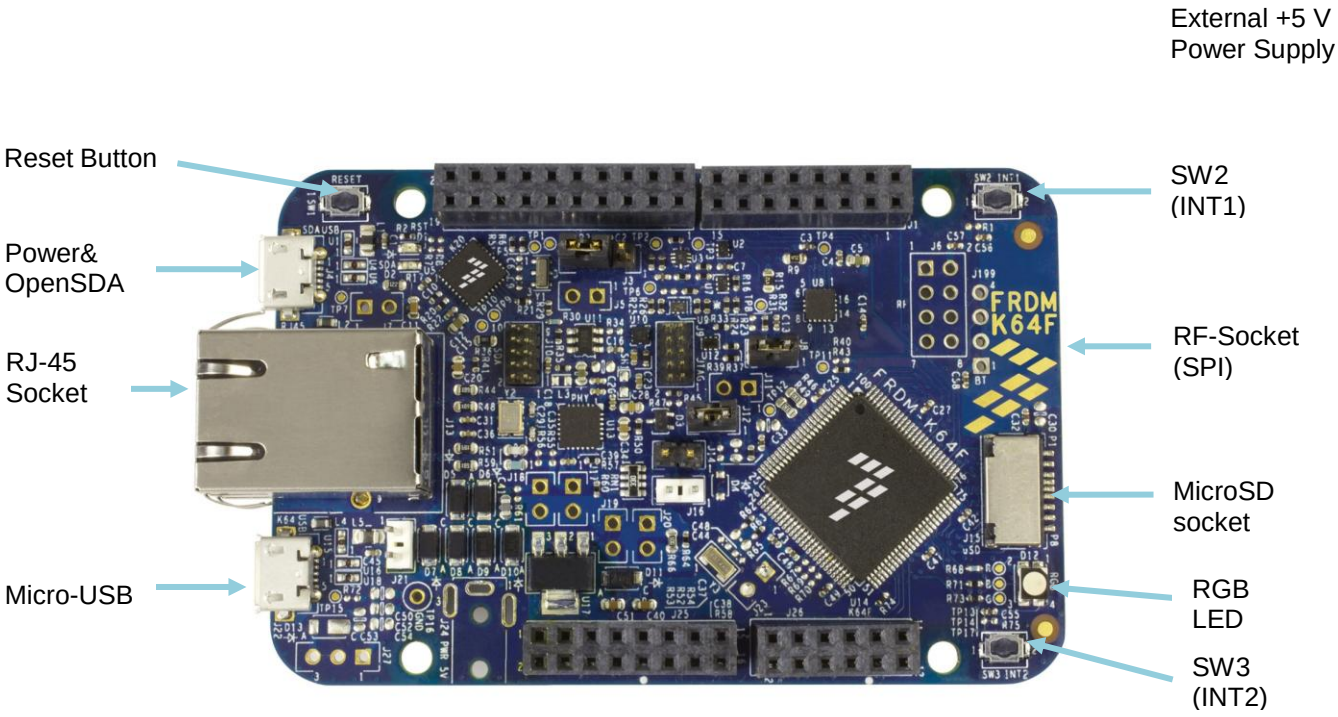
MCU Power VDDA for current measurement	J22	ON	Connect MCU_PWR (3.3 V or 1.8 V) to VDDA and VREFH
VBAT Power Source	J20	1-2	Connect VBAT to on-board 3.3 V or 1.8 V supply
LED connections	J30	1-2	Connect PTE6 to Yellow/Green LED (D5)
		3-4	Connect PTE7 to Yellow LED (D6)
		5-6	Connect PTE8 to Orange LED (D7)
		7-8	Connect PTE9 to Blue LED (D9)
OpenSDA (K20) UART Selection	J10	2-3	Connect OpenSDA (K20) UART TX to UART1_RX
	J15	2-3	Connect OpenSDA (K20) UART RX to UART1_TX
SWD_CLK_TGTMCU Output Selection	J39	ON	Enable the SWD_CLK_TGTMCU connection between the OpenSDA and target MCU

3.2 FRDM-K64F Introduction

3.2.1 FRDM-K64F features

- K64FN1M0VLL12 MCU (120 MHz, 1024 KB Flash, 260 KB RAM, low power, 100 LQFP package)
- Dual-role USB interface with Micro-AB USB connector
- On-board debug circuit: K20DX128VFM5 open (OpenSDAv2) with virtual serial port
- Six-axis sensor with integrated linear accelerometer and magnetometer (FXOS8700CQ)
- Tri-color user-controllable LEDs
- Two (2) user push-button switches for GPIO interrupts (SW2/SW3)
- RF and Bluetooth socket
- On-board Ethernet Physical Interface(KSZ8081)

3.2.2 FRDM-K64F first look



3.2.3 FRDM-K64F jumper settings

Ensure that the jumpers are set as shown in this table. For more details, see the FRDM-K64F User Manual.

Table-2 FRDM-K64 jumper settings

Option	Jumper	Setting	Description
Optional USB Host Functionality	J21	OFF	Disable USB Host functionality
VBAT Power Source	J23	OFF	Disconnect K64 3.3 V supply to VBAT
OpenSDAv2 debug source select	J11	1-2	Connect MCU with OpenSDAv2 debug interface
RST Push-Button Bypass	J3	1-2	Default setting for RST button
OpenSDA connections	J12	1-2	Connect OpenSDA DIO signal to target MCU
OpenSDA connections	J8	1-2	Connect OpenSDA CLK signal to target MCU
KSZ8081 interrupt connections	J14	OFF	Disconnect KSZ8081 interrupt signal to target MCU

4 Build and Run a KSDK Demo Application

This section describes the configuration process for the IAR Embedded Workbench and GCC ARM Embedded Tool to build, run, and debug demo applications provided in the Freescale KSDK.

The shell test demo application, targeted for the TWR-K64F120M hardware platform or FRDM-K64F board, is used as an example.

Note that the SDK libraries must be built before building a demo application. To build the SDK libraries, see Appendix A of this document.

4.1 IAR Embedded Workbench

4.1.1 Build a demo application

Because the platform driver library is already included in the lib folder of the demo application project, you can open the demo application project and build the demo applications directly whenever the platform_lib.a is ready.

Demo applications workspace files are located in:

```
<install_dir>/apps/<demo_name>/<compiler>/<board_name>/<demo_name>.eww
```

If the shell test demo application is used as an example, the IAR workspace file is located here:

```
<install_dir>/apps/shell_test/iar/twrk64f120m/shell_test.eww
```

Alternatively, it is located here:

```
<install_dir>/apps/shell_test/iar/frdmk64f120m/shell_test.eww
```

To build a demo application, click on the “Make” button:

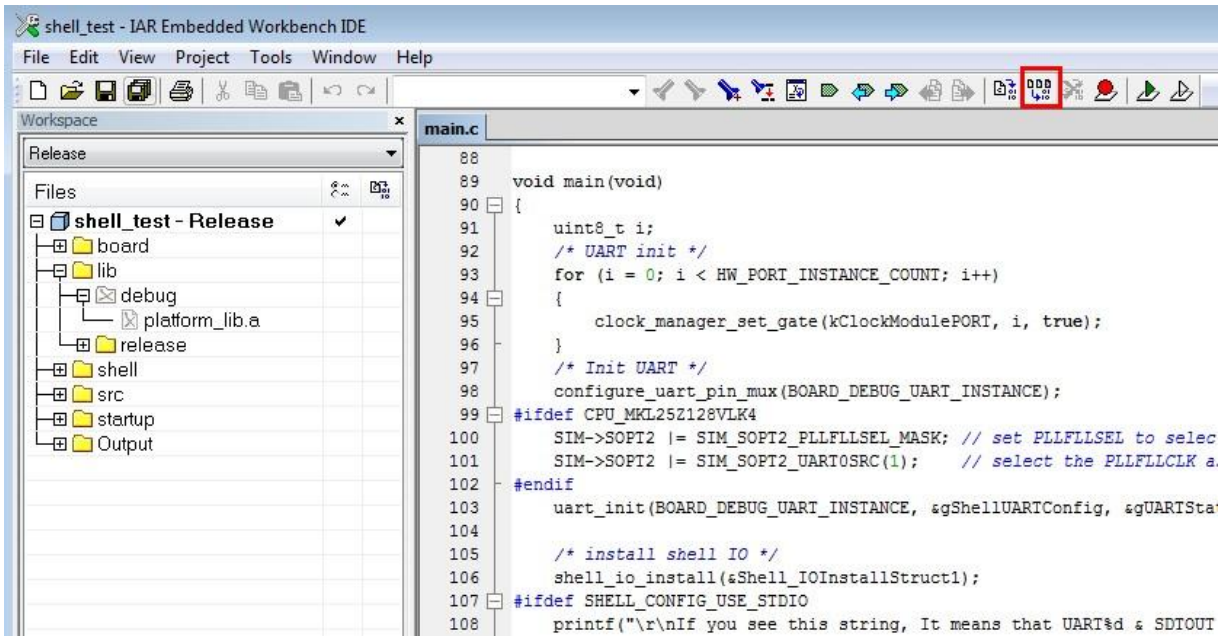


Figure-3 Build shell test demo application

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

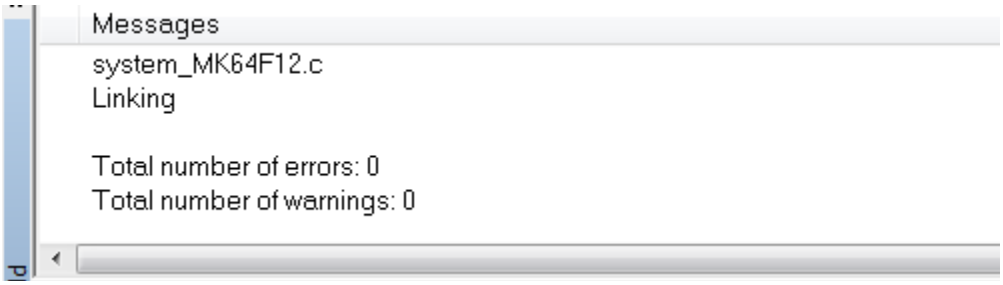


Figure-4 Build UART demo successfully

4.1.2 Run a demo application

Downloading and debugging KSDK demo applications in IAR Embedded Workbench is a standard process for all applications. Follow these steps to download and run the application:

1. Download and install the latest OpenSDA drivers from www.pemicro.com/opensda.
2. If your target board is the FRDM-K64F, follow these instructions to update your debugger firmware to match the KSDK demo applications. Otherwise, continue with step 3. **NOTE**: You may choose to continue using the factory installed CMSIS-DAP debugger firmware. If so, follow the instructions in Appendix B to update your demo applications to use the CMSIS-DAP firmware and you may skip to step 5. **However**, please be aware that there are potential issues when using the CMSIS-DAP firmware. For more information, see the *Kinetis SDK Release Notes* (document KSDKRN).
    a. First, ensure that the jumper, J25, is set to pins 1-2 (OpenSDA reset).

b. Press and hold the reset button while connecting the FRDM-K64F board to your PC via a USB A to micro B cable. The micro B connector should connect to the OpenSDA connector (J26). The green LED near the OpenSDA circuit (D2) should blink and the FRDM-K64F board will install a mass storage device as a boot loader, as shown.

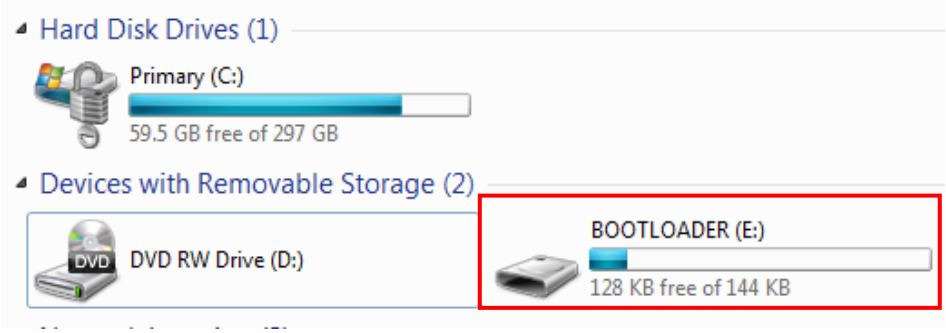


Figure-5 FRDM-K64F connected to PC in boot loader mode

c. Locate the PE Micro firmware binary, DEBUG_K64F_MBED_PEMICRO_V108.bin, in the root directory of your Kinetis SDK installation.

d. Copy the PE Micro firmware binary and paste it into the FRDM-K64F mass storage device.

e. After the file has been fully copied, the green LED near the OpenSDA circuit (D2) blinks more rapidly, and the boot loader mass storage device is removed from your PC system.

f. Unplug and plug back in the FRDM-K64F board without holding the reset button. The OpenSDA debugger should now be installed and you can continue.

3. After the successful installation of the OpenSDA driver, connect the OpenSDA USB connector, J2 for the TWR-K64F System module or J4 for the FRDM-K64F board, to the USB port on a PC.

4. Ensure that the debugger configuration is correct in the project options.

   The flash loader must be selected to support downloading the binary to the internal Flash:

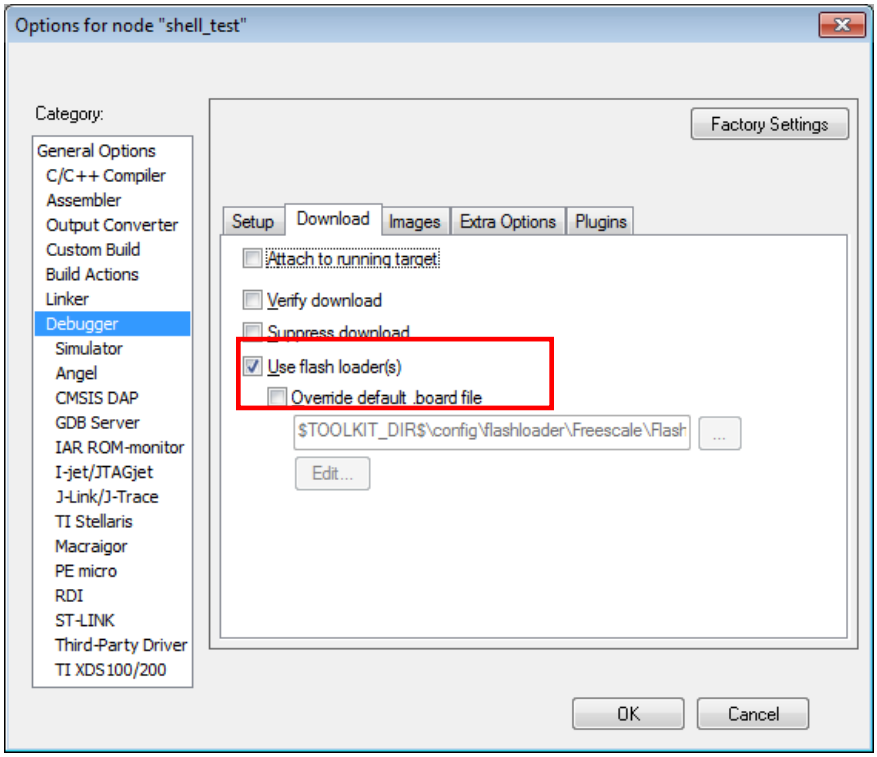


Figure-6 Flash loader configurations

PE micro driver is selected in the debugger setup by default for both TWR-K64F120M and FRDM-K64F. If you have the FRDM-K64F board with CMSIS-DAP debugger firmware installed, see Appendix B for the CMSIS-DAP debugger setup.

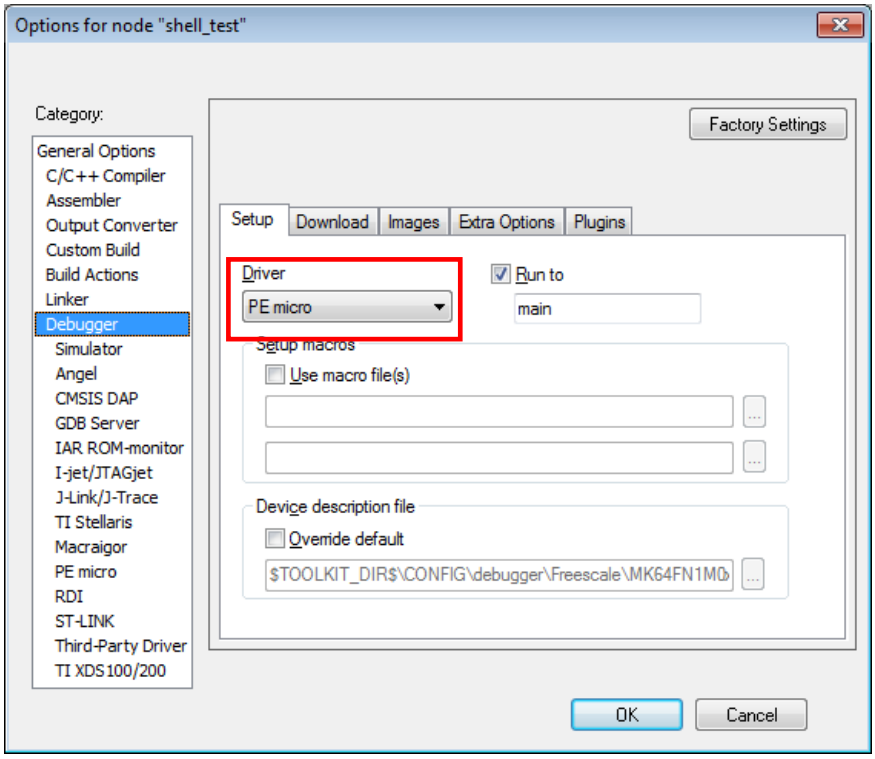


Figure-7 Debugger configurations for driver

OpenSDA-USB and SWD are configured in the PE micro settings as shown:

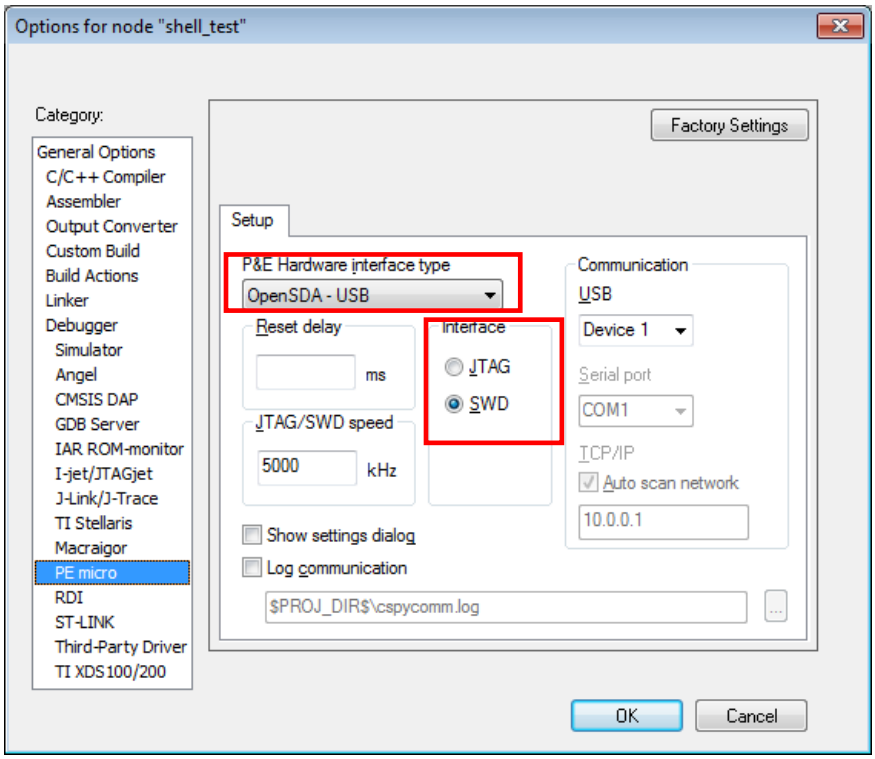


Figure-8 Debugger configurations for PE micro

5. Open the terminal application on a PC, such as PuTTy, and connect to the OpenSDA COM port number. At that point, Windows® operating system Device Manager shows the COM port number assigned to the OpenSDA.

This figure is an example which shows the OpenSDA serial port in the Device Manager . If you are using the FRDM-K64F factor installed firmware, it enumerates as an mbed serial port.

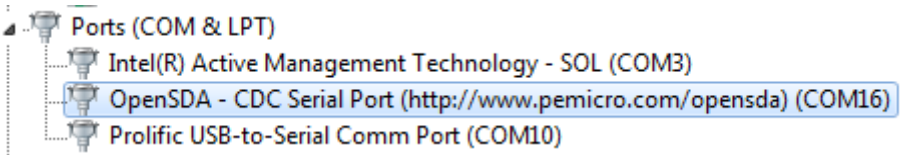


Figure-9 OpenSDA serial port in Device Manager

6. Configure the terminal settings as shown here:
    a. 115200 baud rate
    b. No parity
    c. 8 data bits
    d. 1 stop bit

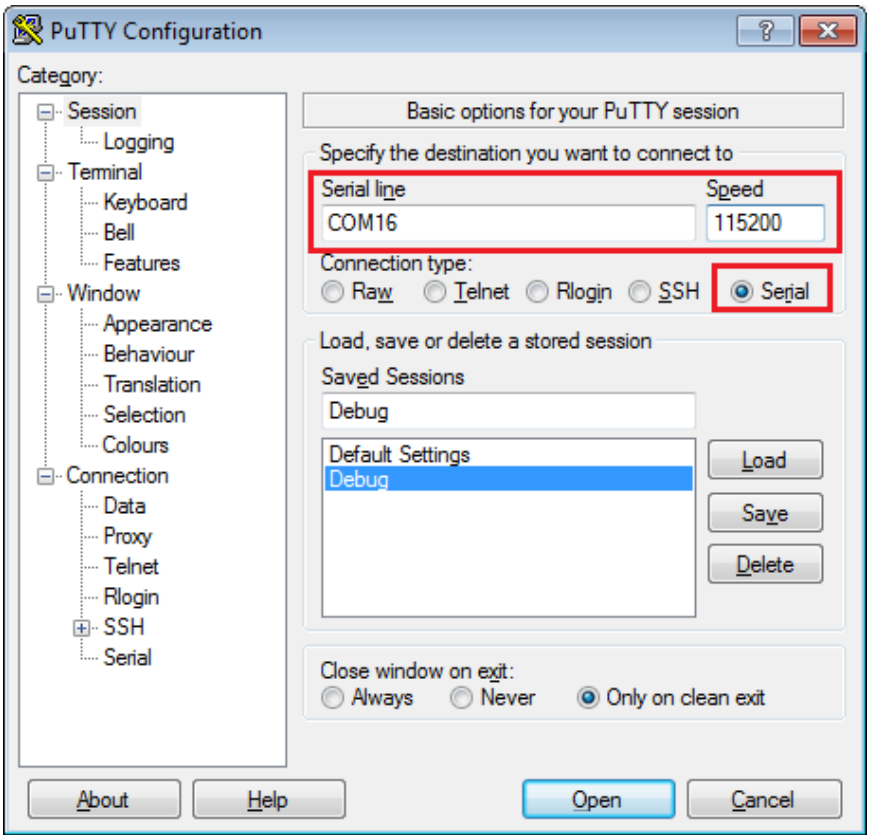


Figure-10 Terminal (PuTTY) configurations

7. After the terminal has successfully connected, click on the "Download and Debug" button to download the application to the target device.

Figure-11 Download and Debug Button

8. Next, the debugger stops executing at the start of the main() function:

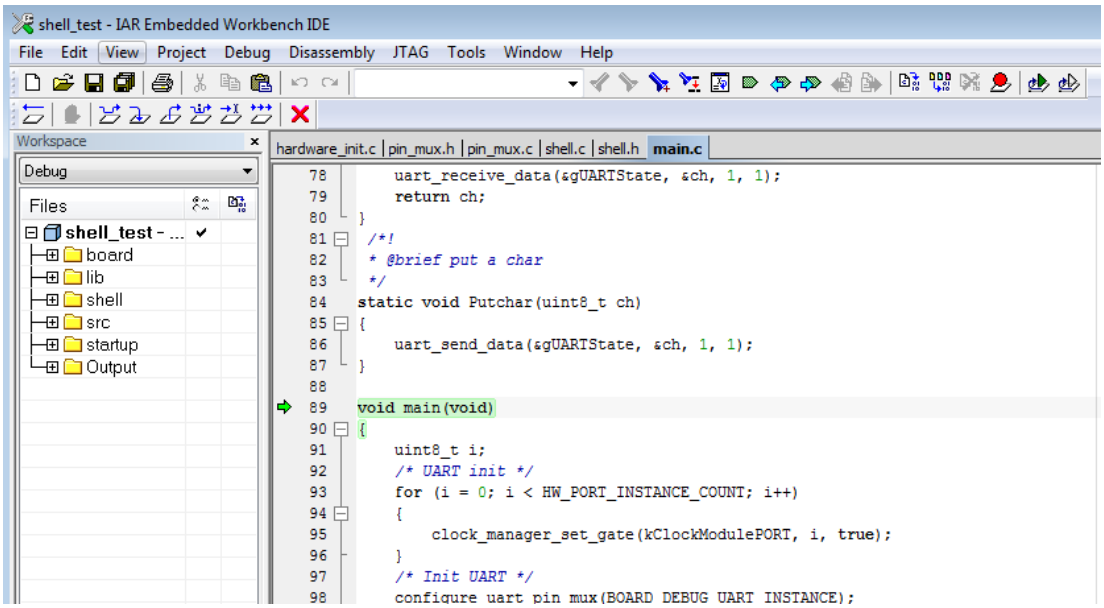


Figure-12 Stop at main() when run debugging

You can resume application execution by clicking on the “Go” button:

Figure-13 Go Button

9. The shell test application should now be running and the following banner should be displayed on the terminal. If this is not the case, please double check your terminal settings and terminal connections.

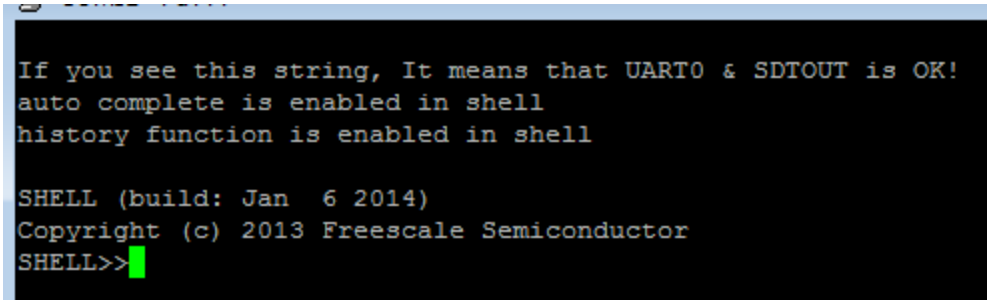


Figure-14 Shell test demo running

For example, type help and it displays the command menu:

```
SHELL>>help
history (0)- print history
help    (1)- print command description/usage
test    (2)- help - app test function
SHELL>>
```

Figure-15 Help command

4.2 GCC ARM Embedded Tool

4.2.1 Environment Setup

4.2.1.1 Install GCC ARM v4.7 2013q3 embedded toolchain

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

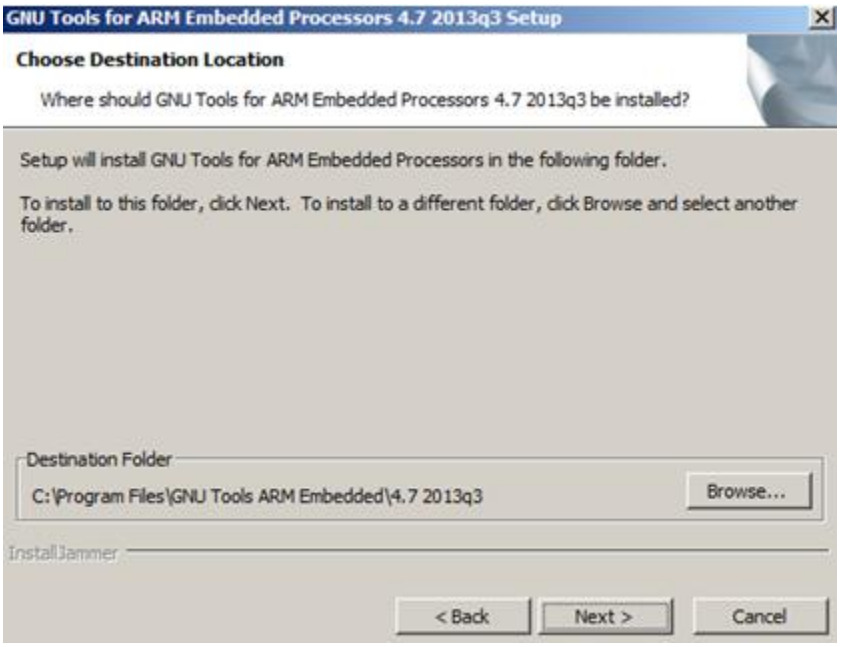


Figure-16 Install GCC ARM embedded toolchain

4.2.1.2 Install MinGW and MSYS

1. Download the MinGW installer, which is located here.
2. Run the 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-17 MinGW Installer.

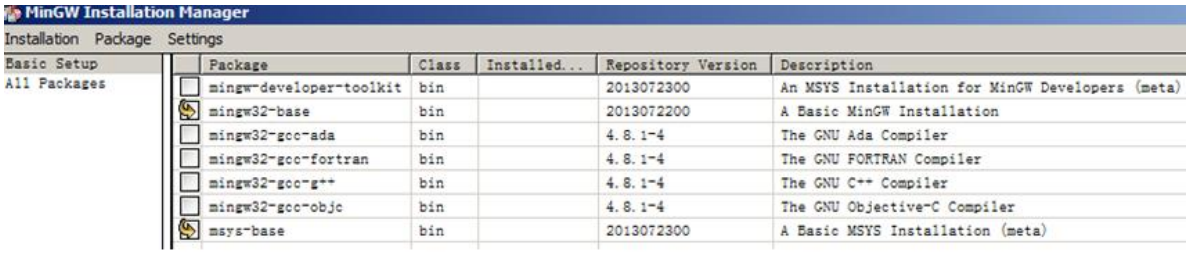


Figure-17 MinGW Installer

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

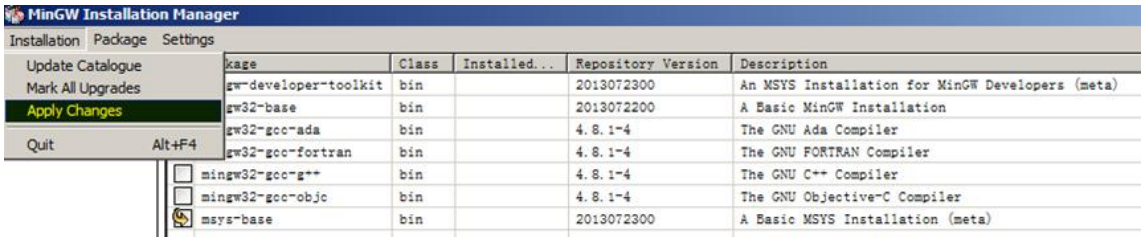


Figure-18 MinGW Installer Apply Change

4.2.1.3 Configure system environment

- Update the system environment variable "Path" to include the MINGW installation folder, such as the <drive>\MINGW\msys\1.0\bin;<drive>:\MINGW\bin.
  Run the GCC Command prompt in Start->All Programs-> GNU Tools ARM Embedded 4.7 2013q3.

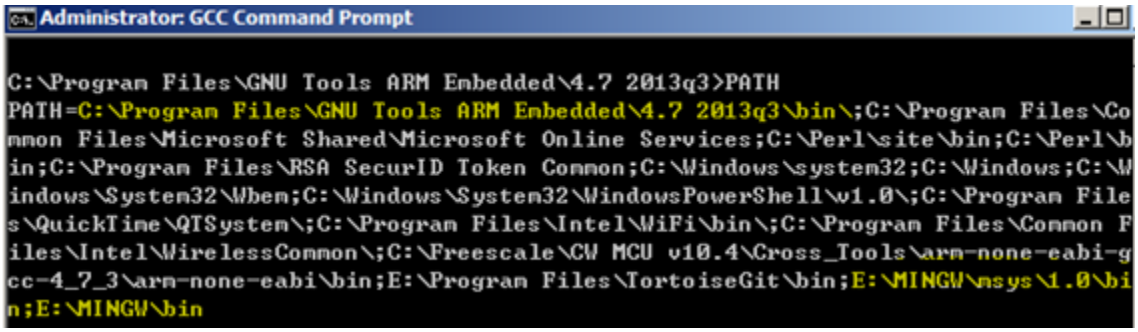


Figure-19 PATH environment

- Open the file <KSDK Install dir >/mk/common.mk and replace the GCC_TOOLCHAIN_DIR to your GCC ARM toolchain path in the short file name. You can convert the short file name from a long file name by using this command:

```
for %A in ("C:\Program Files\GNU Tools ARM Embedded\4.7 2013q3") do @echo %~sA
```

4.2.2 Build the platform driver library

Before building and debugging any demo application in KSDK, the driver library project should be built to generate the library archives: platform_lib.a. Because this library contains all binary codes for HAL and the peripheral drivers specific to the chip, each SoC has its own platform.a library archives.

To build the platform library, change the current directory in GCC Command prompt to:

```
<install_dir>/lib/gcc/<device_name>/platform_lib/
For example: <install_dir>/lib/gcc/K64F12/platform_lib/
```

Run the command mingw32-make build=debug or mingw32-make build=release.

When the build is complete, the platform_lib.a is generated in the directory according to the build target:

```
Debug - <install_dir>/lib/gcc/<device_name>/output/Debug
Release - <install_dir>/lib/gcc/<device_name>/output/Release
```

Demo applications use the platform_lib.a to call the HAL and peripheral driver functions.

4.2.3 Build a demo application

Change the directory in GCC command prompt to this:

```
<install_dir>/apps/shell_test/gcc/twrk64f120m
```

Alternatively, change the directory in the GCC command prompt to this:

```
<install_dir>/apps/shell_test/gcc/frdmk64f120m
```

Run the mingw32-make build=debug target=flash command.

When the build is complete, the shell_test.elf and the shell_test.bin are generated in this path:

```
<install_dir>/apps/shell_test/gcc/ twrk64f120m /output/Flash_Debug
```

Alternatively, they are generated in this path:

```
<install_dir>/apps/shell_test/gcc/frdmk64f120m/output/Flash_Debug
```

4.2.4 Run a demo application

This section describes how to download and debug the applications by using J-Link.

Note

To use an external debugger, such as J-Link, you may need to disconnect the OpenSDA SWD from the K64. To do this on the FRDM-K64F board, cut the shorting trace which connects the pins of jumper holes on connectors J8 and J12.

You can debug the application by using Eclipse with CDT or command line. This example shows how to use the GDB command line.

- Download and install J-Link software & documentation package for Windows operating system here. You may need a J-Link serial number to download that software.
- K64 is now supported in the Segger GDB server. Setup the GDB server by running the GDB server JLinkGDBServer.exe.

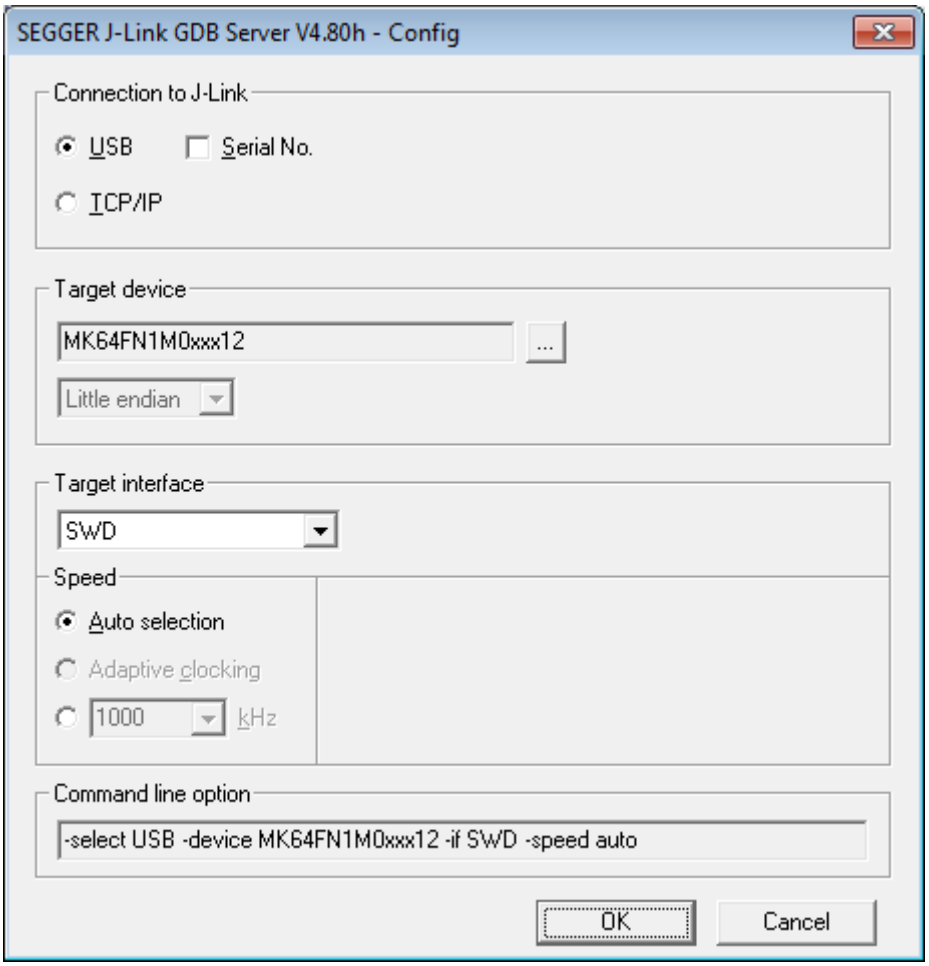


Figure-20 J-Link GDB Server GUI

- Select the connection options for your board. For K64, select the MK64FN1M0xxx12 device as the Target device. After configuration, click on the "OK" button to connect to the board.

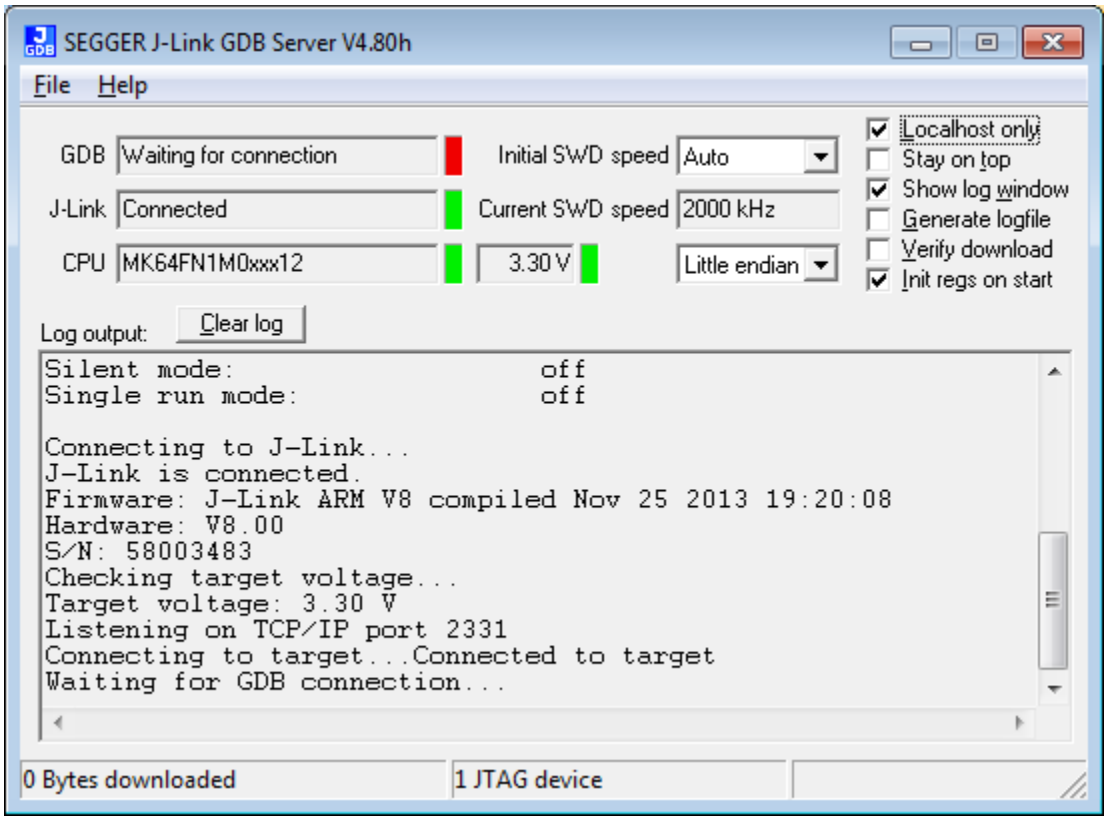


Figure-21 JLinkGDBServer connection

- Download and debug from command line

  Change the directory like this:

```
<install_dir>/apps/shell_test/gcc/frdmk64f120m/output/Flash_Debug
```

  Alternatively, change the directory like this:

```
<install_dir>/apps/shell_test/gcc/frdmk64f120m/output/Flash_Debug
```

- Run the arm-none-eabi-gdb to start the GDB. After the GDB is launched, run these commands to load and start the application:

```
(gdb) target remote localhost:2331
Remote debugging using localhost:2331
0x00000000 in __isr_vector ()
(gdb) monitor reset
Resetting target
(gdb) monitor halt
(gdb) monitor load
(gdb) load
Loading section .text, size 0x495c lma 0x0
Loading section .ARM_exidx, size 0x8 lma 0x495c
Loading section .data, size 0xac lma 0x4964
Start address 0x8d1, load size 18960
Transfer rate: 1089 KB/sec, 4740 bytes/write.
(gdb) monitor reg pc = (0x00000004)
Writing register (PC = 0x000008D1)
(gdb) monitor reg sp = (0x00000000)
Writing register (SP = 0x1FFF2588)
(gdb) monitor go
(gdb)
```

Figure-22 Run arm-none-eabi-gdb

- Debugging

  Use the GDB general debug command to do debug. To quit the debugging, use the monitor halt to halt the target CPU, then quit the debug.

Appendix A: Build the platform driver library

Before building and debugging any demo application in KSDK, the driver library project should be built to generate the library archive: **platform_lib.a**. Because this library contains all binary codes for HAL and the peripheral drivers specific to the chip, each SoC has its own platform.a library archive.

To build the platform library, open the workspace file in IAR. The platform driver library project is located in:

```
<install_dir>/lib/iar/<device_name>
```

The workspace file is named lib.eww:

```
<install_dir>/lib/iar/<device_name>/lib.eww
```

The project file is named platform_lib.ewp:

```
<install_dir>/lib/iar/<device_name>/platform_lib.ewp
```

To build the platform driver library for the K64, open the workspace file in IAR:

```
< install _dir>/lib/iar/K64F12/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 compiler optimization is set to maximum. The debug information is not generated. This target should be used for the final application release.

Note:

Code is downloaded to Flash instead of RAM in both Debug and Release configurations.

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

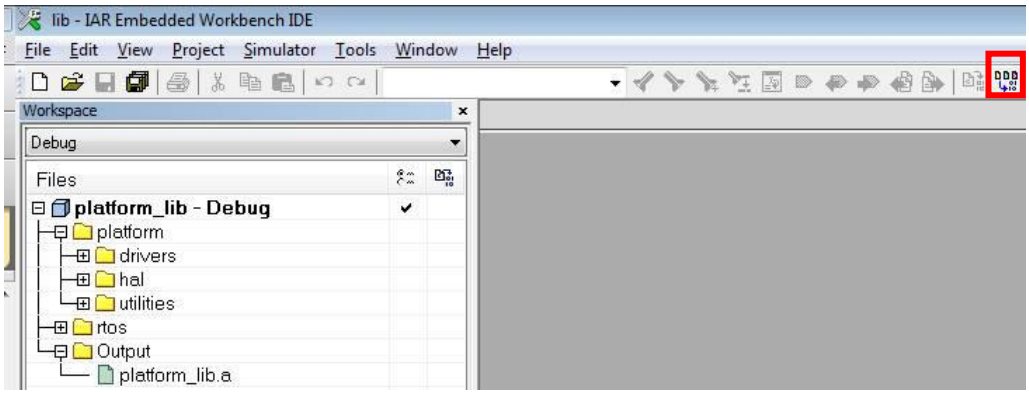


Figure-23 Platform driver library build

When the build is complete, the platform_lib.a is generated in the directory according to the build target:

```
Debug - <install_dir>/lib/iar/<device_name>/output/Debug
Release - <install_dir>/lib/iar/<device_name>/output/Release
```

Demo applications use the platform_lib.a to call the functions of the HAL and the peripheral drivers.

Appendix B: Configure the CMSIS-DAP Debugger

If you have the CMSIS-DAP debugger firmware on your FRDM-K64F board, please follow the steps here to setup the debugger environment:

1. Download and install the latest CMSIS-DAP drivers from: mbed.org/handbook/Windows-serial-configuration#1-download-the-mbed-windows-serial-port. Note that only the Windows operating system needs drivers. The mbed drivers work by default on Linux and Mac systems.
2. After the successful installation of the CMSIS-DAP driver, connect the CMSIS-DAP USB connector, J4 on the FRDM-K64F board, to the USB port on a PC.
3. Open the terminal application on a PC, such as PuTTy, and connect to the mbed COM port number.
4. Configure the Debugger in IAR as below:
   - CMSIS-DAP is selected in the debugger setup.

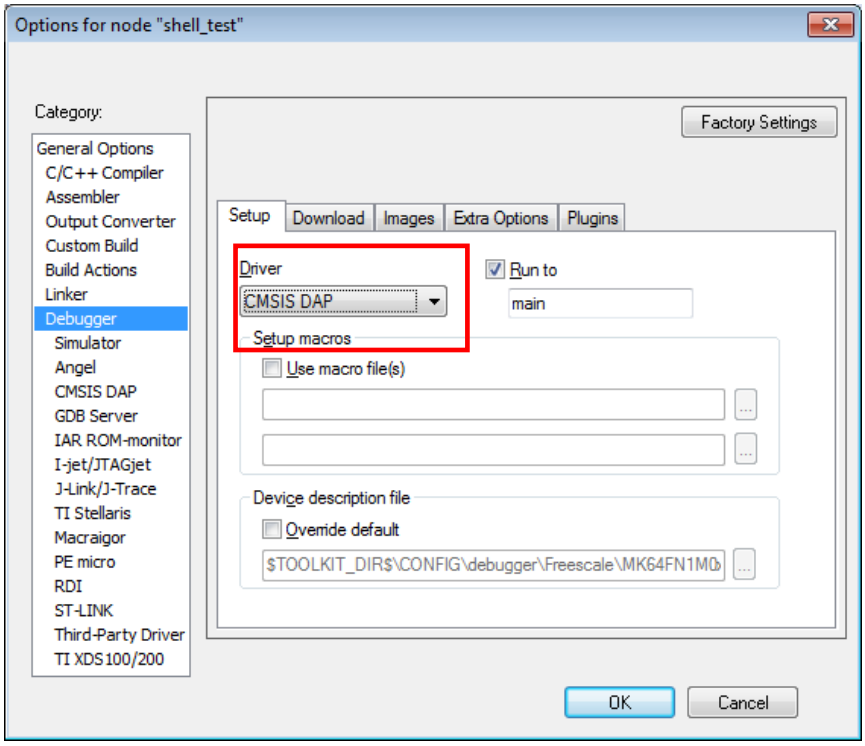


Figure-24 CMSIS-DAP Debugger configurations

   - The JTAG/SWD settings of the debugger are configured in the CMSIS-DAP category under the JTAG/SWD tab and are shown below.

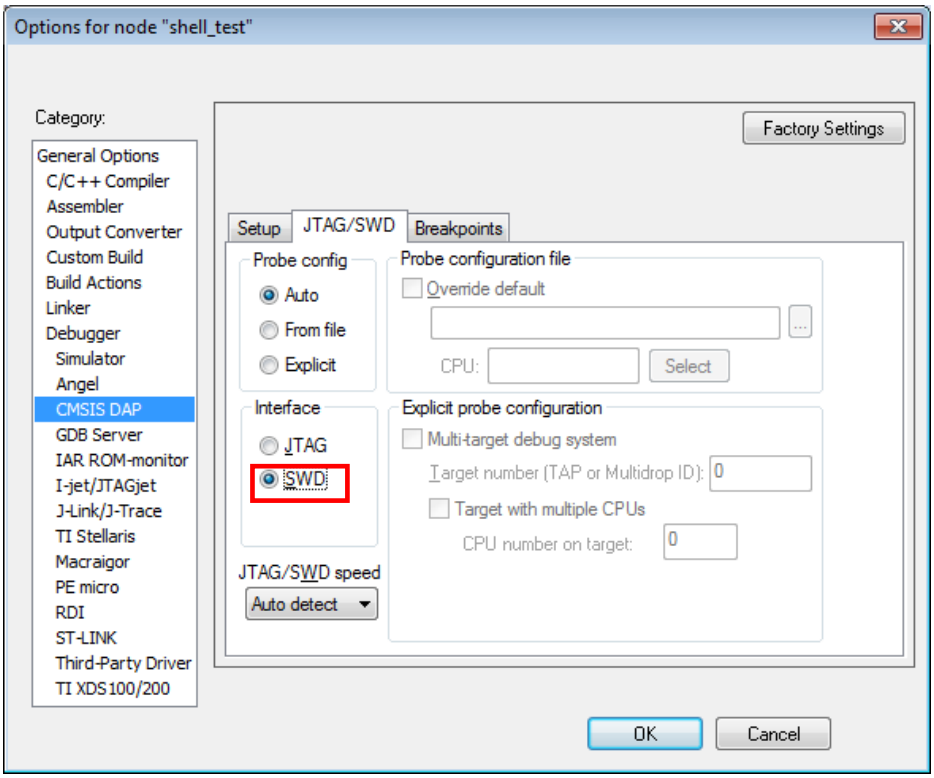


Figure-25 CMSIS-DAP JTAG/SWD configurations

5 Revision history

This table summarizes revisions to this document.

Revision History	
Location	**Change description**
Entire document	Initial release – 1.0.0-beta

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KSDKK64UG
Rev. #1.0.0-beta
03/2014