Creating a New *MQX RTOS for KSDK* Project in KDS

By: Technical Information Center

As there is not a New MQX Project Wizard it is necessary to create a project from scratch. This document explains 2 ways to create a new MQX for KSDK project.

- Create a copy from an example project.
    - **IMPORTANT: The clue is to edit all the include paths to make the project independent from its original location in the Examples folder.**
- Create a new project from scratch and add all the necessary files and edit all the necessary settings
    - At the end this project must look the same as a copy of an example.

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

1 Requirements – DO NOT Skip

1.1 Install KDS (Kinetis Design Studio), you can download from www.freescale.com/kds

1.2 Install KSDK (Kinetis Software Development Kit), you can download from www.freescale.com/ksdk

1.3 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'***

1.4 It is necessary to build the following libraries:

1.4.1 Platform Library for MQX, '***libksdk_platform_mqx.a'***

- This project is located in ***${KSDK_PATH}/lib/ksdk_mqx_lib/kds/<mcu>***

1.4.2 MQX Library, ***'lib_mqx.a'***

- This project is located in ***"${KSDK_PATH}/rtos/mqx/mqx/build/kds/mqx_<board>"***

1.4.3 MQX Standard Library, ***'lib_mqx_stdlib.a'***

- This project is located in ***"${KSDK_PATH}/rtos/mqx/mqx_stdlib/build/kds/mqx_stdlib_<board>"***

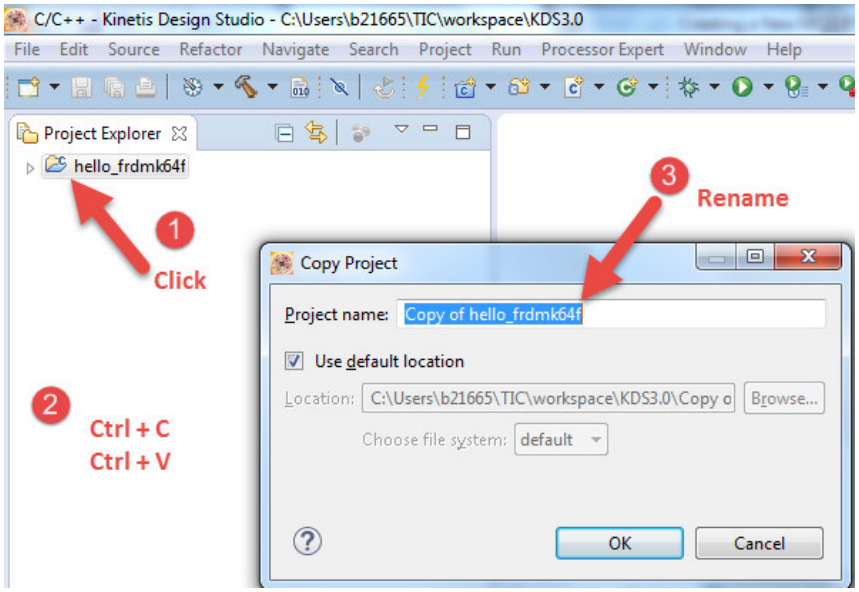
NOTE: For information about building KSDK and MQX libraries in KDS please see chapter *'**3 Building MQX RTOS example and libraries**'* of ***'MQX-KSDK-KDS-Getting-Started.pdf'*** located in MQX for KSDK installation path ***C:\Freescale\KSDK_1.2.0\doc\rtos\mqx\MQX RTOS IDE Guides***

2 Create a Copy from an Example Project

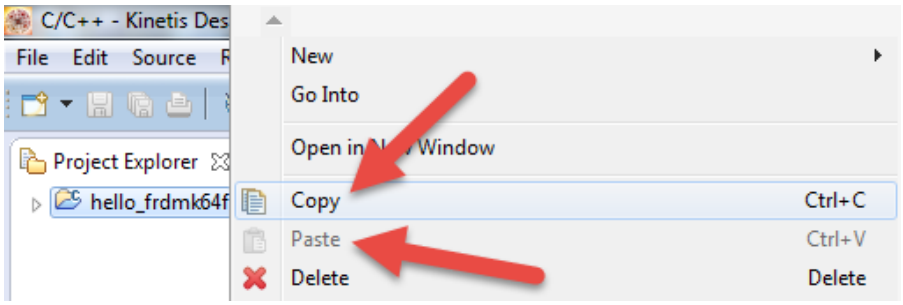
2.1 Go to ***menu File > Import*** and browse for the **hello_frdmk64f** located in the following path: ***C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\examples\hello\build\kds\hello_frdmk64f***

Please refer to chapter '**3 Building MQX RTOS example project and libraries'** of ***'MQX-KSDK-KDS-Getting-Started.pdf'*** for instructions to import a project. This file is located in ***C:\Freescale\KSDK_1.2.0\doc\rtos\mqx\MQX RTOS IDE Guides***

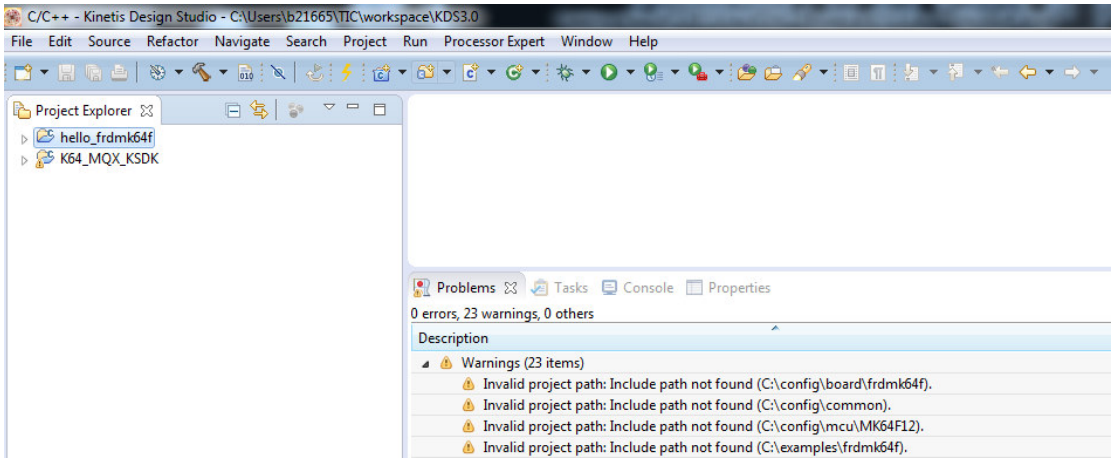
2.2 Once the **hello_frdmk64f** example is imported into the workbench you can just copy-paste it using **Ctrl + C** and **Ctrl + V** keys and you will be prompted to set a new name.



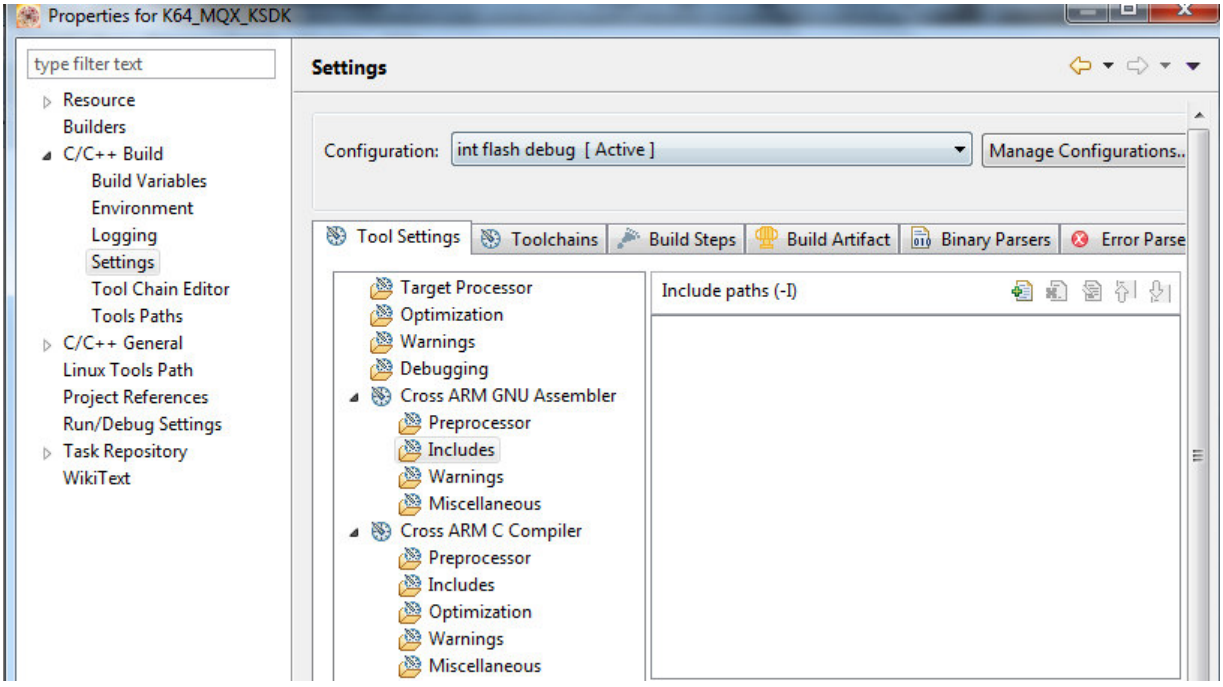
Alternately you can right click on the project and select **'Copy'** and **'Paste'** from context menu.



2.3 The new project will be created and many warnings will be shown. This is because the new project's location is your current workspace and the project's include paths are relative to the original location of the **hello_frdmk64f** example.



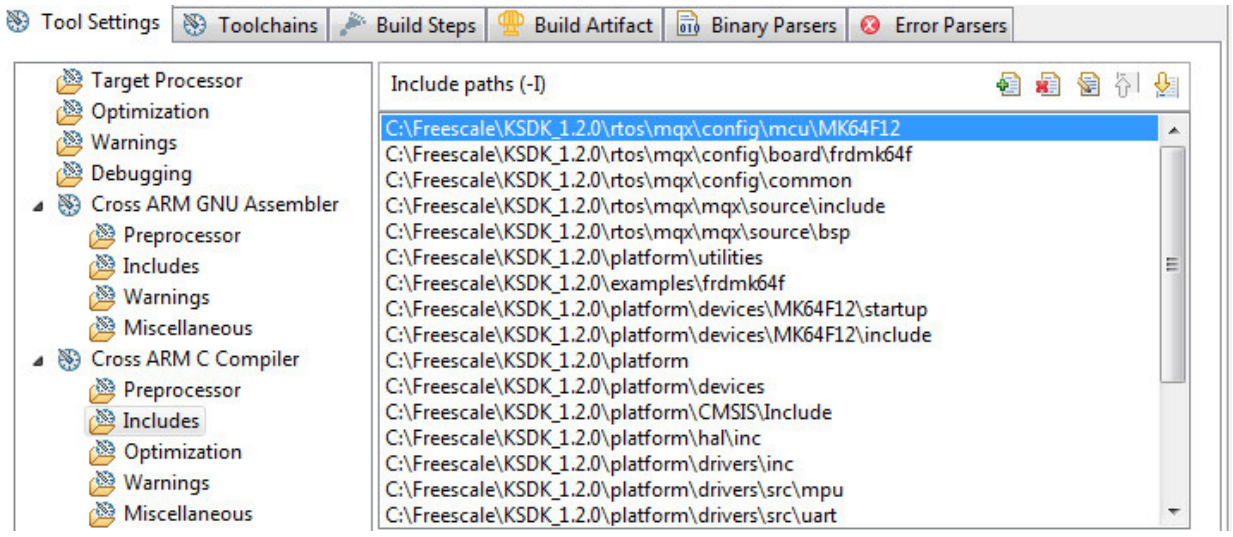
2.4 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM GNU Assembler > Includes** and delete all the paths. These paths are not used in this example.



2.5 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Includes** and fix all the paths by setting absolute paths. You can copy and paste the paths below.

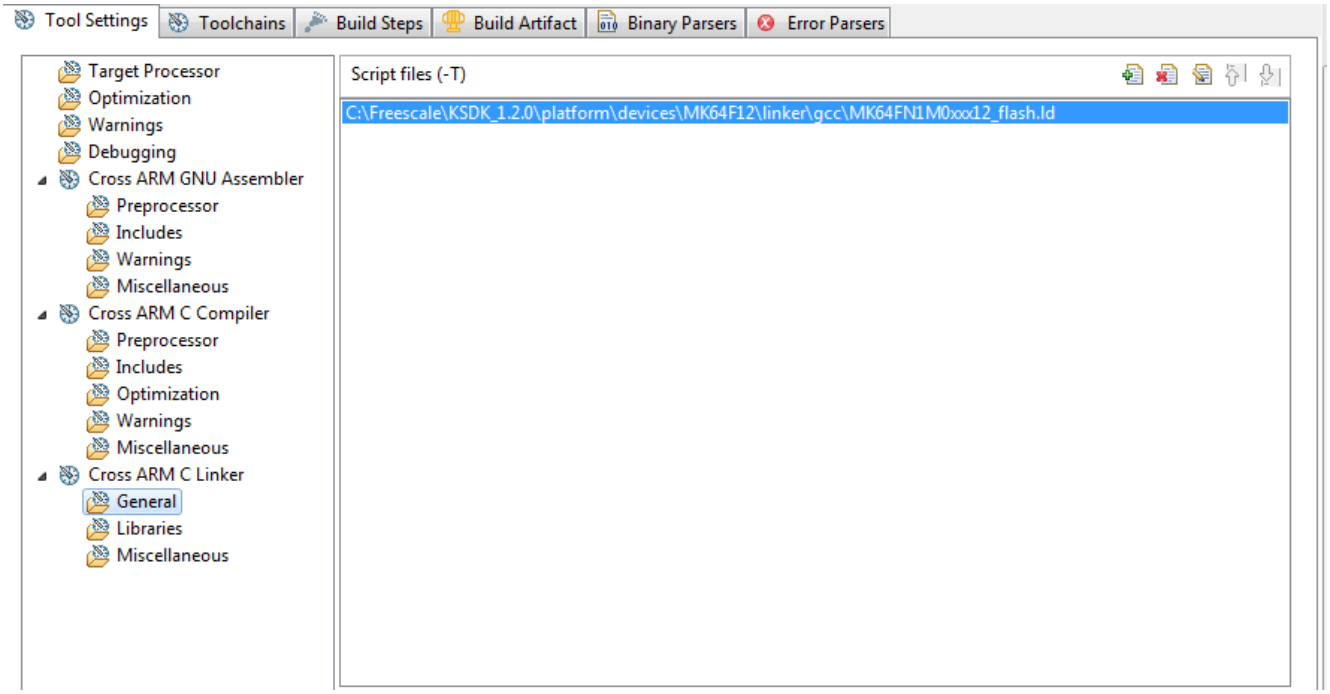
It is possible to add all the paths at the same time, please refer to the post in the following link: http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/

```
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\mcu\MK64F12
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\board\frdmk64f
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common
C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\include
C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp
C:\Freescale\KSDK_1.2.0\platform\utilities\src
C:\Freescale\KSDK_1.2.0\platform\utilities\inc
C:\Freescale\KSDK_1.2.0\examples\frdmk64f
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\startup
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\include
C:\Freescale\KSDK_1.2.0\platform
C:\Freescale\KSDK_1.2.0\platform\devices
C:\Freescale\KSDK_1.2.0\platform\CMSIS\Include
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\drivers\src\mpu
C:\Freescale\KSDK_1.2.0\platform\drivers\src\uart
C:\Freescale\KSDK_1.2.0\platform\utilities\inc
C:\Freescale\KSDK_1.2.0\platform\osa\inc
C:\Freescale\KSDK_1.2.0\platform\system\inc
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\config
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib
```



2.6 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C Linker > General** and fix the linker file path by setting the absolute path.

C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\linker\gcc\ MK64FN1M0xxx12_flash.ld

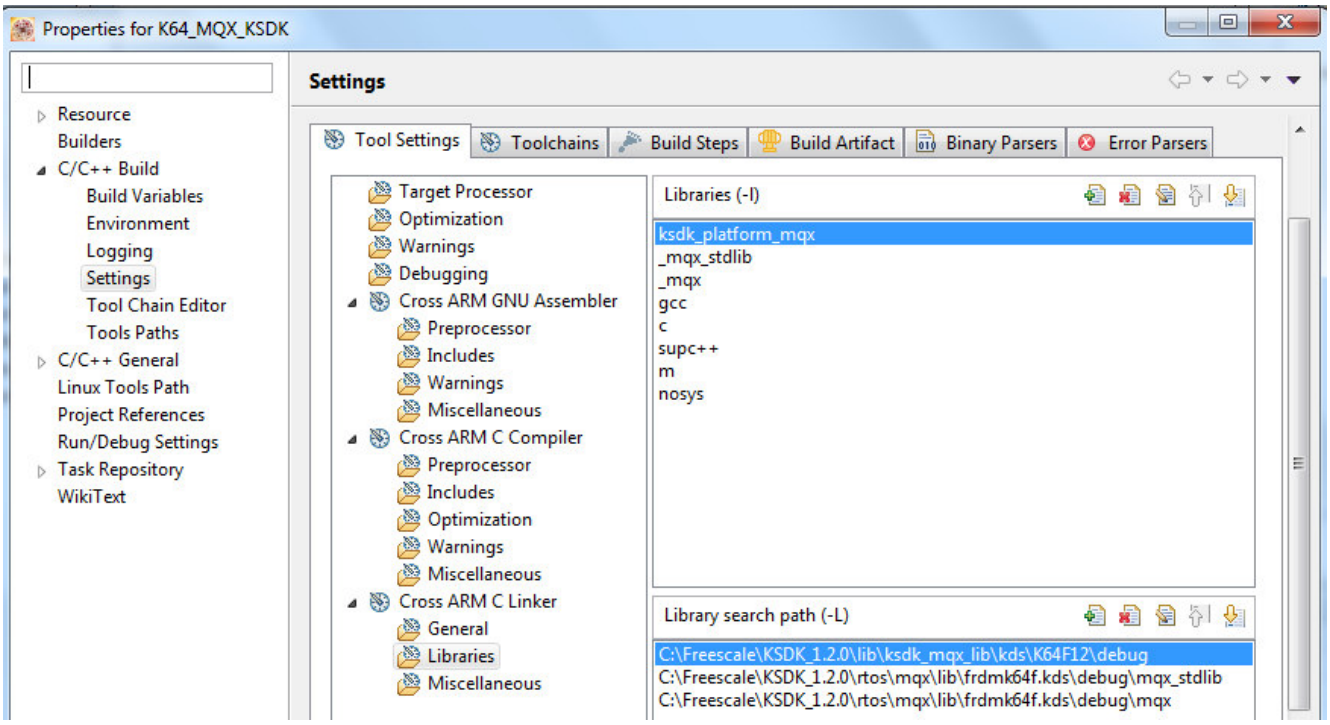


NOTE: If you plan to edit the linker file it is strongly recommended to make a copy of the .ld file because all the new projects and example projects will be affected by the .ld changes.

2.7 Go to menu **Project > Properties > C/C++ Build > Settings > Cross ARM C Linker > Libraries** and fix all the Libraries search paths by setting the absolute paths.

It is possible to add all the paths at the same time, please refer to the post in the following link:
http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/

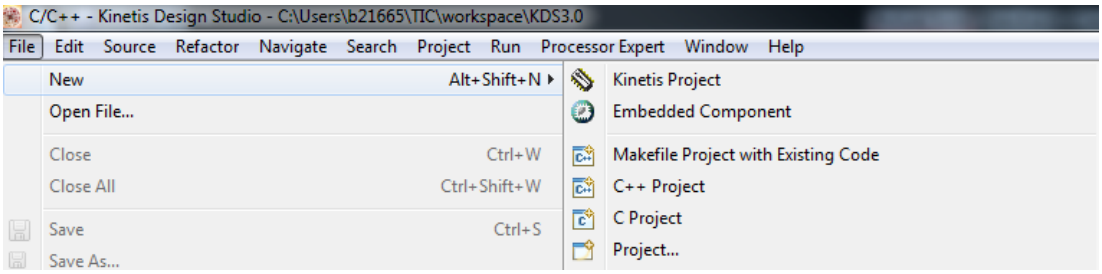
C:\Freescale\KSDK_1.2.0\lib\ksdk_mqx_lib\kds\K64F12\debug
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx



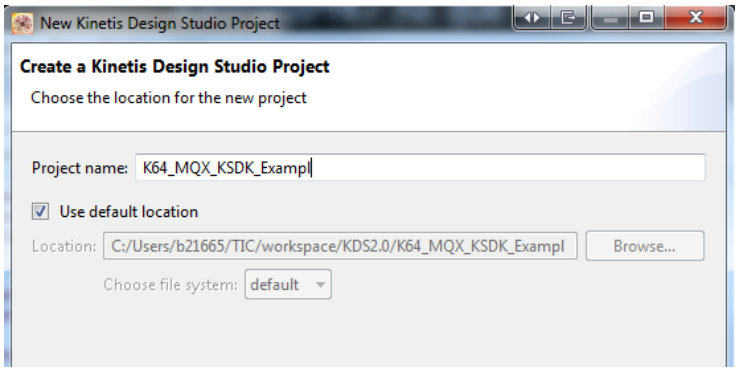
2.8 Now the new project is completely self-dependent and ready to run. If you have any problem at this point make sure that de KSDK update is installed, please refer to step 1.3 in this document.

3 Create a new MQX RTOS for KSDK Project from Scratch

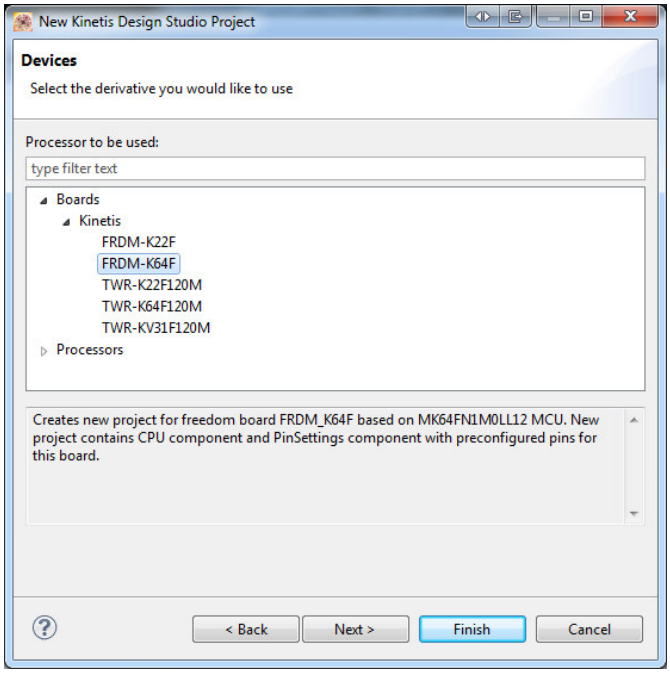
3.1 Go to ***menu File > New > Kinetis Design Studio Project***.



3.2 Write a name for your project and click ***'Next'***.

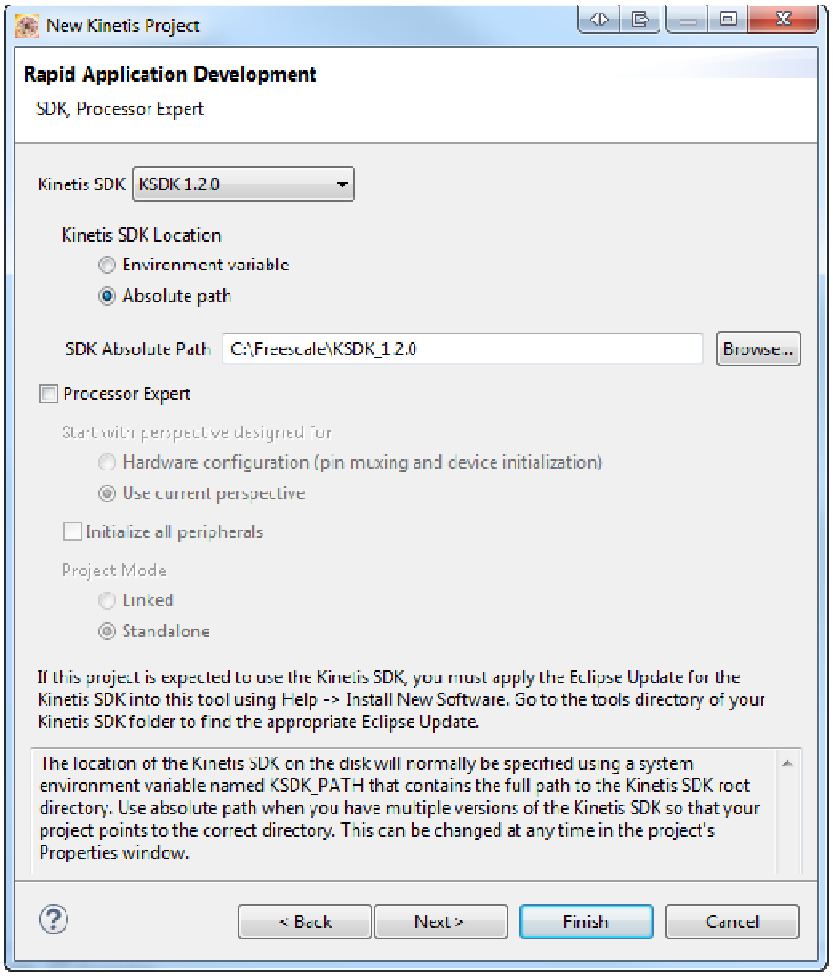


3.3 Select your target and click '**Next'.**



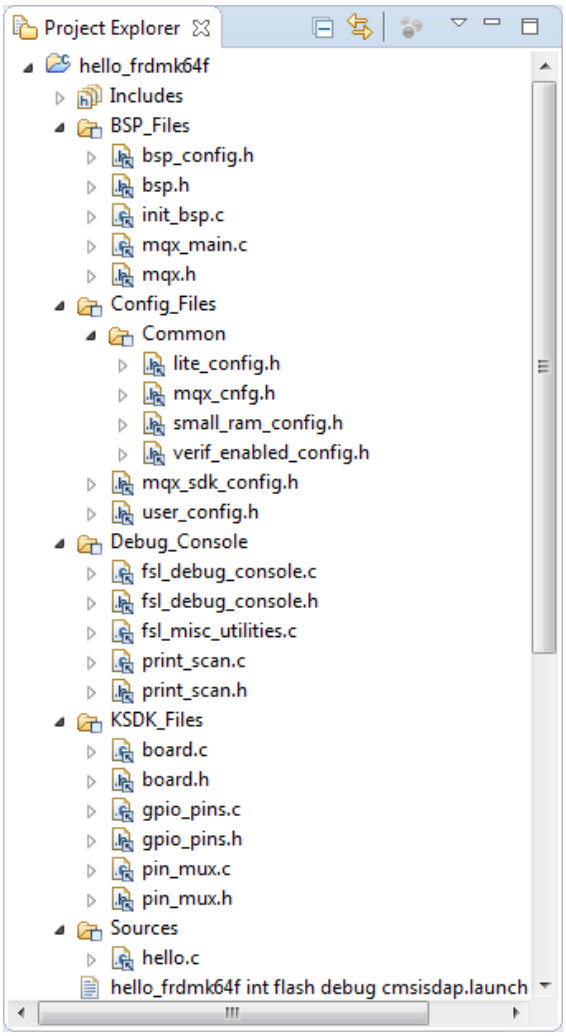
Creating a New *MQX RTOS for KSDK* Project in KDS

3.4 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



3.5 The easiest way to add all the KSDK and MQX support required for a new project is to duplicate the virtual folders structure of an MQX for KSDK example.

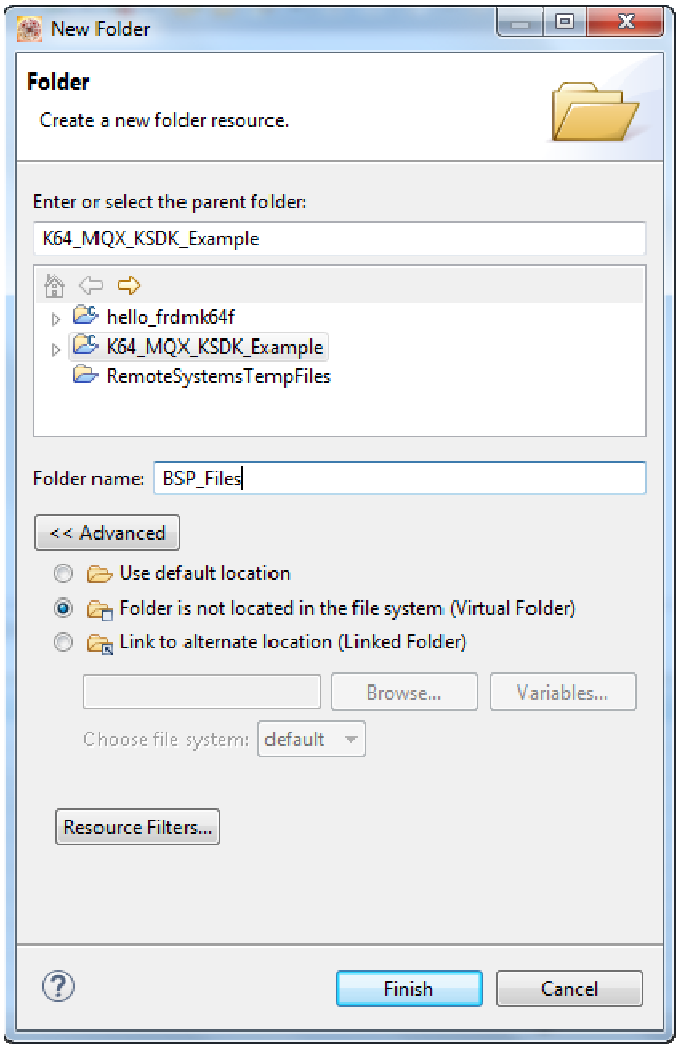
You can simply copy-paste this folder structure from the **'hello_frdmk64f'** example project. In this guide the following steps explain how to create this structure manually.



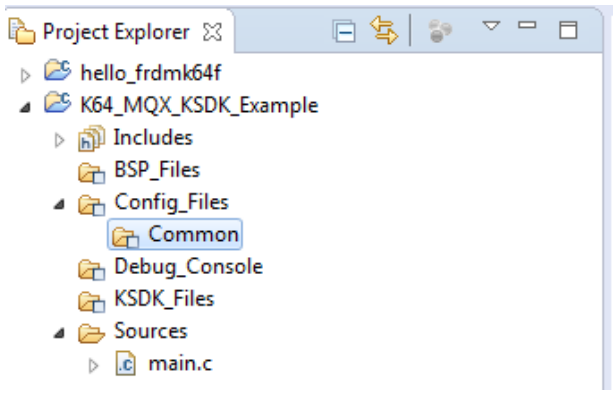
3.6 Erase '**Project_Settings'** and **'SDK'** folders which are created in the default project. Then create 4 folders with the following names.

- BSP_Files
- Config_Files
- Debug_Console
- KSDK_Files

NOTE: You can either create physical folders or virtual folders. To create a virtual folder right click in the project's name and select **New > Folder**. In the New Folder window click **'Advanced'** button and select **'Virtual Folder'.**



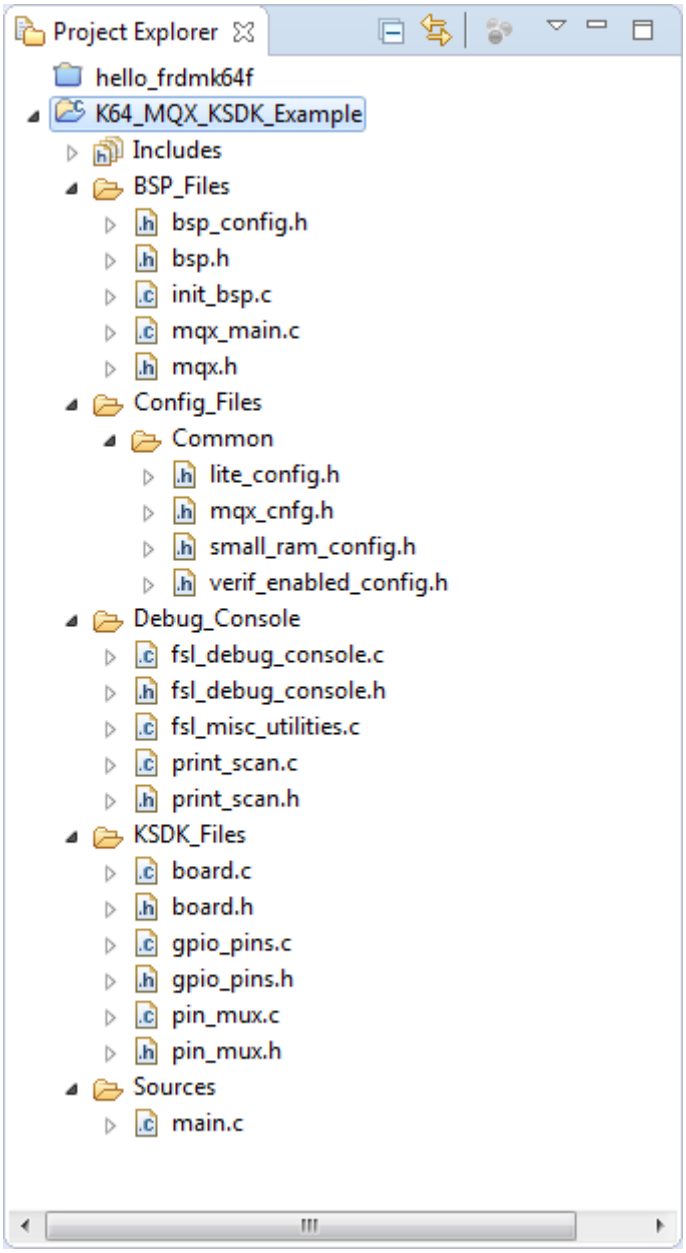
3.7 Then create a sub-folder inside **'Config_Files'** named **'Common'.**



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

- **BSP_Files**
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp\bsp_config.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp\bsp.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp\init_bsp.c
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp\mqx_main.c
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\include\mqx.h
- **Config_Files**
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\mcu\MK64F12\mqx_sdk_config.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\board\frdmk64f\user_config.h
- **Debug_Console**
  - C:\Freescale\KSDK_1.2.0\platform\utilities\src\fsl_debug_console.c
  - C:\Freescale\KSDK_1.2.0\platform\utilities\inc\fsl_debug_console.h
  - C:\Freescale\KSDK_1.2.0\platform\utilities\src\fsl_misc_utilities.c
  - C:\Freescale\KSDK_1.2.0\platform\utilities\src\print_scan.c
  - C:\Freescale\KSDK_1.2.0\platform\utilities\src\print_scan.h
- **KSDK_Files**
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\board.c
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\board.h
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\gpio_pins.c
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\gpio_pins.h
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\pin_muc.c
  - C:\Freescale\KSDK_1.2.0\examples\frdmk64f\pin_mux.h
- **Common**
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common\lite_config.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common\mqx_cnfg.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common\small_ram_config.h
  - C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common\verif_enabled_config.h

NOTE: If you are using physical folders you will be asked whether you want to copy or link the file, in this example physical folders were created and the files are copied into the project.

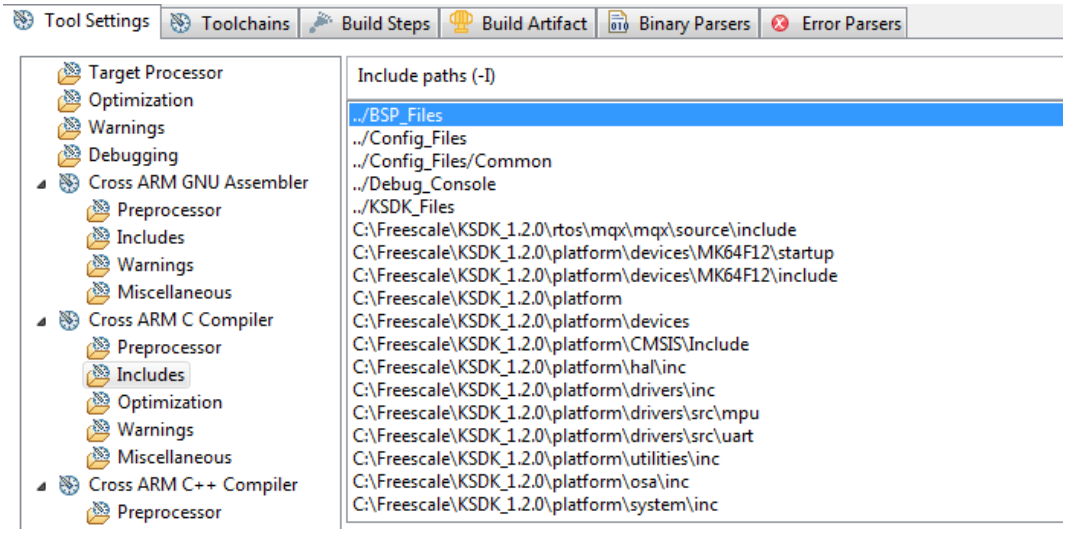

Project Explorer
hello_frdmk64f
K64_MQX_KSDK_Example
Includes
BSP_Files
bsp_config.h
bsp.h
init_bsp.c
mqx_main.c
mqx.h
Config_Files
Common
lite_config.h
mqx_cnfg.h
small_ram_config.h
verif_enabled_config.h
Debug_Console
fsl_debug_console.c
fsl_debug_console.h
fsl_misc_utilities.c
print_scan.c
print_scan.h
KSDK_Files
board.c
board.h
gpio_pins.c
gpio_pins.h
pin_mux.c
pin_mux.h
Sources
main.c

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

If you created physical folders 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/

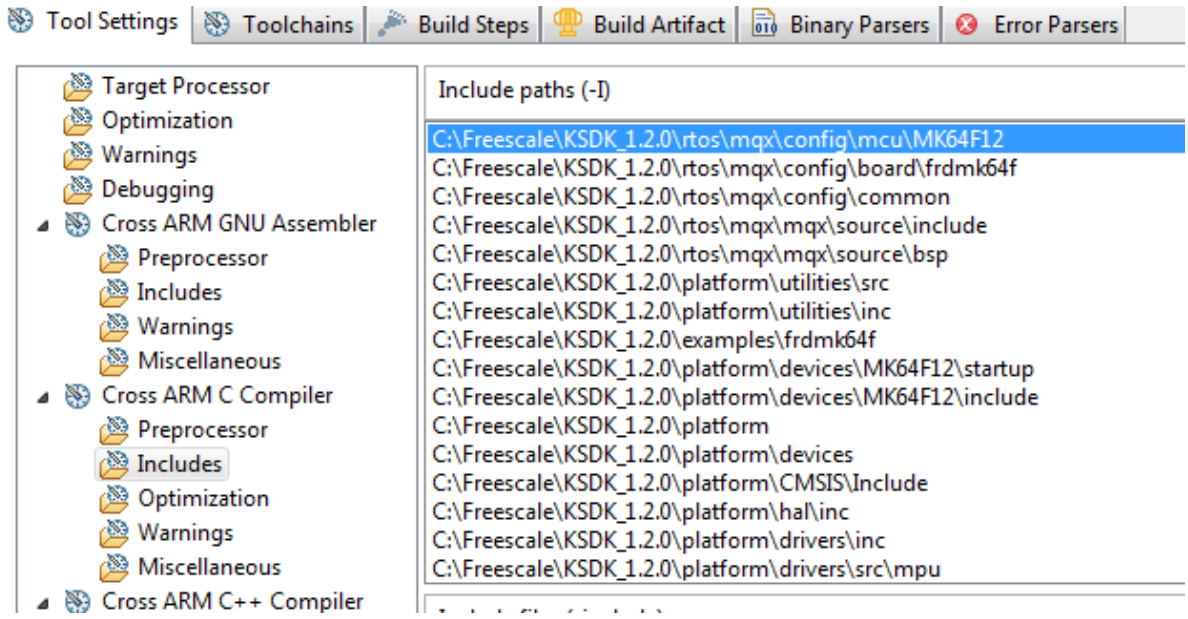
```
../BSP_Files
../Config_Files
../Config_Files/Common
../Debug_Console
../KSDK_Files
C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\include
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\startup
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\include
C:\Freescale\KSDK_1.2.0\platform
C:\Freescale\KSDK_1.2.0\platform\devices
C:\Freescale\KSDK_1.2.0\platform\CMSIS\Include
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\drivers\src\mpu
C:\Freescale\KSDK_1.2.0\platform\drivers\src\uart
C:\Freescale\KSDK_1.2.0\platform\utilities\inc
C:\Freescale\KSDK_1.2.0\platform\osa\inc
C:\Freescale\KSDK_1.2.0\platform\system\inc
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\config
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib
```



If you created virtual folders erase all the content and replace it with the following paths. You can copy and paste the paths below.

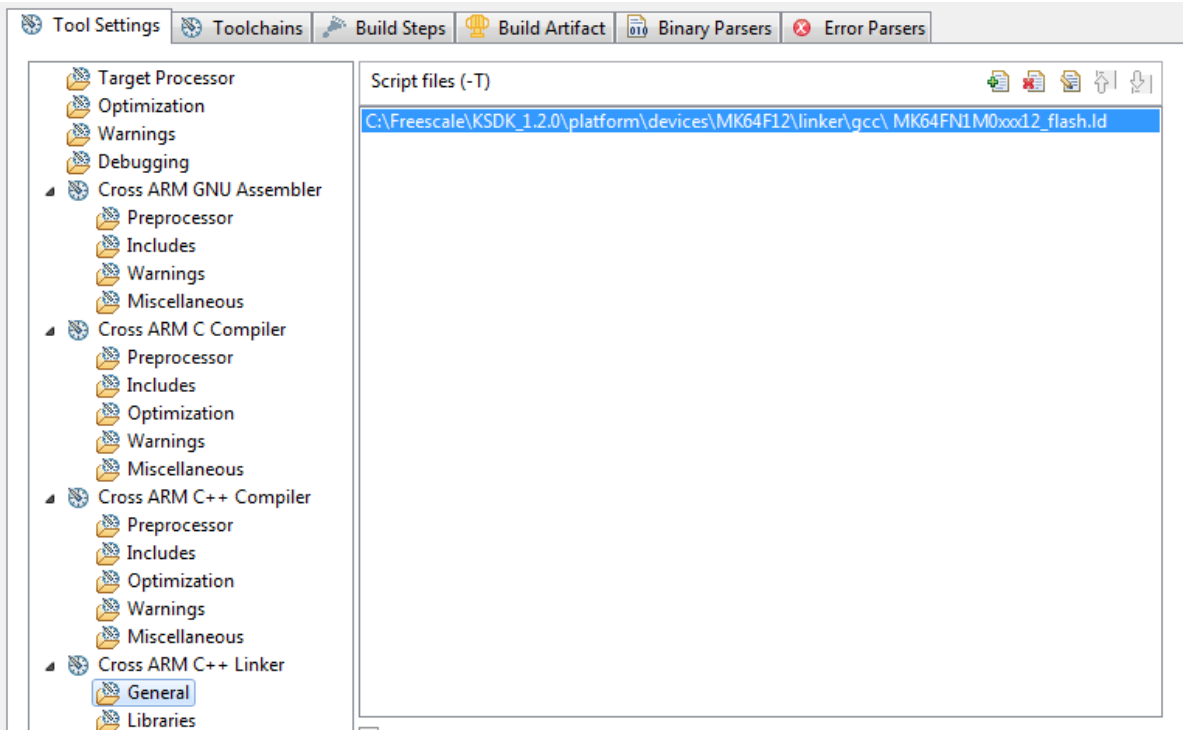
It is possible to add all the paths at once, please refer to the post in the following link:
http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/

```
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\mcu\MK64F12
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\board\frdmk64f
C:\Freescale\KSDK_1.2.0\rtos\mqx\config\common
C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\include
C:\Freescale\KSDK_1.2.0\rtos\mqx\mqx\source\bsp
C:\Freescale\KSDK_1.2.0\platform\utilities\src
C:\Freescale\KSDK_1.2.0\platform\utilities\inc
C:\Freescale\KSDK_1.2.0\examples\frdmk64f
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\startup
C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\include
C:\Freescale\KSDK_1.2.0\platform
C:\Freescale\KSDK_1.2.0\platform\devices
C:\Freescale\KSDK_1.2.0\platform\CMSIS\Include
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\drivers\src\mpu
C:\Freescale\KSDK_1.2.0\platform\drivers\src\uart
C:\Freescale\KSDK_1.2.0\platform\utilities\inc
C:\Freescale\KSDK_1.2.0\platform\osa\inc
C:\Freescale\KSDK_1.2.0\platform\system\inc
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\config
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib
```



3.10 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > General** and fix the linker file path by setting the absolute path.

C:\Freescale\KSDK_1.2.0\platform\devices\MK64F12\linker\gcc\MK64FN1M0xxx12_flash.ld

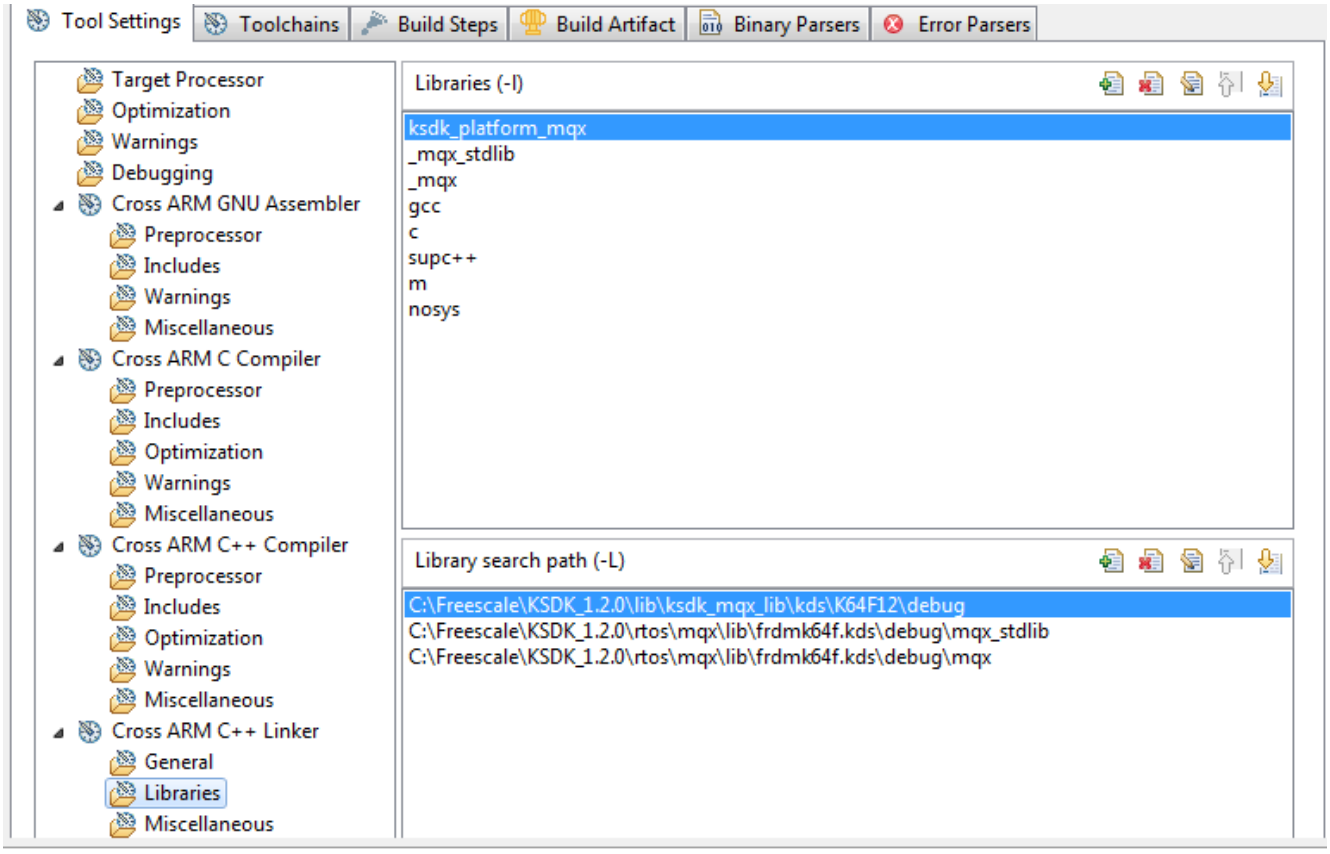


NOTE: If you plan to edit the linker file it is strongly recommended to make a copy of the .ld file because all the new projects and example projects will be affected by the .ld changes.

3.11 Go to menu **Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > Libraries** and add all the Libraries search paths.

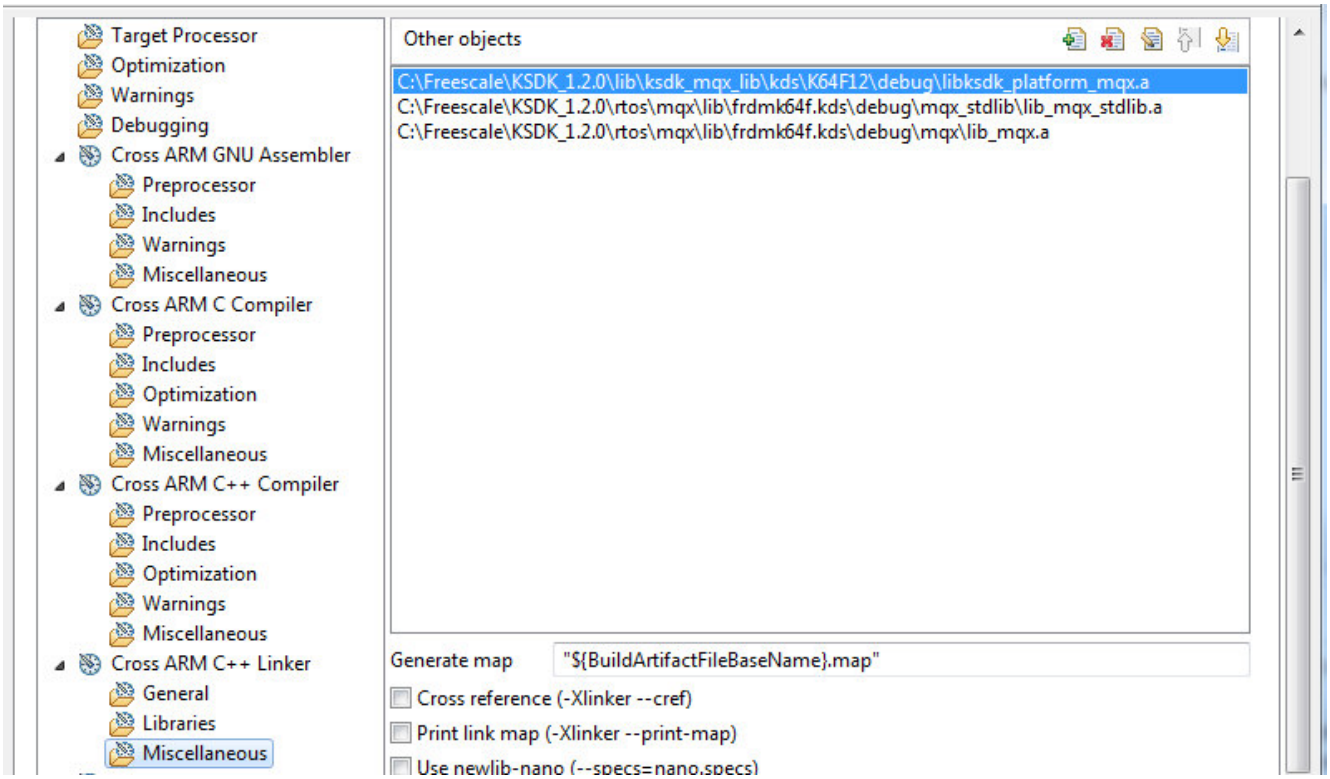
It is possible to add all the paths at the same time, please refer to the post in the following link: http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/

```
ksdk_platform_mqx
_mqx_stdlib
_mqx
gcc
c
supc++
m
nosys


C:\Freescale\KSDK_1.2.0\lib\ksdk_mqx_lib\kds\K64F12\debug
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx
```

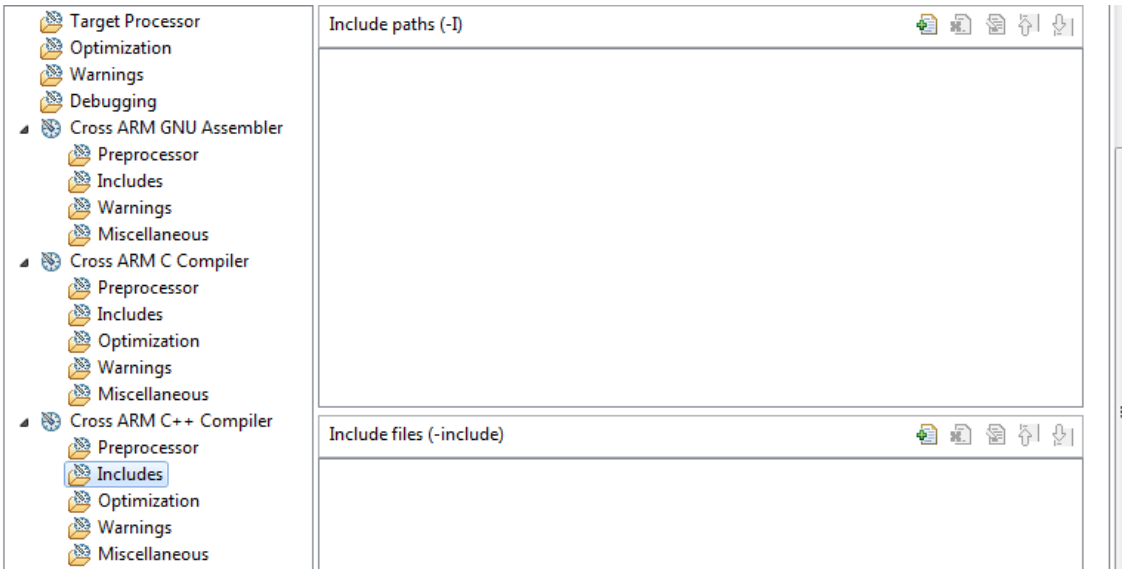


3.12 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > Miscellaneous** and add all the Libraries with their full paths.
It is possible to add all the paths at the same time, please refer to the post in the following link:
http://mcuoneclipse.com/2014/12/23/adding-multiple-include-paths-to-build-settings-in-eclipse/

```
C:\Freescale\KSDK_1.2.0\lib\ksdk_mqx_lib\kds\K64F12\debug\libksdk_platform_mqx.a
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx_stdlib\lib_mqx_stdlib.a
C:\Freescale\KSDK_1.2.0\rtos\mqx\lib\frdmk64f.kds\debug\mqx\lib_mqx.a
```



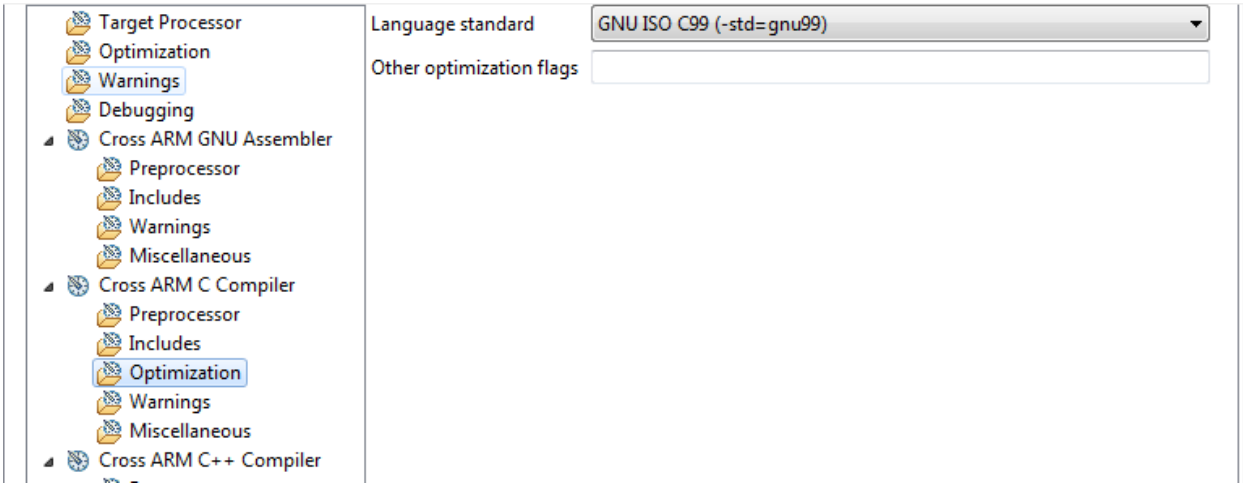
3.13 At this point you may see some warnings. This is because in this project there are Compiler settings for C and C++ and we have not fixed the C++ Compiler settings. In this case we are not using C++, go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Compiler > Includes** and erase all the paths.



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

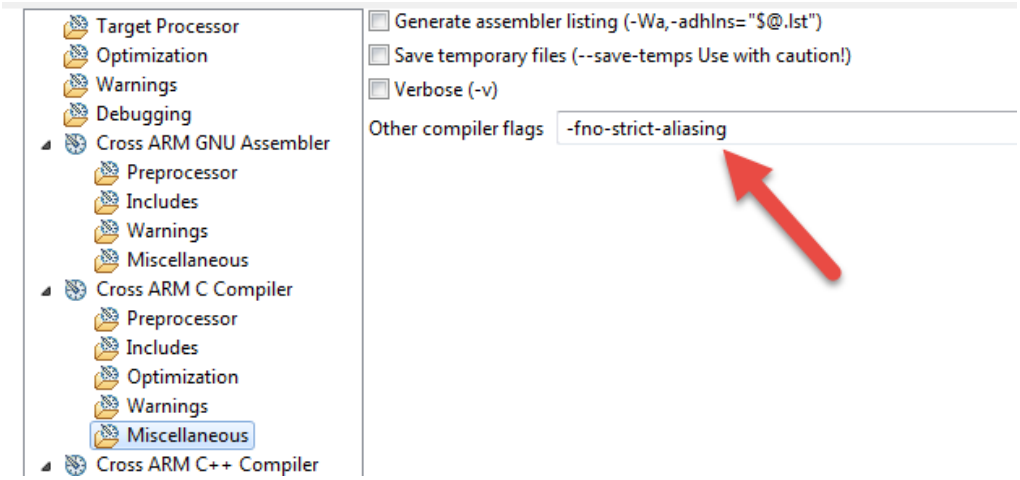
```
CPU_MK64FN1M0VMD12=1
FSL_RTOS_MQX=1
_AEABI_LC_CTYPE=C
__STRICT_ANSI__=1
_DEBUG=1
```

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



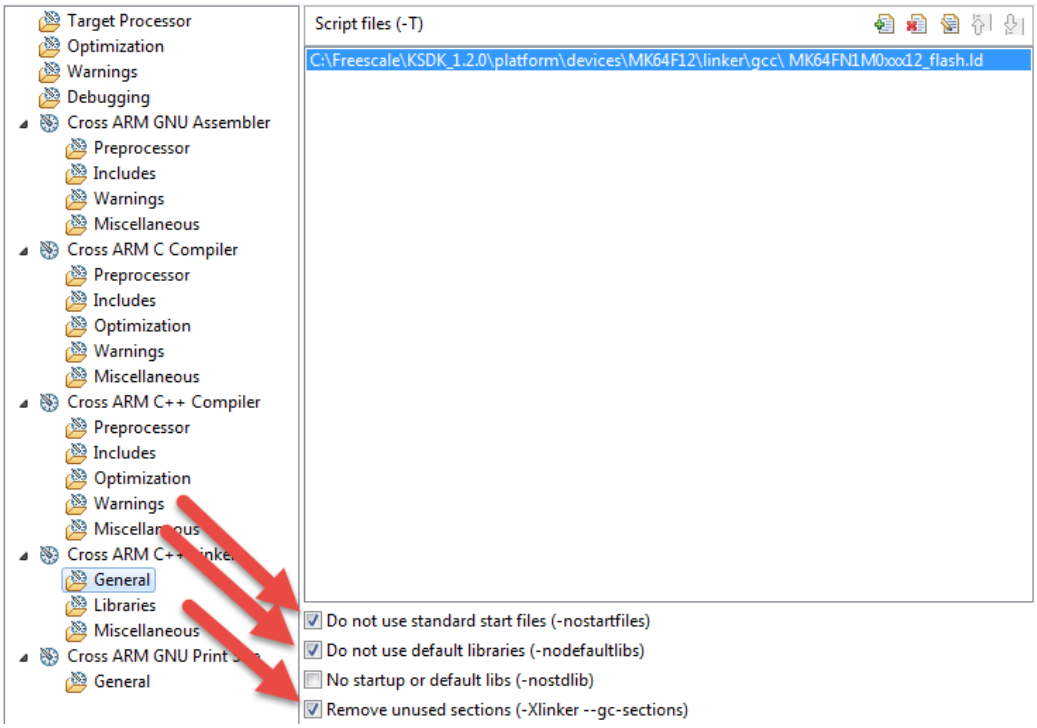
3.16 Go to menu **Project > Properties > C/C++ Build > Settings > Cross ARM C Compiler > Miscellaneous** and add the following flag.

```
-fno-strict-aliasing
```



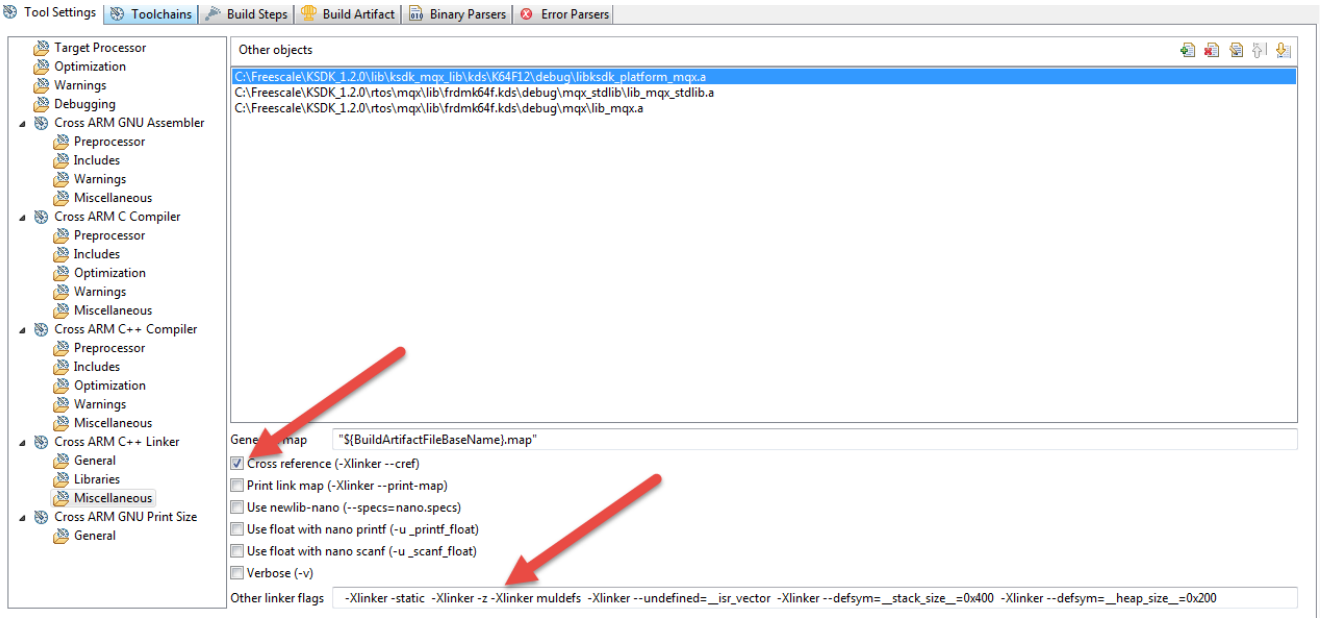
3.17 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > General** and check the following options.

Do not use standard start files Do not use default libraries Remove unused sections



3.18 Go to **menu Project > Properties > C/C++ Build > Settings > Cross ARM C++ Linker > Miscellaneous** check '**Cross Reference'** option and set the following flags, then press **'Apply'** and **'Ok'** buttons.

```
-Xlinker -static -Xlinker -z -Xlinker muldefs -Xlinker --undefined=__isr_vector -Xlinker --defsym=__stack_size__=0x400 -Xlinker --defsym=__heap_size__=0x200
```

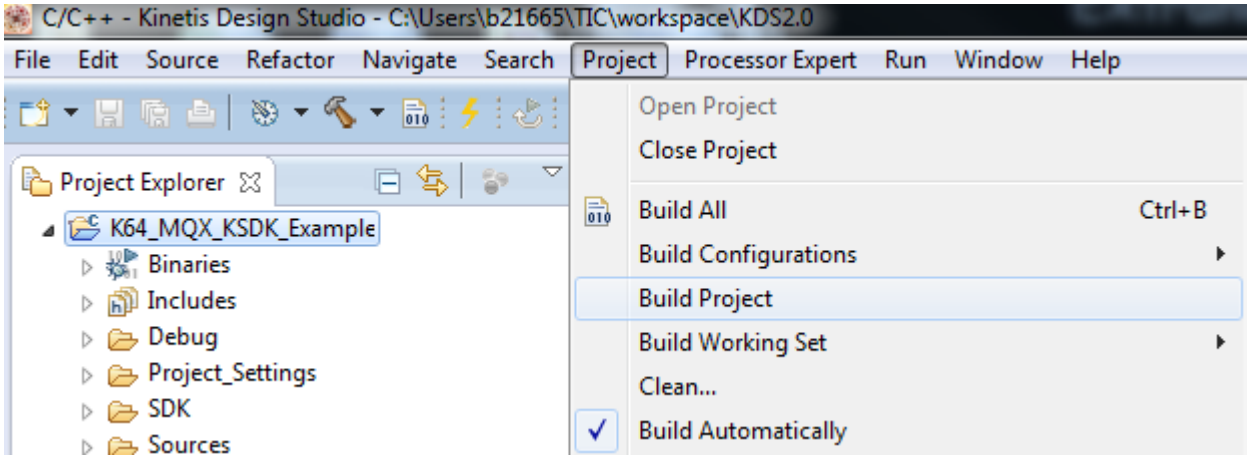


3.19 Erase the code in ***'main.c'*** and replace it with the code below. As you can notice there is no ***'main()'*** this is because it is in the MQX library code.

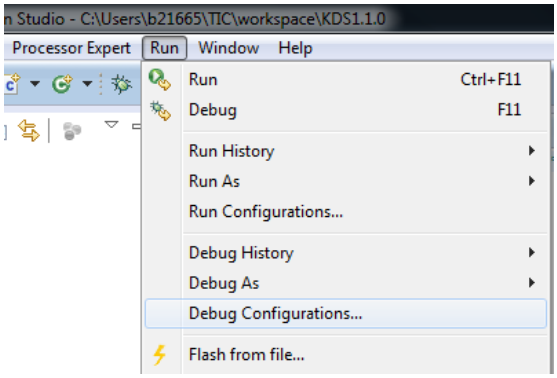
```
#include <stdio.h>
#include <mqx.h>
#include <bsp.h>

/* Task IDs */
#define HELLO_TASK  5
#define WORLD_TASK  6

extern void hello_task(uint32_t);
extern void world_task(uint32_t);

const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index,   Function,   Stack,  Priority, Name,     Attributes,          Param, Time Slice */
    {WORLD_TASK, world_task, 700, 9, "world", MQX_AUTO_START_TASK, 0, 0},
    {HELLO_TASK, hello_task, 700, 8, "hello", 0,                   0, 0},
    {0}
};

/*TASK*-----------------------------------------------------
*
* Task Name    : world_task
* Comments     :
*    This task creates hello_task and then prints " World ".
*
*END*-----------------------------------------------------*/
void world_task(uint32_t initial_data)
{
   _task_id hello_task_id;

   hello_task_id = _task_create(0, HELLO_TASK, 0);
   if (hello_task_id == MQX_NULL_TASK_ID)
   {
      printf ("\n Could not create hello_task\n");
   }
   else
   {
      printf(" World \n");
   }

   _task_block();
}

/*TASK*-----------------------------------------------------
*
* Task Name    : hello_task
* Comments     :
*    This task prints " Hello".
*
*END*-----------------------------------------------------*/

void hello_task(uint32_t initial_data)
{
   printf("\n Hello\n");
   _task_block();
}

/* EOF */
```

4 Run the application

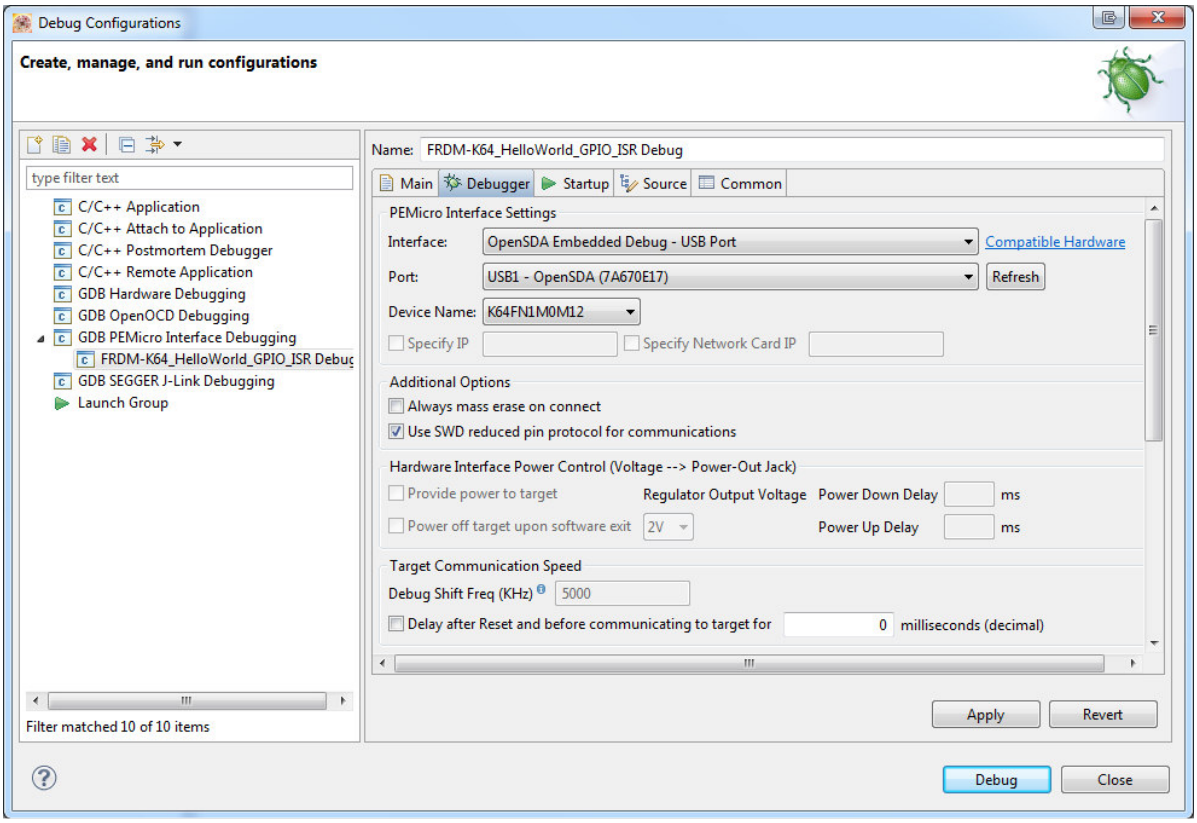
4.1 Build your application, go to ***menu Project > Build Project***. Alternately click the hammer button.



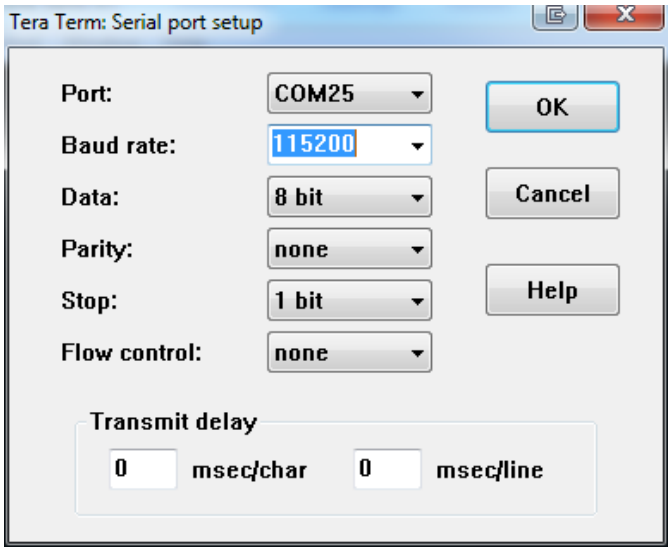
4.2 Go to menu Run > Debug Configurations…



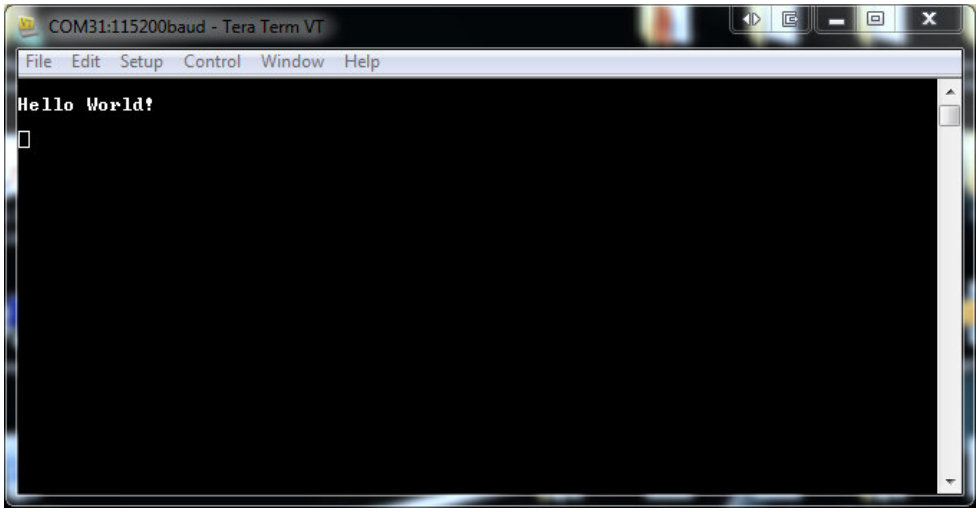
4.3 Select the ***'Debug Configuration'*** that matches your connection type, in this example **P&E Micro** connection is used, if you don't know which your connection type is or you want to change your connection type see '***Appendix C'*** at the end this document. Once you double click the appropriate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



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



4.5 Run the application, you will see ***"Hello World"*** in terminal.



APPENDIX C: Connection Types

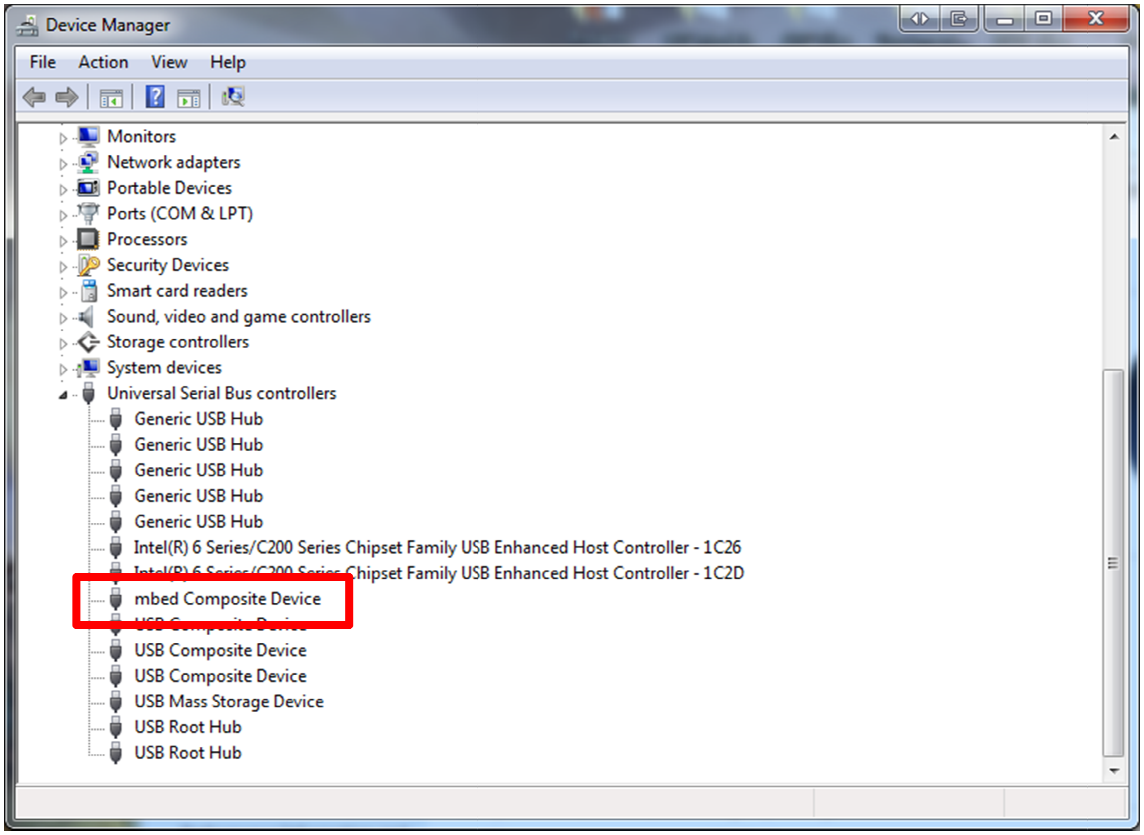
- KDS works with devices which support OpenSDAv2 connection.
- You can find Open SDA User's Guide here: http://www.freescale.com/files/32bit/doc/user_guide/OPENSDAUG.pdf
- You can learn more about Open SDA in the following link: https://community.freescale.com/docs/DOC-100720

Identify your Connection Type

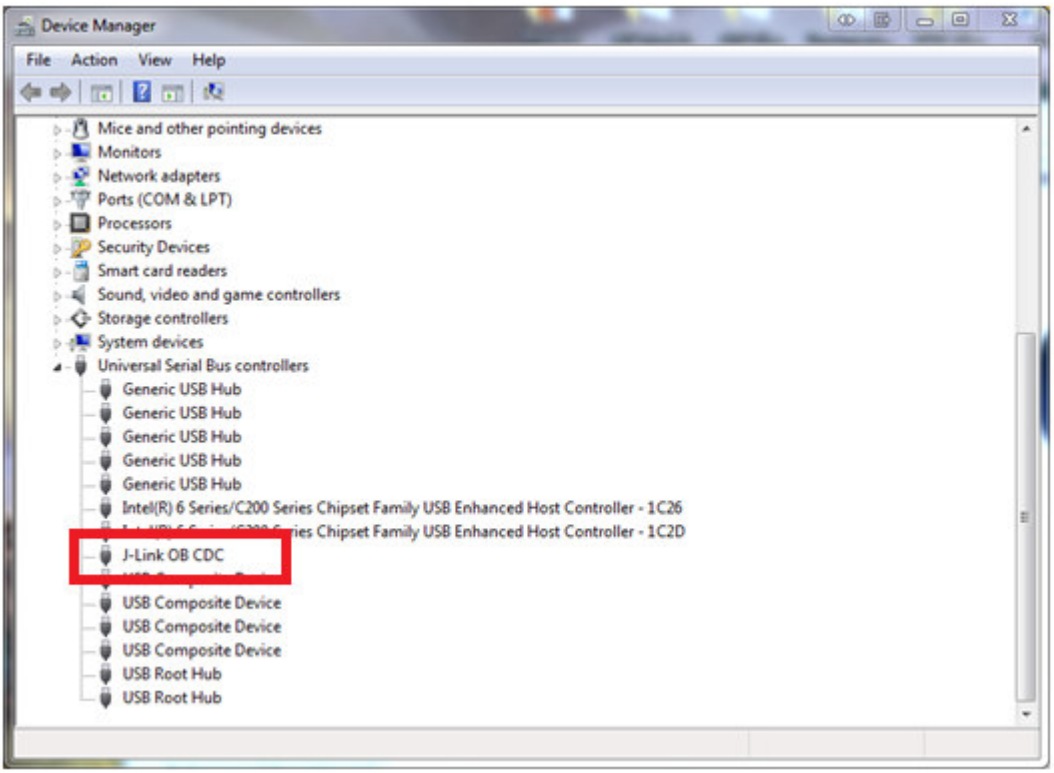
To find out which your connection type is you must connect your device to your computer and go to Windows Device Manager, here you can see the connection used by your device. You can see how to open Windows device manager in the link below:
http://windows.microsoft.com/en-us/windows/open-device-manager#1TC=windows-7

MBED Connection

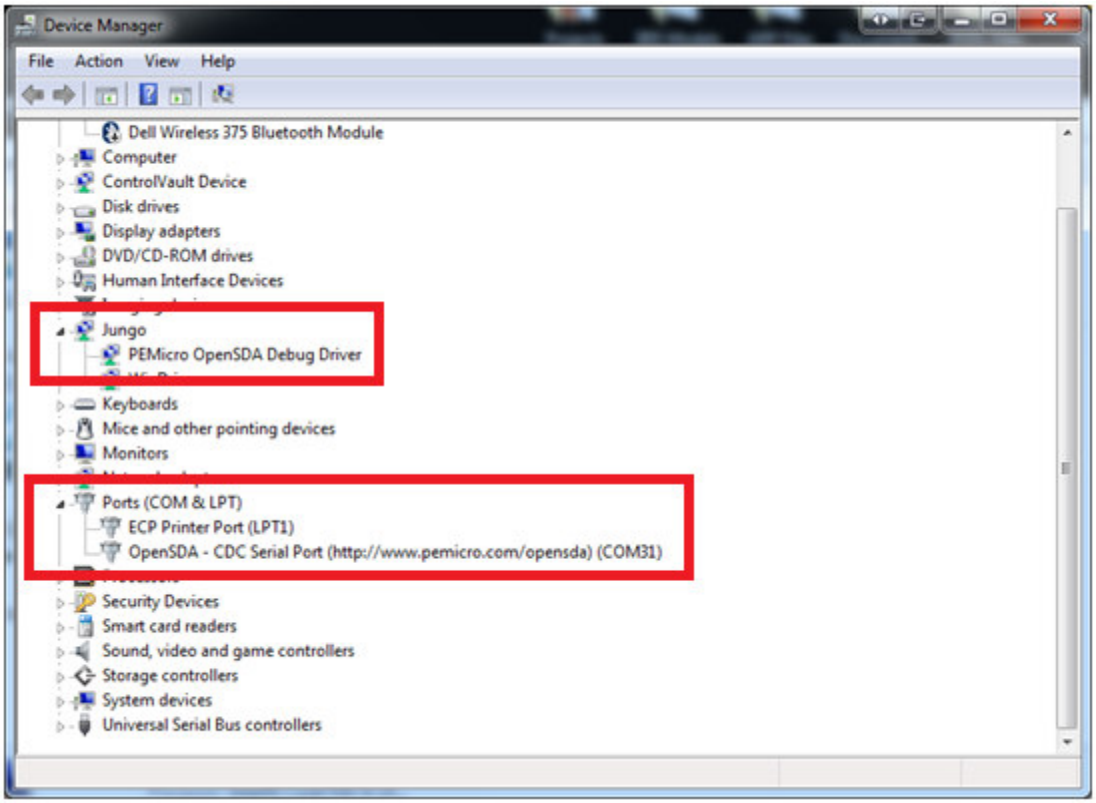
Please note that this connection is not supported in KDS yet.



Segger J-link connection



P&E Micro Connection

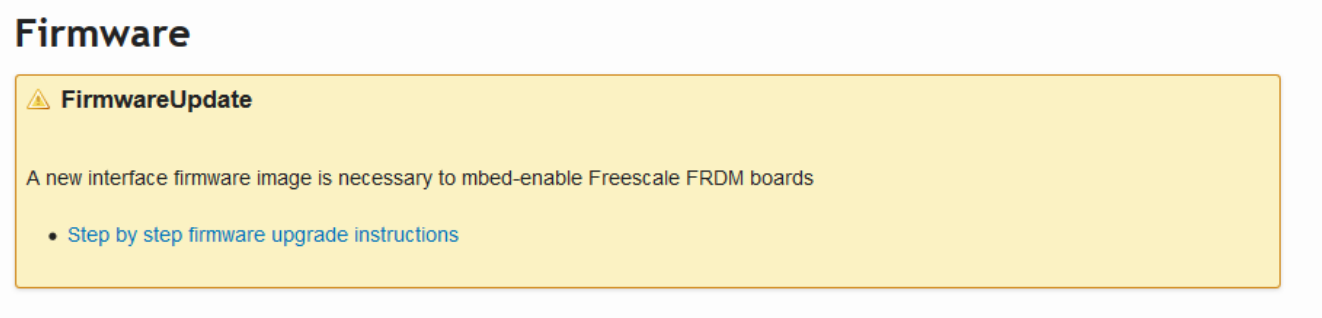


Switching or updating your connection firmware

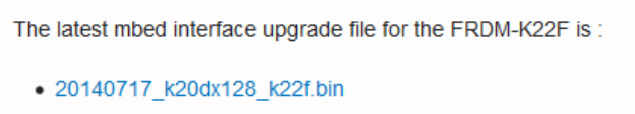
You can download different versions of OpenSDA form our partners' web sites.

MBED

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

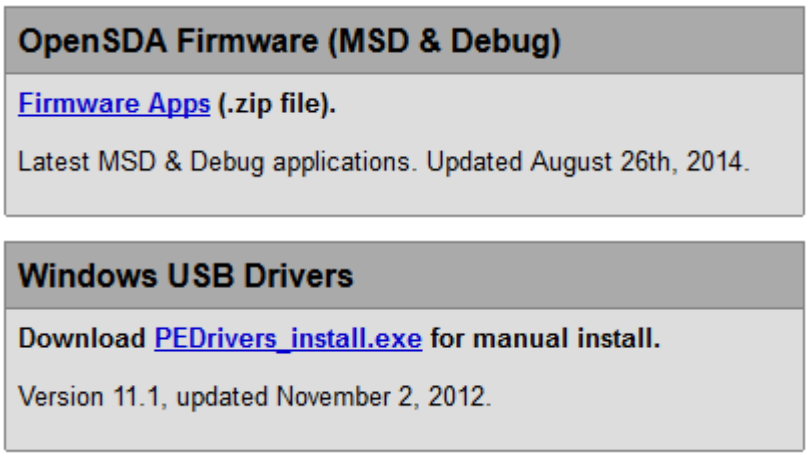


4) Save the latest firmware and follow the instructions to do the update



P&E Micro

1) Go to http://www.pemicro.com/opensda/
2) Download '***Open SDA Firmware'*** and optionally '***Windows USB Drivers'***



3) 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

Getting started with OpenSDA V2

Later Freescale boards like the FRDM-K64F come with a new version of OpenSDA, called OpenSDA V2. This version comes with a different bootloader and needs a different firmware (*.sda file format is no longer supported).
The firmware can be downloaded here: Firmware download
The steps for the firmware update etc. are equal to the old OpenSDA version and are explained above.

Getting started with OpenSDA V2.1

Later Freescale boards come with a new version of OpenSDA, called OpenSDA V2.1. This version comes with a different bootloader and needs a different firmware (*.sda file format is no longer supported).
The firmware can be downloaded here: Firmware download
The steps for the firmware update etc. are equal to the old OpenSDA version and are explained above.

3) 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/