## Creating a New *FreeRTOS for KSDK* Project in KDS

**By: Technical Information Center**

This document explains how to create a new FreeRTOS for KSDK project from scratch.

The tools used in this guide are KSDK1.2, KDS3.0 and FRDM-K64.

### Requirements – DO NOT Skip

#### Install KDS (Kinetis Design Studio), you can download from [www.freescale.com/kds](http://www.freescale.com/kds)

#### Install KSDK (Kinetis Software Development Kit), you can download from [www.freescale.com/ksdk](http://www.freescale.com/ksdk)

#### Install ***‘KSDK\_1.2.0\_Eclipse\_Update’*** you can find the update in ***‘C:\Freescale\KSDK\_1.2.0\tools\eclipse\_update’***. The instruction to make the updates are described in chapter 2 and 2.1 of ***‘C:\Freescale\KSDK\_1.2.0\doc\rtos\mqx\MQX RTOS IDE Guides\MQX-KSDK-KDS-Getting-Started.pdf’***

#### It is necessary to build freertos library:

##### Platform Library for FreeRTOS, ‘***libksdk\_platform\_freertos.a’***

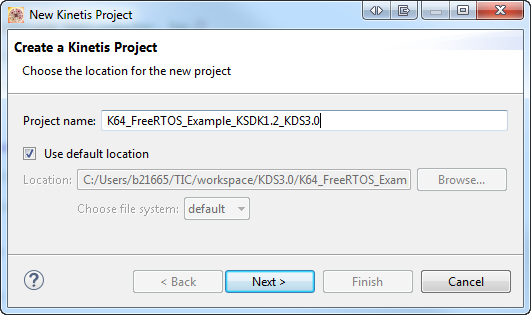
###### This project is located in ***${KSDK\_PATH}/lib/ksdk\_freertos\_lib/kds/<mcu>***

### Create a new FreeRTOS for KSDK Project from Scratch

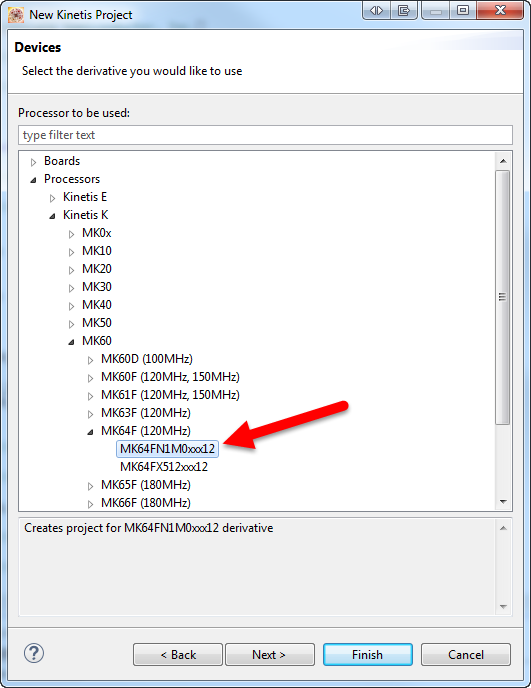
#### Go to ***menu File > New > Kinetis Project***.



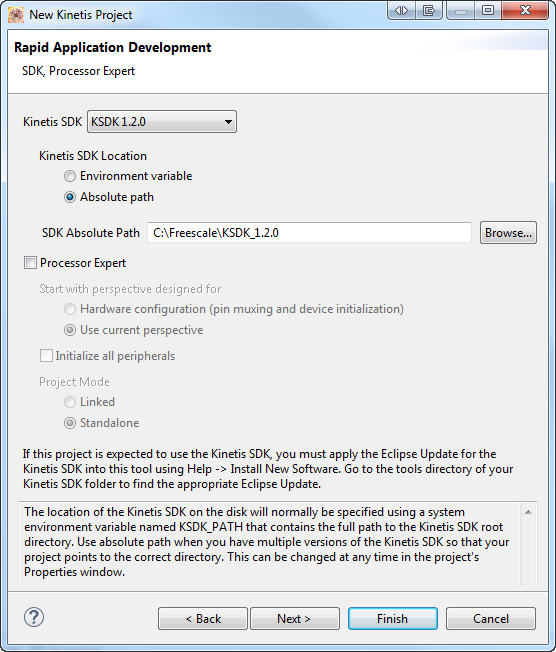
#### Write a name for your project and click ***‘Next’***.



#### Select your target and click ‘**Next’.**



#### Select ***‘KSDK 1.2.0’*** to include KSDK support and be sure that the SDK Absolute Path is correct. Then click ***‘Finish’***. If you have any problem at this point make sure that de KSDK update is installed, please refer to step 1.3 in this document



#### Create 3 folders with the following names.

* Board
* FreeRTOS
* Utilities

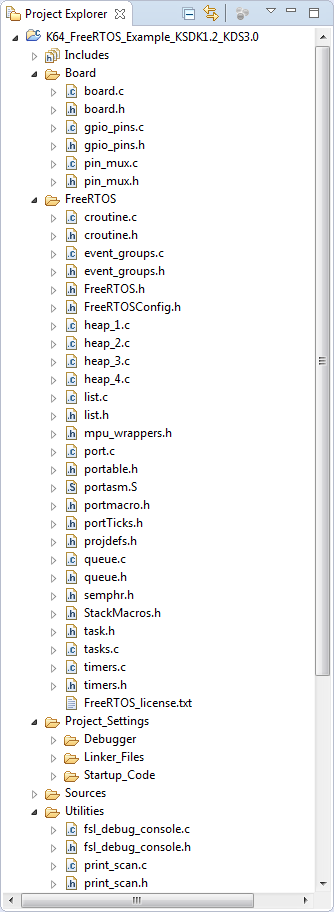
#### Then erase **‘SDK’** folder which is created in the default project.

#### 

#### Add the following files to each of the folders. You can just drag and drop from Windows Explorer.

* **Board**
  + C:\Freescale\KSDK\_1.2.0\examples\<board>
    - All source files (\*.c and \*.h). DO NOT copy .peb file.
* **FreeRTOS**
  + C:\Freescale\KSDK\_1.2.0\rtos\FreeRTOS\src
    - All files
  + C:\Freescale\KSDK\_1.2.0\rtos\FreeRTOS\include
    - All files
  + C:\Freescale\KSDK\_1.2.0\rtos\FreeRTOS\port\gcc
    - All files
  + C:\Freescale\KSDK\_1.2.0\rtos\FreeRTOS\config\<mcu>\gcc
    - FreeRTOSConfig.h
* **Utilities**
  + C:\Freescale\KSDK\_1.2.0\platform\utilities\src
    - fsl\_debug\_console.c
    - print\_scan.c
    - print\_scan.h
  + C:\Freescale\KSDK\_1.2.0\platform\utilities\inc
    - fsl\_debug\_console.h

The final folder tree should look as image below.



#### Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Includes.**

#### Erase all the content and replace it with the following paths. You can copy and paste the paths below.

It is possible to add all the paths at once, please refer to the post in the following link:

<http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/>

../Board

../FreeRTOS

../Utilities

../Sources

../Project\_Settings/Startup\_Code

C:\Freescale\KSDK\_1.2.0\platform\osa\inc

C:\Freescale\KSDK\_1.2.0\platform\utilities\inc

C:\Freescale\KSDK\_1.2.0\platform\CMSIS\Include

C:\Freescale\KSDK\_1.2.0\platform\devices

C:\Freescale\KSDK\_1.2.0\platform\devices\MK64F12\include

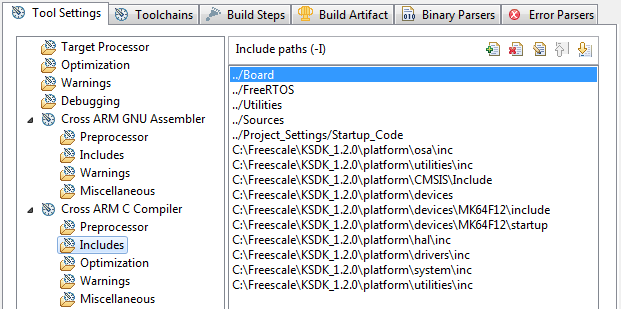
C:\Freescale\KSDK\_1.2.0\platform\devices\MK64F12\startup

C:\Freescale\KSDK\_1.2.0\platform\hal\inc

C:\Freescale\KSDK\_1.2.0\platform\drivers\inc

C:\Freescale\KSDK\_1.2.0\platform\system\inc

C:\Freescale\KSDK\_1.2.0\platform\utilities\inc



#### Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > Libraries** and add FreeRTOS platform library and its search path.

ksdk\_platform\_freertos

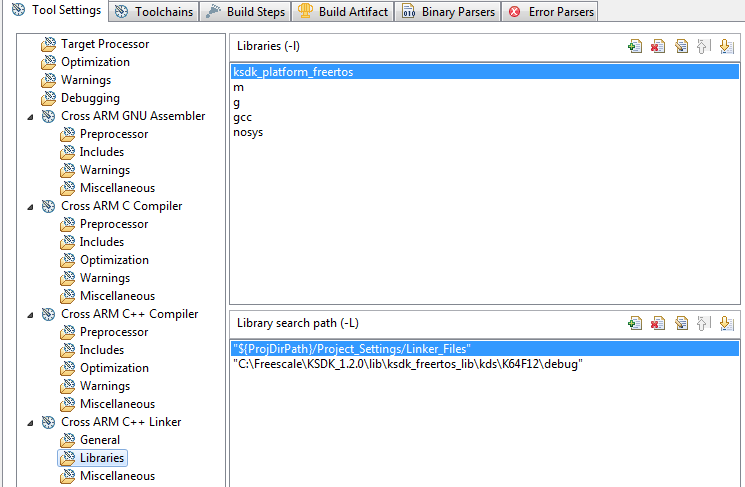
m

g

gcc

nosys

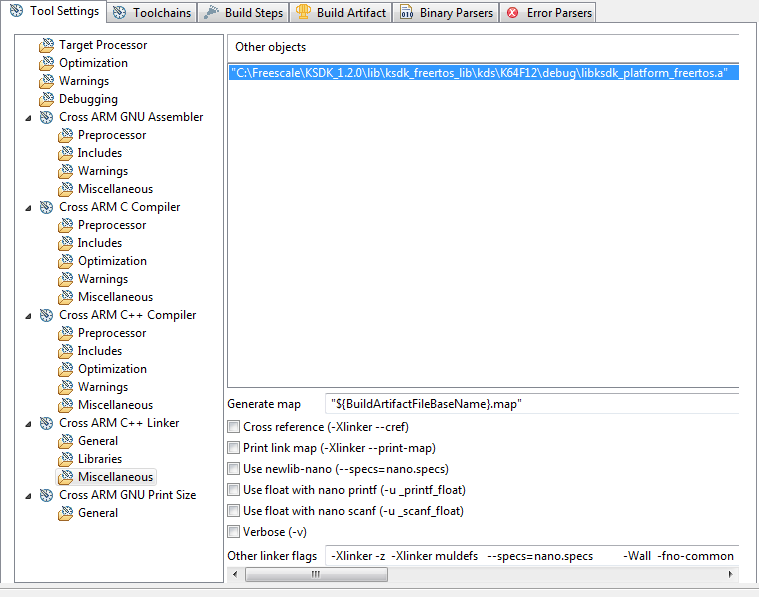
C:\Freescale\KSDK\_1.2.0\lib\ksdk\_freertos\_lib\kds\K64F12\debug



#### Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > Miscellaneous** and add FreeRTOS library with its full path and then replace the flags in **‘Other linker flasgs’** box with the flags below.

C:\Freescale\KSDK\_1.2.0\lib\ksdk\_freertos\_lib\kds\K64F12\debug\libksdk\_platform\_freertos.a

-Xlinker -z -Xlinker muldefs --specs=nano.specs -Wall -fno-common -ffunction-sections -fdata-sections -ffreestanding -fno-builtin -Os -mapcs -Xlinker -static -Xlinker --defsym=\_\_stack\_size\_\_=0x2000 -Xlinker --defsym=\_\_heap\_size\_\_=0x2000 -Xlinker --defsym=\_\_ram\_vector\_table\_\_=1

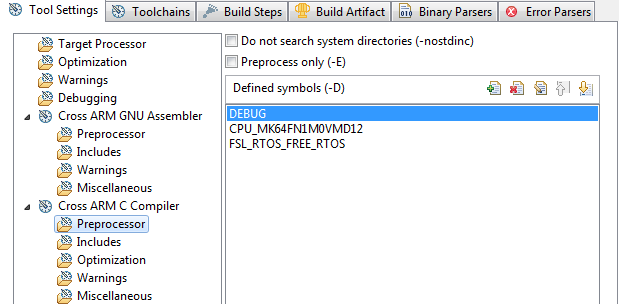


#### Go to menu **Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Preprocessor** and add/replace the following definitions.

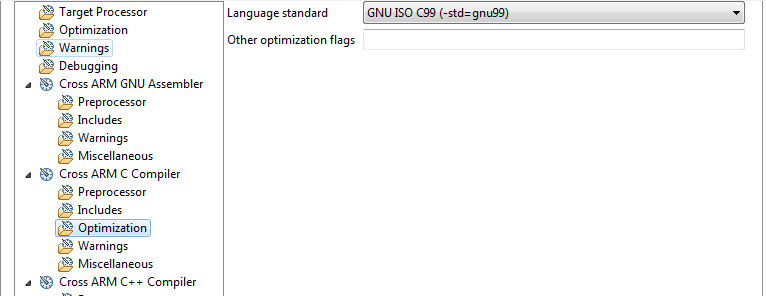
DEBUG

CPU\_MK64FN1M0VMD12

FSL\_RTOS\_FREE\_RTOS

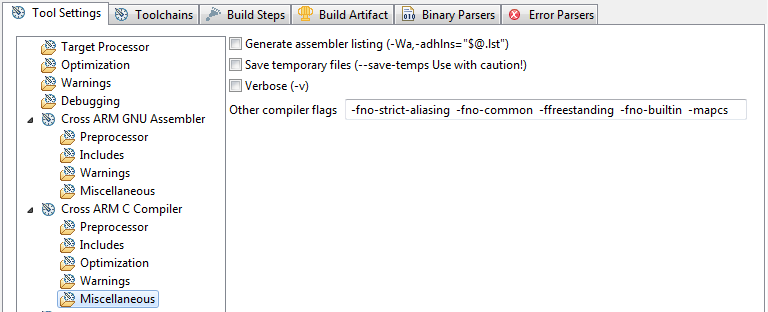


#### Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Optimization** and select **GNU ISO C99 (-std=gnu99)** as ‘Language standard’.



#### Go to menu **Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Miscellaneous** and add the following flags in **‘Other compiler flags’**, the click **‘Apply’** and **‘OK’**.

-fno-strict-aliasing -fno-common -ffreestanding -fno-builtin -mapcs



#### Erase the code in ***‘main.c’*** and replace it with the code below.

**#include** "board.h"

**#include** "pin\_mux.h"

**#include** "fsl\_clock\_manager.h"

**#include** "fsl\_debug\_console.h"

**#include** "fsl\_os\_abstraction.h"

///////////////////////////////////////////////////////////////////////////////

// Definitions

///////////////////////////////////////////////////////////////////////////////

// task priority

**#define** TASK\_ONE\_PRIO 6U

// task size

**#define** TASK\_ONE\_STACK\_SIZE 1536U

// task define

OSA\_TASK\_DEFINE(task\_one, TASK\_ONE\_STACK\_SIZE);

///////////////////////////////////////////////////////////////////////////////

// Prototypes

///////////////////////////////////////////////////////////////////////////////

**void** **task\_one**(task\_param\_t param);

///////////////////////////////////////////////////////////////////////////////

// Code

///////////////////////////////////////////////////////////////////////////////

**int** **main**(**void**)

{

osa\_status\_t result = *kStatus\_OSA\_Error*;

hardware\_init();

//set up the basic RTOS services

OSA\_Init();

//create task

result = OSA\_TaskCreate(task\_one,

(uint8\_t \*)"task\_one",

TASK\_ONE\_STACK\_SIZE,

task\_one\_stack,

TASK\_ONE\_PRIO,

(task\_param\_t)0,

false,

&task\_one\_task\_handler);

//validate task creation

**if**(result != *kStatus\_OSA\_Success*)

{

PRINTF("Failed to create task one\n\r");

}

//start the RTOS scheduler

OSA\_Start();

**while**(1){} //Should not reach this point

**return** 0;

}

**void** **hardware\_init**(**void**)

{

/\* enable clock for PORTs \*/

CLOCK\_SYS\_EnablePortClock(PORTA\_IDX);

CLOCK\_SYS\_EnablePortClock(PORTB\_IDX);

CLOCK\_SYS\_EnablePortClock(PORTC\_IDX);

CLOCK\_SYS\_EnablePortClock(PORTE\_IDX);

/\* Init board clock \*/

BOARD\_ClockInit();

dbg\_uart\_init();

// Initializes GPIO driver

GPIO\_DRV\_Init(switchPins, ledPins);

}

/\*!

\* Task one

\*

\* Send message and blinks LED

\*/

**void** **task\_one**(task\_param\_t param)

{

PRINTF("This is Task One\n\n\r");

PRINTF("Press SW2 to Toggle Blue LED...\n\n\r");

**while**(1)

{

// Poll sw2 in FRDM-K64

**if**(GPIO\_DRV\_ReadPinInput(*kGpioSW2*) == 0) **//sw# depends on the board that is used**

{

OSA\_TimeDelay(200);

GPIO\_DRV\_TogglePinOutput(BOARD\_GPIO\_LED\_BLUE);

}

}

}

////////////////////////////////////////////////////////////////////////////////

// EOF

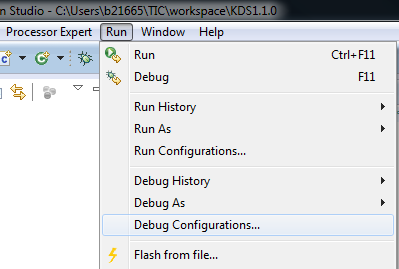
////////////////////////////////////////////////////////////////////////////////

### Run the application

#### Build your application, go to ***menu Project > Build Project***. Alternately click the hammer button.

#### 

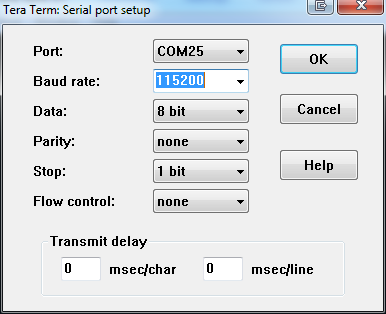
#### Go to **menu Run > Debug Configurations…**



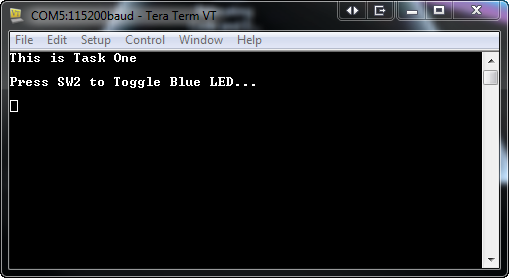
#### Select the ***‘Debug Configuration’*** that matches your connection type, in this example **P&E Micro** connection is used, if you don’t know which your connection type is or you want to change your connection type see ‘***Appendix A’*** at the end this document. Once you double click the appropriate ***‘Debug Configuration’***, the connection settings will appear. In ***‘Debugger’*** tab select the right ***‘Interface’***, ***‘Port’*** and ***‘Device Name’***, then click ***‘Apply’*** and ***‘Debug’***.

#### 

#### Open a terminal, select the appropriate port and set baudrate to 115200.



#### Run the application, you will see the following message in terminal.



**APPENDIX A: Connection Types**

* KDS works with devices which support OpenSDAv2 connection.
* You can find Open SDA User’s Guide here: <http://www.freescale.com/files/32bit/doc/user_guide/OPENSDAUG.pdf>
* You can learn more about Open SDA in the following link: <https://community.freescale.com/docs/DOC-100720>

**Identify your Connection Type**

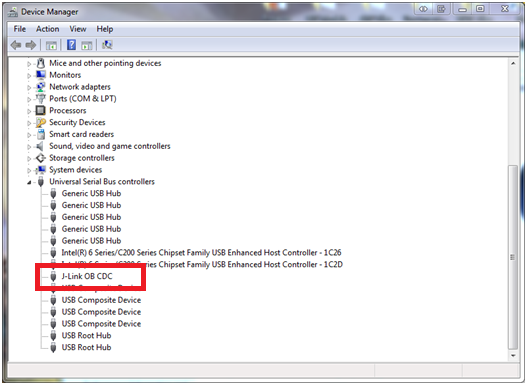
To find out which your connection type is you must connect your device to your computer and go to Windows Device Manager, here you can see the connection used by your device. You can see how to open Windows device manager in the link below:

<http://windows.microsoft.com/en-us/windows/open-device-manager#1TC=windows-7>

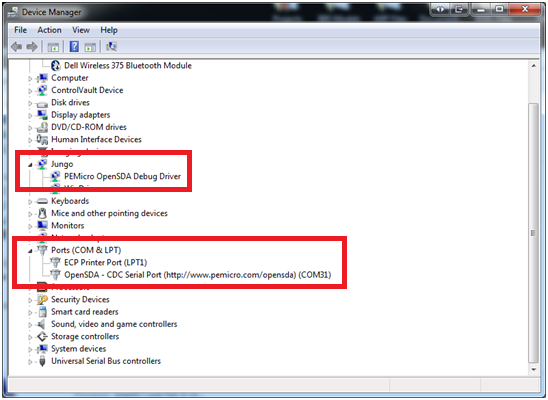
**MBED Connection**

Please note that this connection is not supported in KDS yet.

**Segger J-link connection**

****

**P&E Micro Connection**

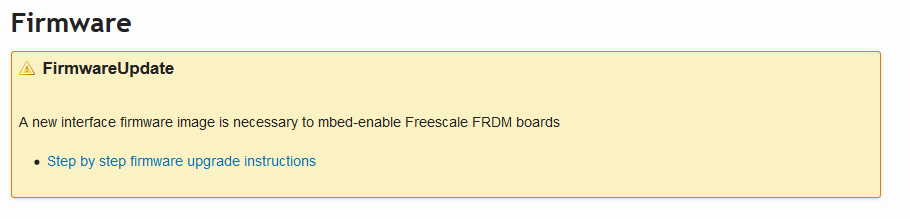
****

**Switching or updating your connection firmware**

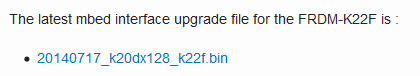
You can download different versions of OpenSDA form our partners’ web sites.

**MBED**

1. Go to <http://mbed.org/platforms/>
2. Select your platform
3. Click on the ***‘Step by step firmware update instructions’*** link

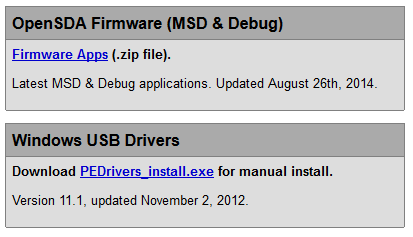


1. Save the latest firmware and follow the instructions to do the update



**P&E Micro**

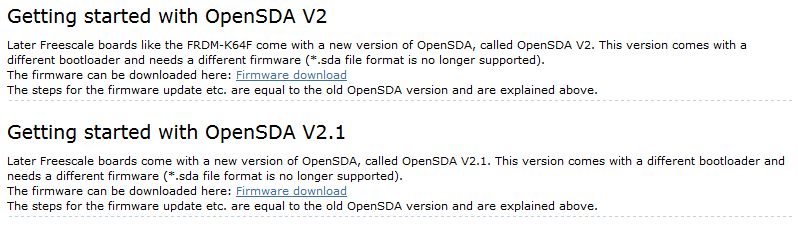
1. Go to <http://www.pemicro.com/opensda/>
2. Download ‘***Open SDA Firmware’*** and optionally ‘***Windows USB Drivers’***



1. Extract the content on the .zip file and follow steps in ‘***Updating the OpenSDA Firmware.pdf’***

**Segger**

1. Go to <http://www.segger.com/opensda.html>
2. Download the required firmware



1. Unzip the content of the .zip file and use the binary file to update the firware. Steps to update the firware are shown in Open SDA User’s Guide mentioned at the beginning of this appendix.

**Other useful links**

**CMSIS DAP**

<https://mbed.org/handbook/CMSIS-DAP>

**Binary Files for the mbed Bootloader with Eclipse and GNU ARM Eclipse Plugins**

<http://mcuoneclipse.com/2014/04/20/binary-files-for-the-mbed-bootloader-with-eclipse-and-gnu-arm-eclipse-plugins/>

**Segger J-Link Firmware for OpenSDAv2**

<http://mcuoneclipse.com/2014/04/27/segger-j-link-firmware-for-opensdav2/>

**FRDM-K22F: Debugging with Segger J-Link OpenSDAv2.1 Firmware**

<https://community.freescale.com/docs/DOC-101790>

**FRDM-K22F: Debugging with P&E OpenSDAv2.1 Firmware**

<https://community.freescale.com/docs/DOC-101792>

**OpenSDA Update Instructions for Freescale Freedom Development Boards for Windows 8.1 and Linux**

<http://www.element14.com/community/docs/DOC-65460/l/opensda-update-instructions-for-freescale-freedom-development-boards-for-windows-81-and-linux>

**P&E Eclipse Update Site for GNU ARM Eclipse Plugins**

<http://mcuoneclipse.com/2014/09/11/pe-eclipse-update-site-for-gnu-arm-eclipse-plugins/>