Freescale Semiconductor

Writing my first KSDK Application in KDS

Hello World and Toggle LED with GPIO Interrupt

By: Technical Information Center

This document explains briefly how to create a Hello World project and toggle a LED using GPIO Interrupts. KSDK1.1 and KDS2.0 are used in this example.

freescale semiconductor

1 Before you begin

1.1 Install KDS (Kinetis Design Studio), you can find an installation guide in the following link: https://community.freescale.com/docs/DOC-102288

1.2 Install KSDK (Kinetis Software Development Kit), you can find an installation guide in the following link: https://community.freescale.com/docs/DOC-102271

1.3 Install ***'Eclipse Update for KSDK 1.1.0-GA'*** and ***'Processor Expert for KDS'*** updates. See '***Appendix A'*** in this document. You may also install ***'MQX Plug-ins'*** if you will use MQX later

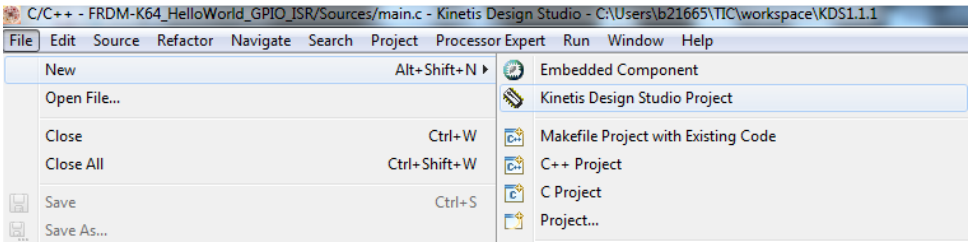
1.4 Verify that ***'KSDK_PATH'*** environment variable is set correctly. See '***Appendix B'*** in this document

1.5 It is necessary to build KSDK Platform Driver Library (libksdk_platform.a)

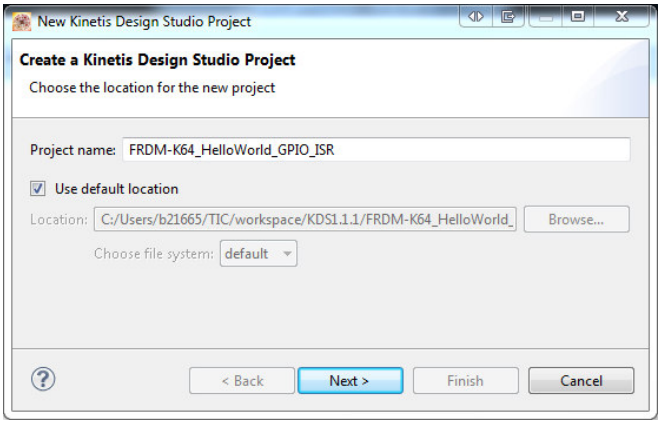
Open ***'Getting Started with Kinetis Software Development Kit (KSDK)'*** located in ***'C:\Freescale\KSDK_1.1.0\doc'*** and go to '**7 Build and run the KSDK demo applications using Kinetis Design Studio IDE'** go through sections '7.2 Open the platform driver library' and '7.3 Build the platform driver library' and follow the steps.

2 Create KDS project using KSDK

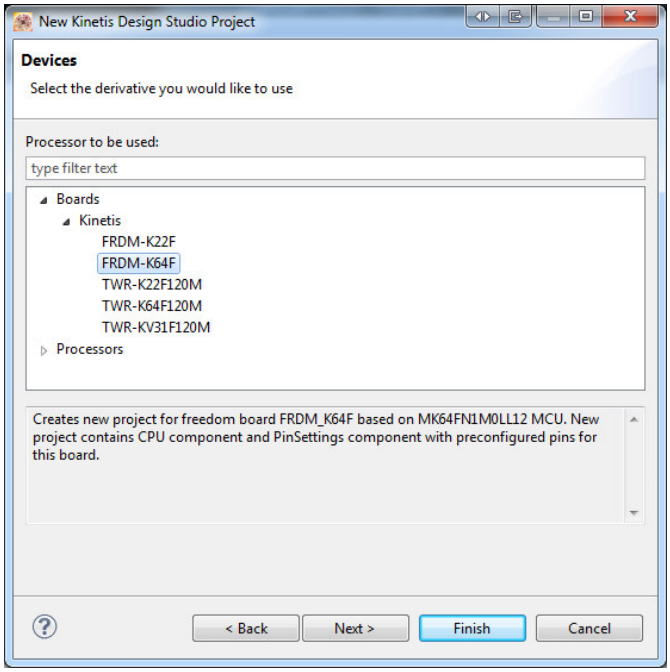
2.1 Go to ***menu File > New > Kinetis Design Studio Project***.



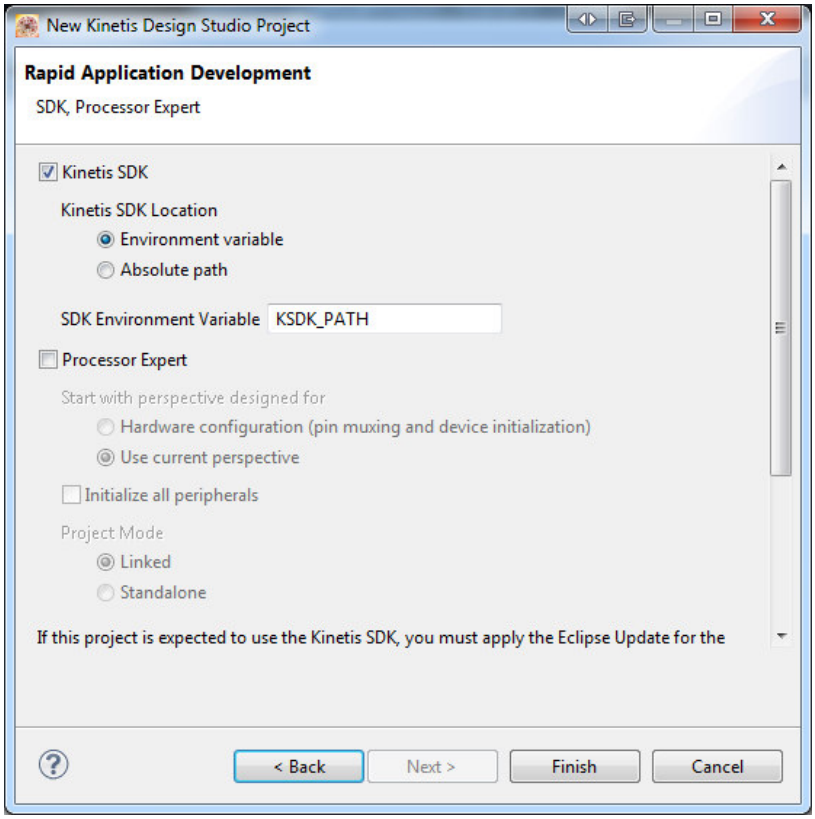
2.2 Write a name for your project and click ***'Next'***.



2.3 Select your target

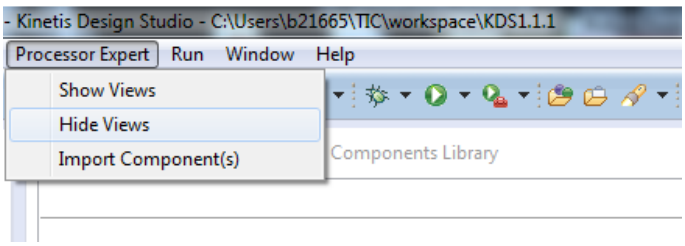


2.4 Check ***'Kinetis SDK'*** to include KSDK support and click ***'Finish'***.

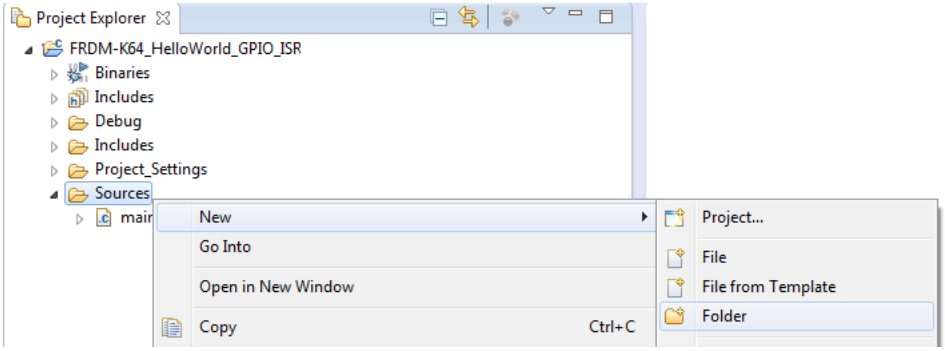


3 Develop the application

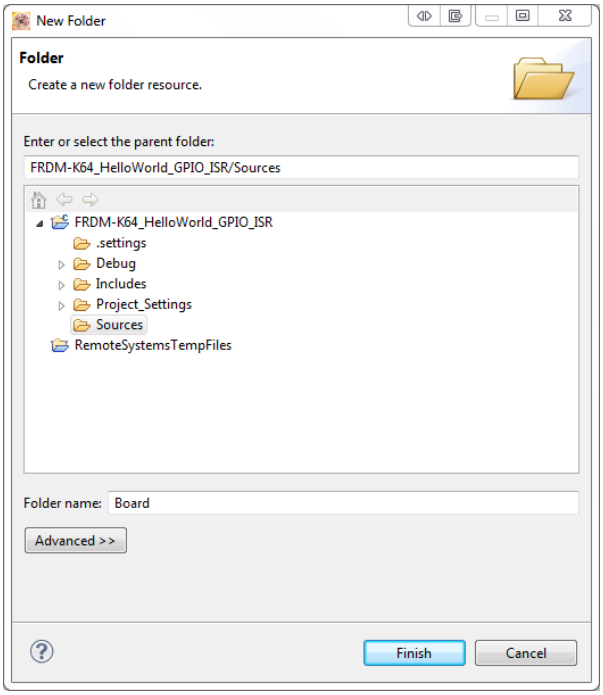
3.1 If Processor Expert views are shown you can hide them as Processor Expert is not required in this application. Go to ***menu Processor Expert > Hide Views***.



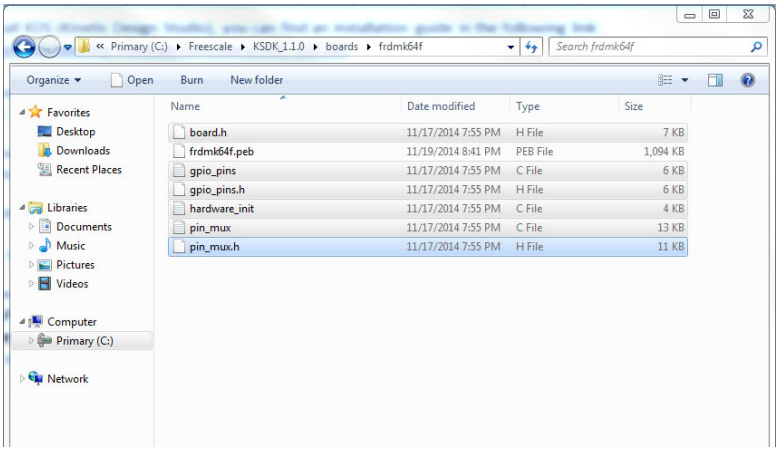
3.2 Right click over ***'Sources'*** folder and select ***New > Folder*** to create a new folder inside ***'Sources'*** folder.



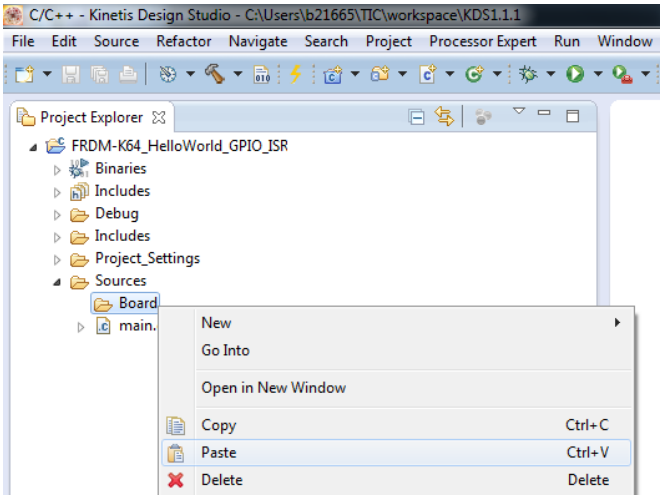
3.3 Name this folder ***'Board'*** and click ***'Finish'***.



3.4 Go to ***C:\Freescale\KSDK_1.1.0\boards\<board>*** and copy all the source files (copy all .c and .h files, do not copy .peb file).



3.5 Now paste them in folder ***'Board'*** which has been created.



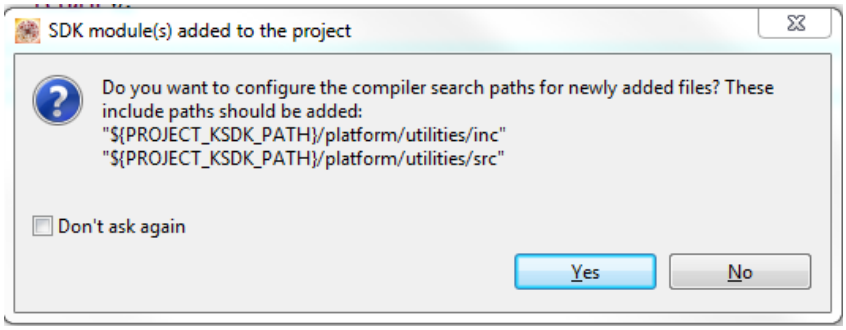
- **board.h**: Contains platform-specific configuration macros for terminal configuration, push buttons, LEDs and other board-specific items.
- **gpio_pins.c/h**: Definitions used by the KSDK GPIO driver for the platform's GPIO pins. These include push buttons and LEDs, but can include other items such as interrupt pins for external sensors, for example.
- **hardware_init.c**: Contains the hardware_init() function called by the main() loop of the demo application. Calls pin configuration functions defined in the pin_mux.c file.
- **pin_mux.c/h**: Contains peripheral-specific pin mux configurations. These functions can be called by the hardware_init() function or individually by the demo application.
- **Processor Expert PEB file**: Reference file for Freescale's Processor Expert tool for the specific hardware platform.

NOTE: For more information please refer to 'Getting Started with Kinetis SDK' document which is included in KSDK1.1.0 installation path *C:\Freescale\KSDK_1.1.0\doc\ Getting Started with Kinetis SDK (KSDK).pdf*.

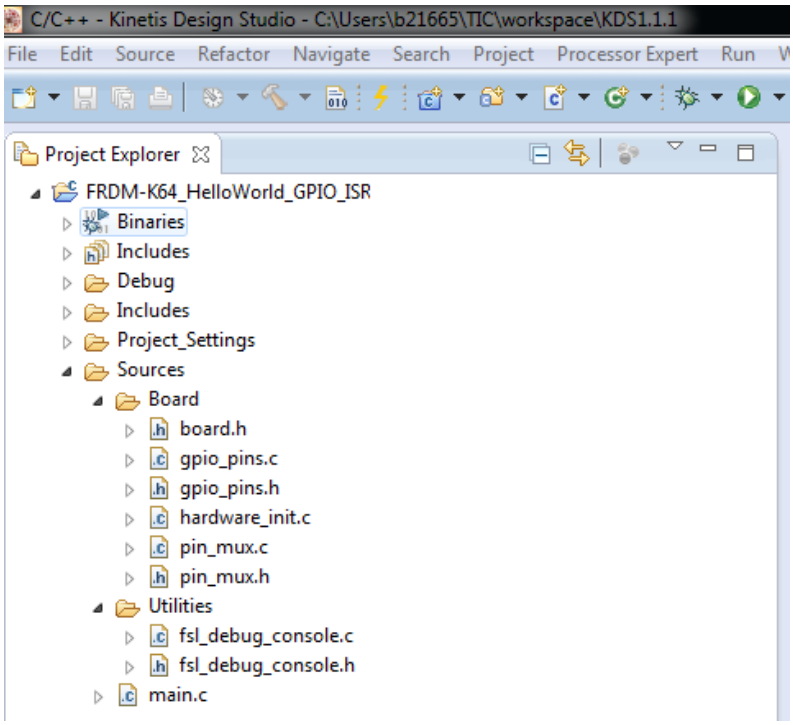
3.6 Repeat the same process in steps 3.2 and 3.3 for another folder and name it ***'Utilities'***, then go to ***C:\Freescale\KSDK_1.1.0\platform\utilities\inc*** and copy ***'fsl_debug_console.h'*** to folder ***'Utilities'***, then go to ***C:\Freescale\KSDK_1.1.0\platform\utilities\src*** and copy ***'fsl_debug_console.c'*** too.

- **fsl_debug_console.c/h**: Provides initialization and basic functionality for the UART module that is connected to the debug interface.

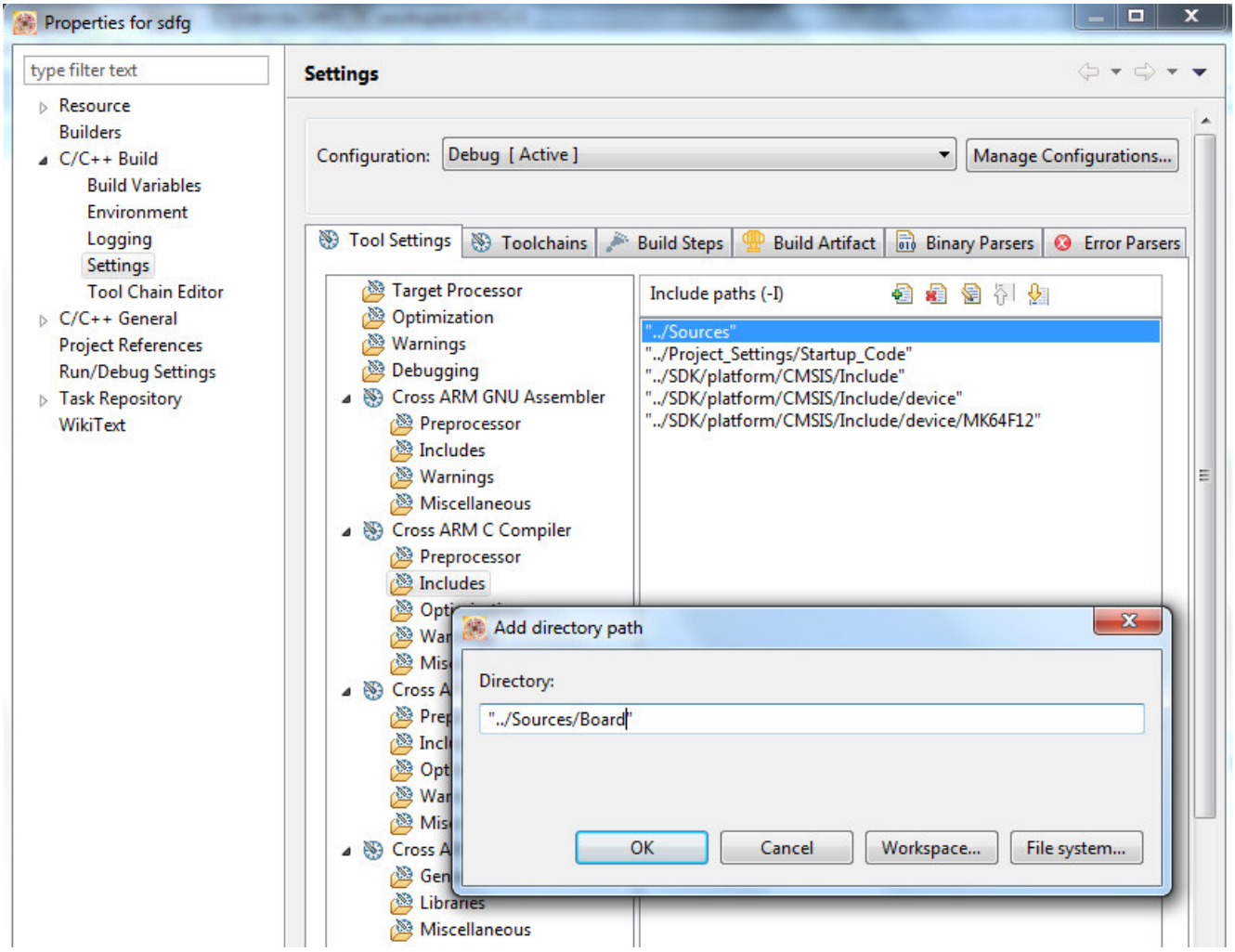
NOTE: After copying 'fsl_debug_console.c' to utilities folder the prompt below may appear, in this case click '***Yes***' and continue.



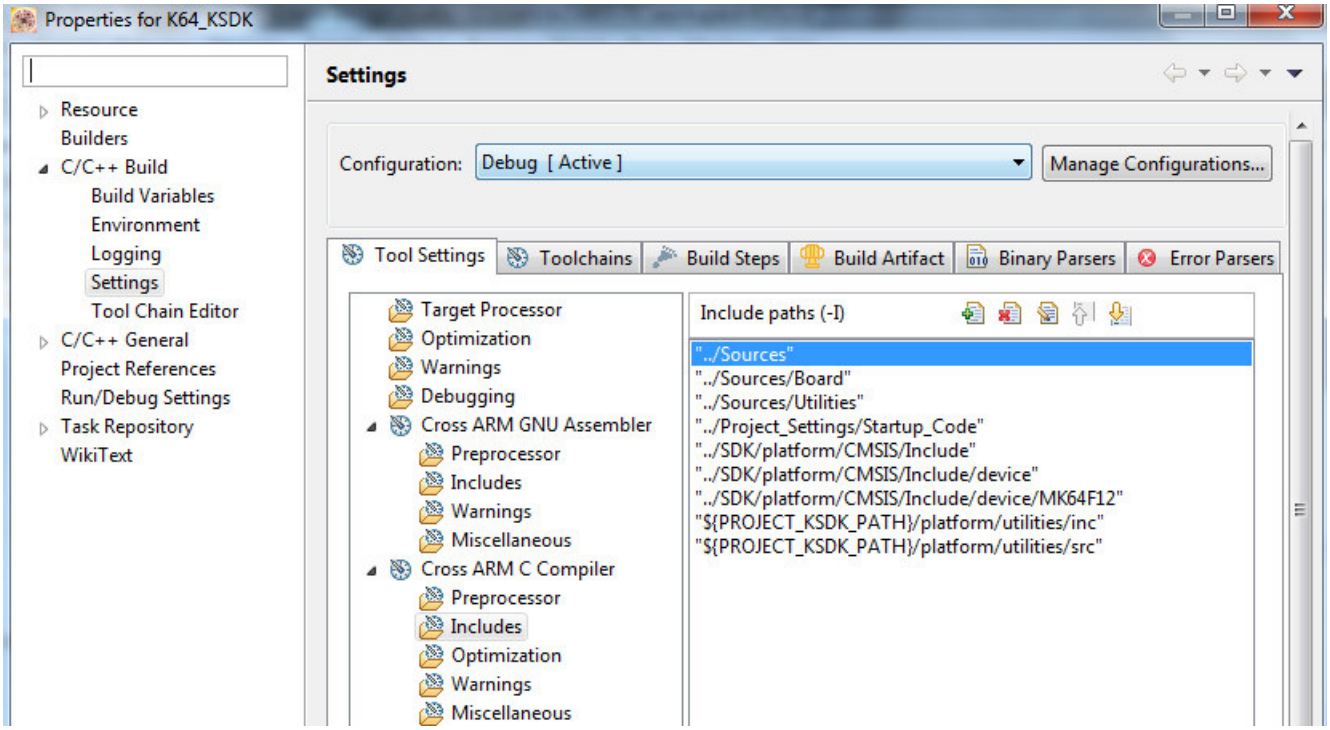
Now your project's source folders tree must look as shown below.



3.7 Go to ***menu Project > Properties C/C++ Build > Settings > Cross ARM C Compiler > Includes*** and add to ***'Include paths (-I)'*** the path ***"../Sources/Board"***. **DO NOT COPY-PASTE** this line because quotes cause problems when copy-pasting.



3.8 Repeat the same process in step 3.7 for the folder ***'Utilities'***, then click ***'Apply'*** and ***'Ok'***. The compiler Includes settings must look as shown below.



3.9 Erase the code in ***'main.c'*** and replace it with the code below.

```
#include "fsl_device_registers.h"
#include "board.h"
#include <stdio.h>

#define GPIO_INTERRUPT    1  //Set value to 1 for interrupt, 0 for polling

int main(void)
{
	hardware_init();
	dbg_uart_init();

	printf("Hello World \n\r");

#if GPIO_INTERRUPT
	// Configure pin (sw1) interrupt if GPIO_INTERRUPT is set
	switchPins[0].config.interrupt = kPortIntFallingEdge;
#endif
	// Initializes GPIO driver
	GPIO_DRV_Init(switchPins, ledPins);

	while(1)
	{
#if !GPIO_INTERRUPT
		// Poll (sw1) if GPIO_INTERRUPT is not set
		if(GPIO_DRV_ReadPinInput(kGpioSW1) == 0)
		{
			GPIO_SW_DELAY;
			GPIO_DRV_TogglePinOutput(BOARD_GPIO_LED_BLUE);
		}
#endif
	}
	return 0;
}

#if GPIO_INTERRUPT
// Define PORTC_IRQHandler (sw1) if GPIO_INTERRUPT is set
void PORTC_IRQHandler()
{
	GPIO_DRV_ClearPinIntFlag (kGpioSW1); // Clear IRQ flag
	GPIO_DRV_TogglePinOutput(BOARD_GPIO_LED_BLUE); // Toggle blue LED
	GPIO_SW_DELAY;
}
#endif

/*EOF*/
```

- **hardware_init():** Is defined in '***hardware_init.c'***, this function sets up the clock source, enables the clock for PORTs and configure the GPIO pins connected to LEDs and push buttons.
- **dbg_uart_init():** Is also defined in **'*hardware_init.c*'** and it sets up the UART port which is connected to the debug interface as well as the pins required by the UART.
- **switchPins[0].config.interrupt = kPortIntFallingEdge:** In ***'gpio_pins.c'*** several structures for GPIO configuration are defined and initialized, these structures provide fields to configure a friendly name for the GPIO, pullup, pulldown, interrupt, slewrate, open drain among other functionalities. The types of these structures are '***gpio_input_pin_user_config_t***' and '***gpio_output_pin_user_config_t'*** and it is necessary to define a structure for each GPIO pin that is going to be configured. In the case of the LEDs and push buttons there are 2 arrays of structures are defined; switchPins and ledPins. This instruction reconfigures the interrupt field of *kGpioSW1* to enable falling edge interrupt if GPIO_INTERRUPT macro is set. You can find more details about the GPIO structures and types in **'Kinetis SDK v.1.1 API Reference Manual'** which is located in KSDK installation path *C:\Freescale\KSDK_1.1.0\doc\Kinetis SDK v.1.1 API Reference Manual*.
- **GPIO_DRV_Init():** This function is part of the GPIO driver, it receives the 2 GPIO arrays of structures to initialize the GPIO driver with these parameters. For more information please see **'Kinetis SDK v.1.1 API Reference Manual'** which is located in KSDK installation path *C:\Freescale\KSDK_1.1.0\doc\Kinetis SDK v.1.1 API Reference Manual.*
- **PORTC_IRQHandler**(**):** In '***gpio_pins.h'*** an enum is declared to associate the GPIO friendly name with a pin, here *kGpioSW1* is associated to PTC6. As you can see in ***'FRDM-K64F Freedom Module User's Guide'*** SW2 in FRDM-K64F board is connected to PTC6.

NOTE: If you are using a different board the pin number may be different.

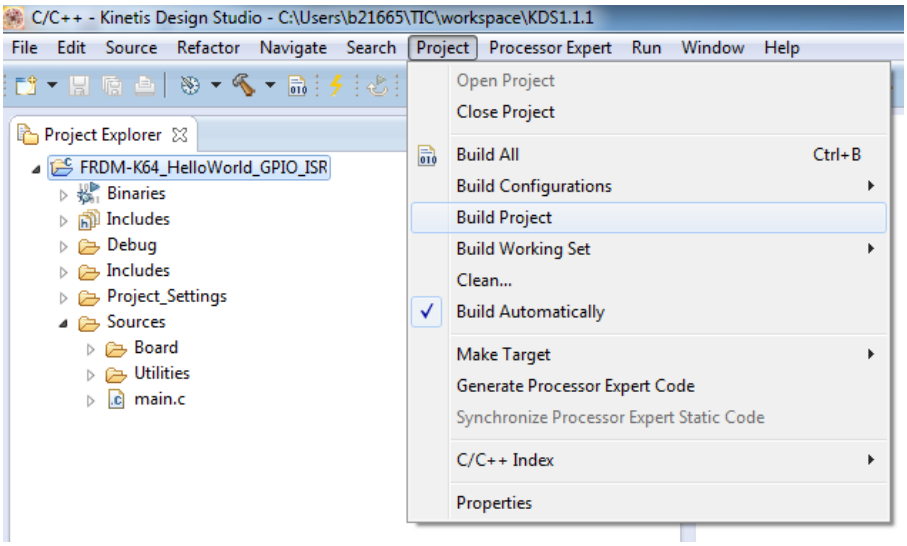
You can find the user's guides of the freedom and tower boards as well as the schematics in www.freescale.com/freedom and www.freescale.com/tower.

- **GPIO_DRV_ClearPinIntFlag(), GPIO_DRV_TogglePinOutput():** These are also functions of the GPIO driver which clear the interrupt flag and toggle the GPIO value.

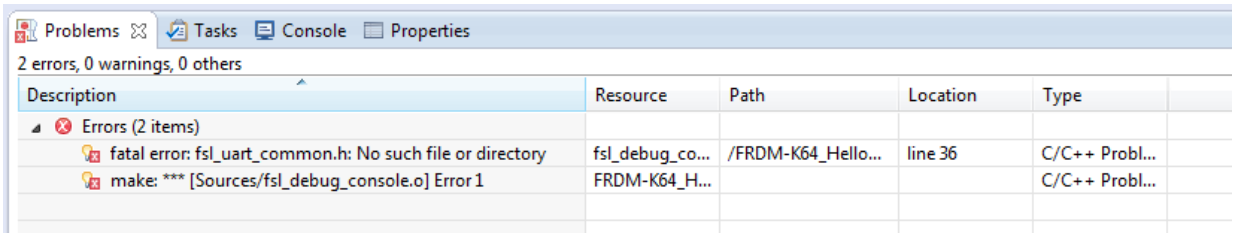
3.10 Add the following code into ***'board.h'*** to generate a delay for the push button debounce. **DO NOT ERASE** anything in this file.

```
#define GPIO_SW_DELAY \
    do \
    { \
        int32_t i; \
        for (i = 0; i < 0x1FFFFF; i++) \
        { \
            __asm("nop"); \
        } \
    } while (0)
```

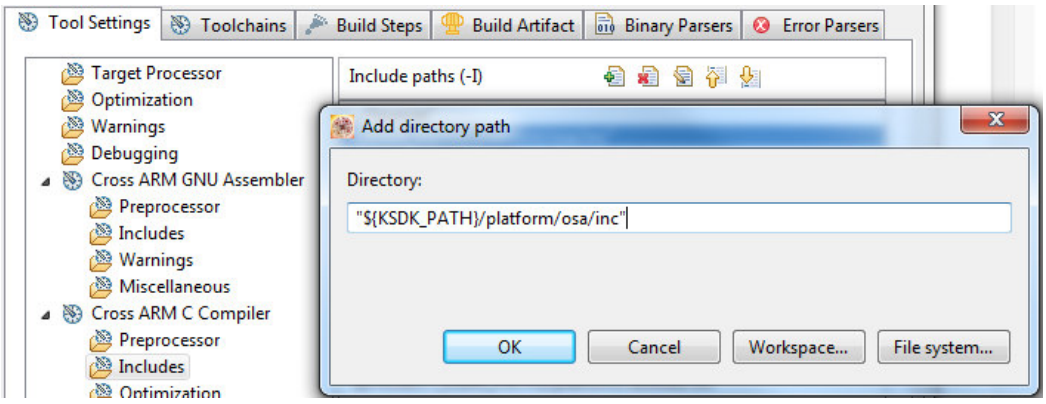
3.11 Build your application, go to ***menu Project > Build Project***. Alternately click the hammer button.



3.12 You will find some errors.

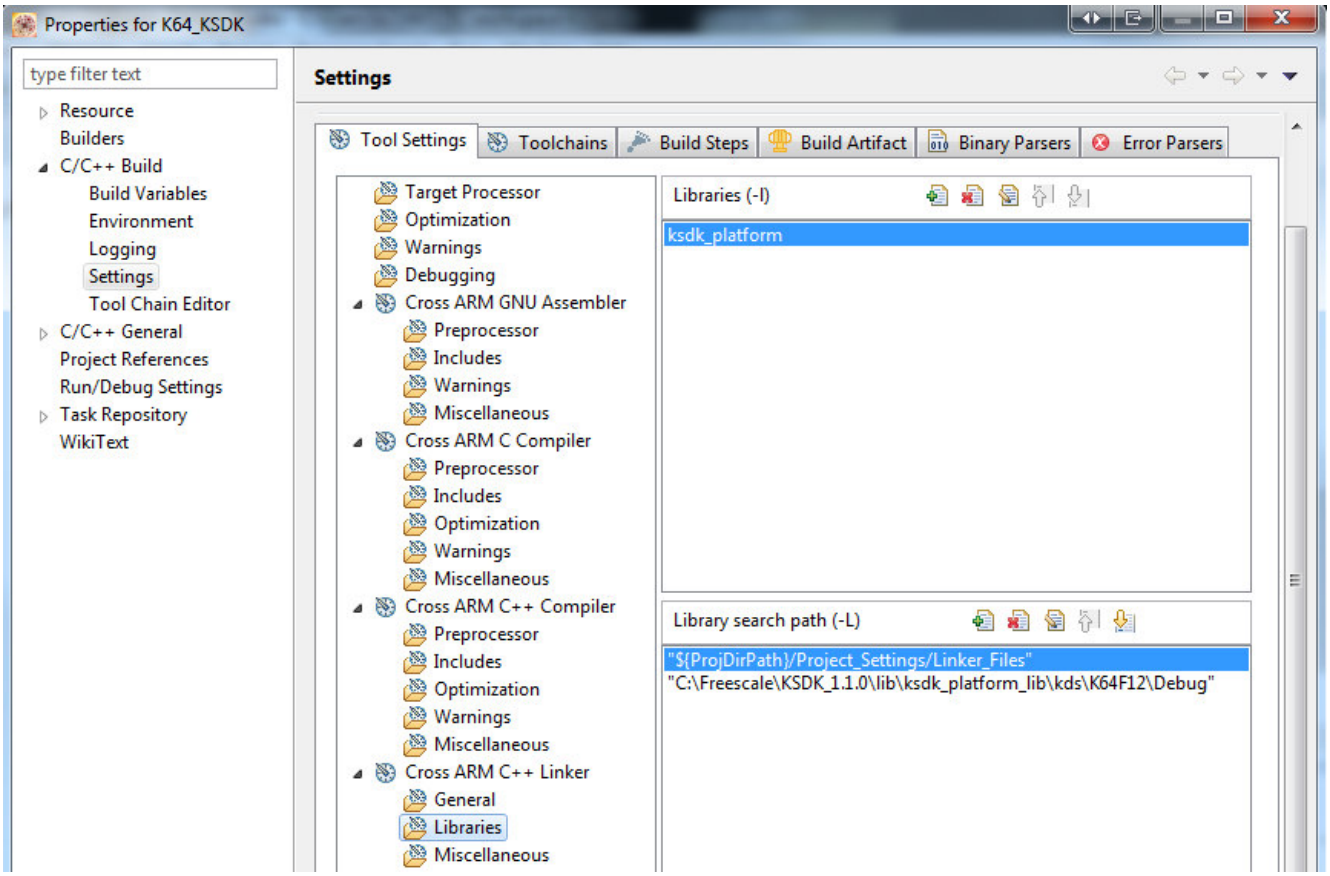


3.13 Go to ***menu Project > Properties C/C++ Build > Settings > Cross ARM C Compiler > Includes*** and add to ***'Include paths (-I)'*** the path ***"${KSDK_PATH}/platform/osa/inc"***. **DO NOT COPY-PASTE** this line because quotes cause problems when copy-pasting.



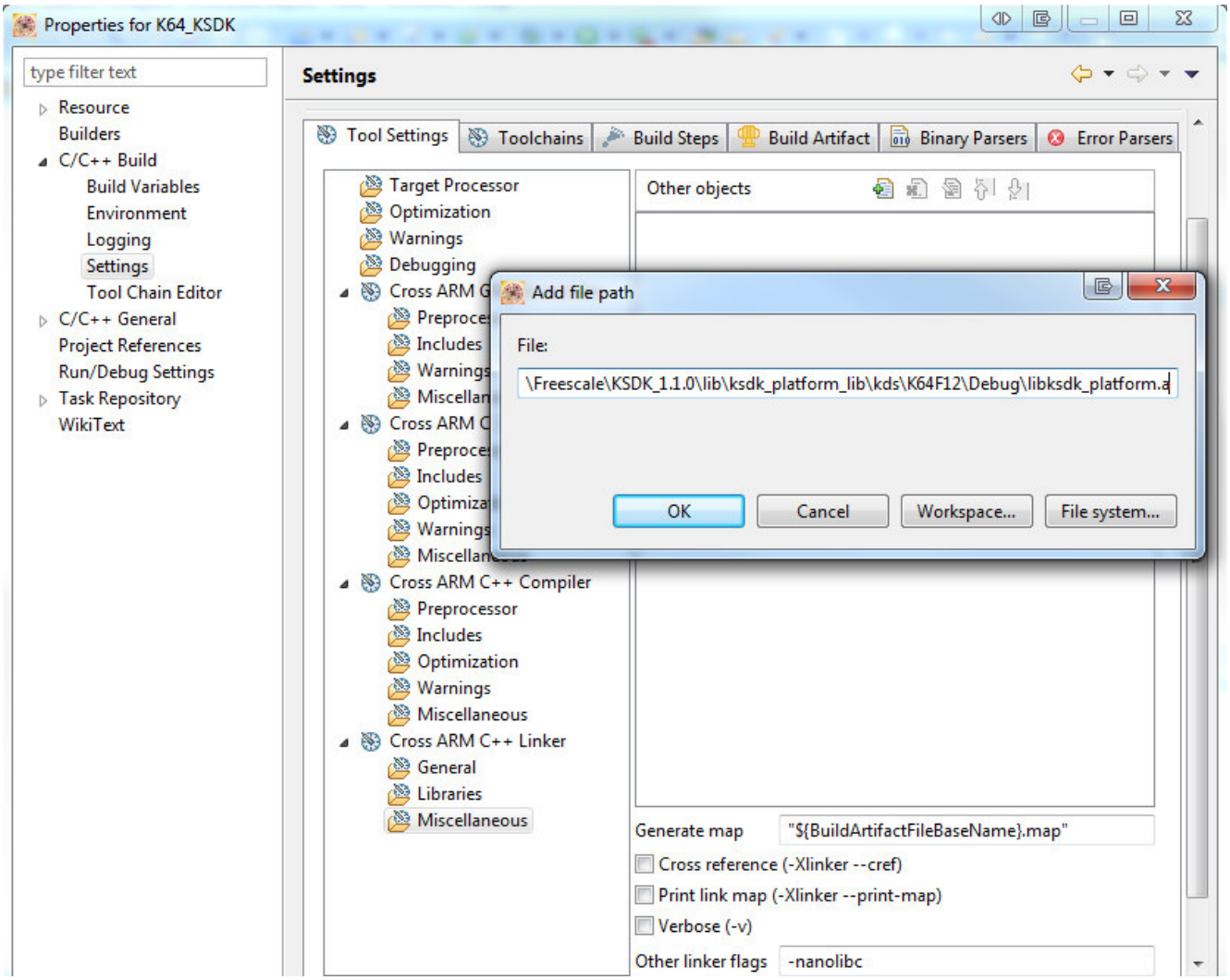
3.14 Repeat the same process in step 3.14 for 3 more paths, ***"${KSDK_PATH}/platform/drivers/inc", "${KSDK_PATH}/platform/system/inc"*** and ***"${KSDK_PATH}/platform/hal/inc"***.

3.15 Go to ***menu Project > Properties C/C++ Build > Settings > Cross ARM C++ Linker > Libraries*** and click the **'+'** to add in ***'Libraries (-l)' 'ksdk_platform'*** (without "lib" and without ".a"). Then in ***'Library search path'*** *add* ***"C:\Freescale\KSDK_1.1.0\lib\ksdk_platform_lib\kds\K64F12\Debug"***



NOTE: At this point '***libksdk_platform***'must be built, please refer to step ***1.5 - It is necessary to build KSDK Platform Driver Library (libksdk_platform.a)'*** on this document.

Alternately you can go to ***menu Project > Properties C/C++ Build > Settings > Cross ARM C++ Linker > Miscellaneous*** and click the ***'+'*** and then ***'File System'*** button to add in ***'Other objects'*** '***libksdk_platform.a'*** located in the next path: ***C:\Freescale\KSDK_1.1.0\lib\ksdk_platform_lib\kds\K64F12\Debug\ libksdk_platform.a.***

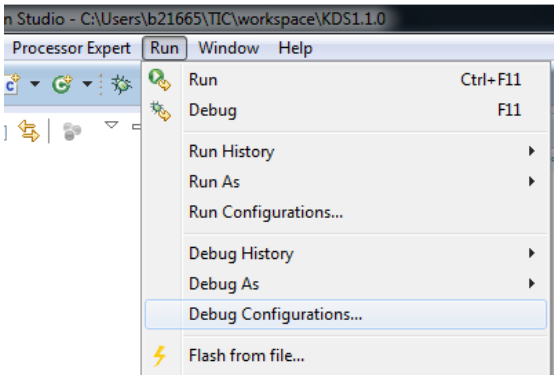


NOTE: If the file does not exist you need to build KSDK platform library. Please refer to step ***1.5 - It is necessary to build KSDK Platform Driver Library (libksdk_platform.a)'*** on this document.

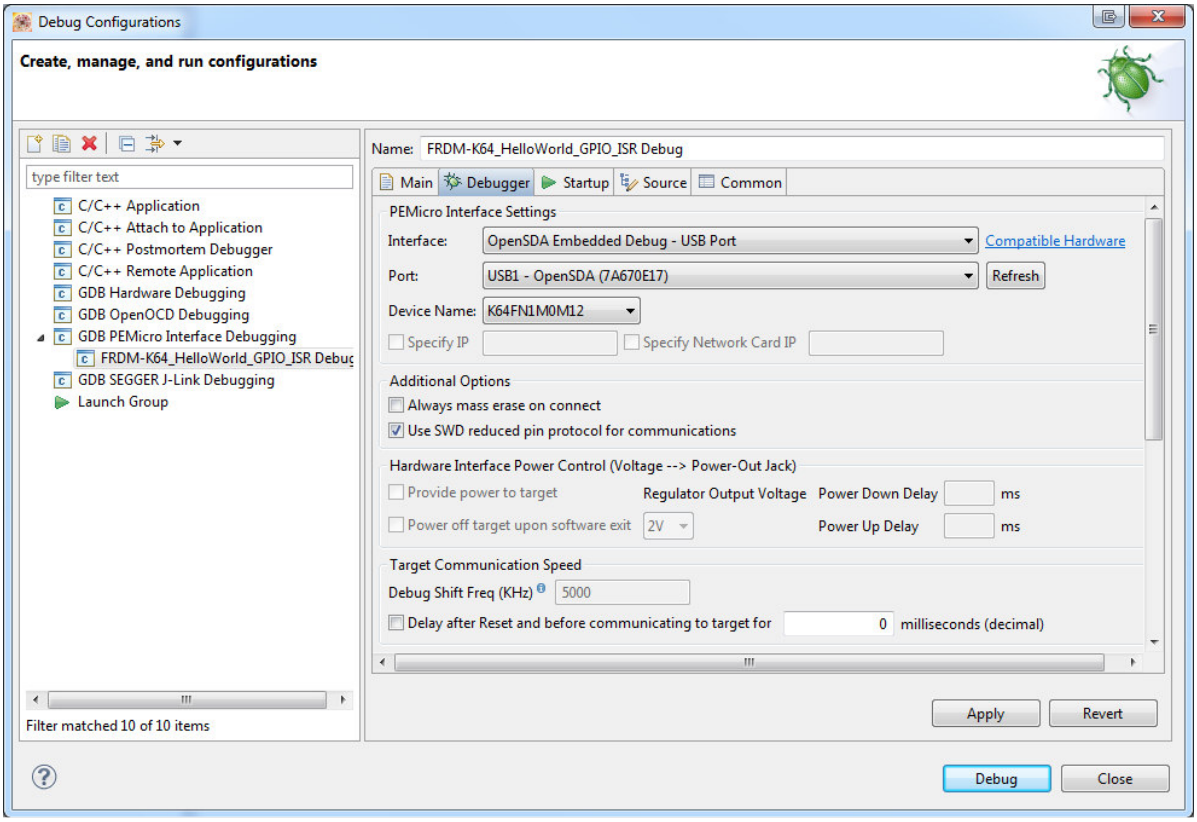
3.16 Build the project again (refer to step 3.12).

4 Debug the application

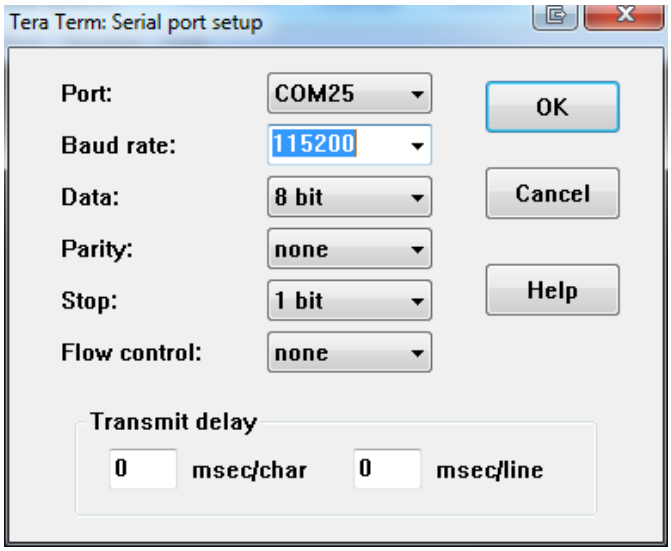
4.1 Go to menu ***Run > Debug Configurations…***



4.2 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'***.

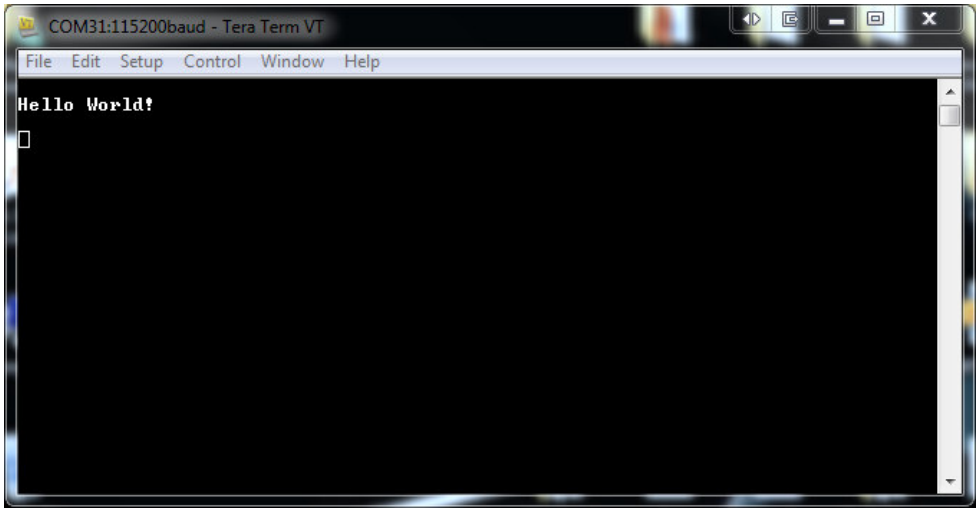


4.3 Open a terminal, select the appropriate port and set baudrate to 115200.



4.4 Run the application, you will see ***"Hello World"*** in terminal and blue LED will toggle when pressing SW1.

NOTE: In some evaluation boards SW1 does not match with the same button number in the board, if you don't see the LED toggling please try with a different button.

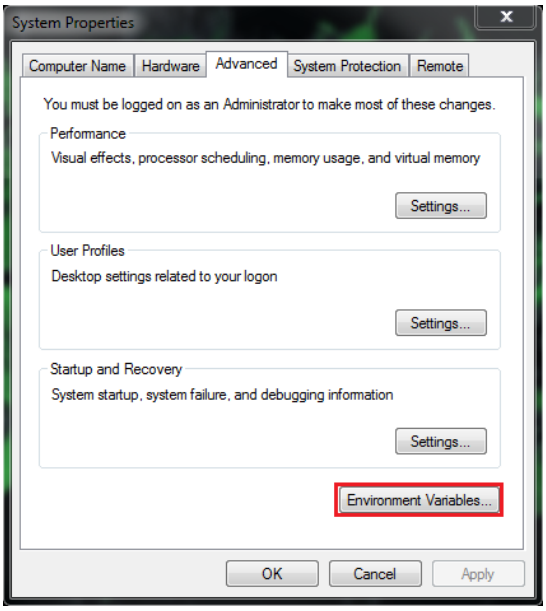


APPENDIX A: Install KDS Software Updates

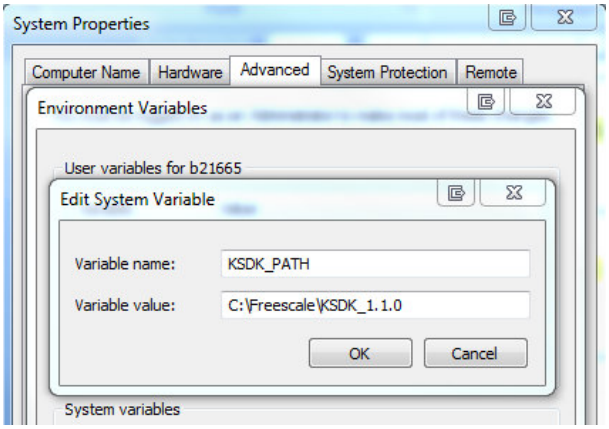
See chapter **5.2 Install Eclipse update** of ***'Getting Started with Kinetis SDK (KSDK)'*** document located in KSDK installation path.
C:\Freescale\KSDK_1.1.0\doc\Getting Started with Kinetis SDK (KSDK).pdf

APPENDIX B: Verify KSDK_PATH

Go to ***'Control Panel > System and Security > System > Advanced system settings'*** and ***'Environment Variables…'***.



1.1 If the variable does not exists click ***'New'*** button under ***'System variables'*** to create 'KSDK_PATH' variable. The path must be where KSDK is installed. The default path is ***'C:\Freescale\KSDK_1.1.0'***. If the variable exists but it does not point to your KSDK installation folder click ***'Edit'*** button to correct it.



IMPORTANT NOTE: If you installed more than one KSDK or MQX for KSDK versions you must be sure that KSDK_PATH variable is pointing to the correct installation.

APPENDIX C: Connection Types

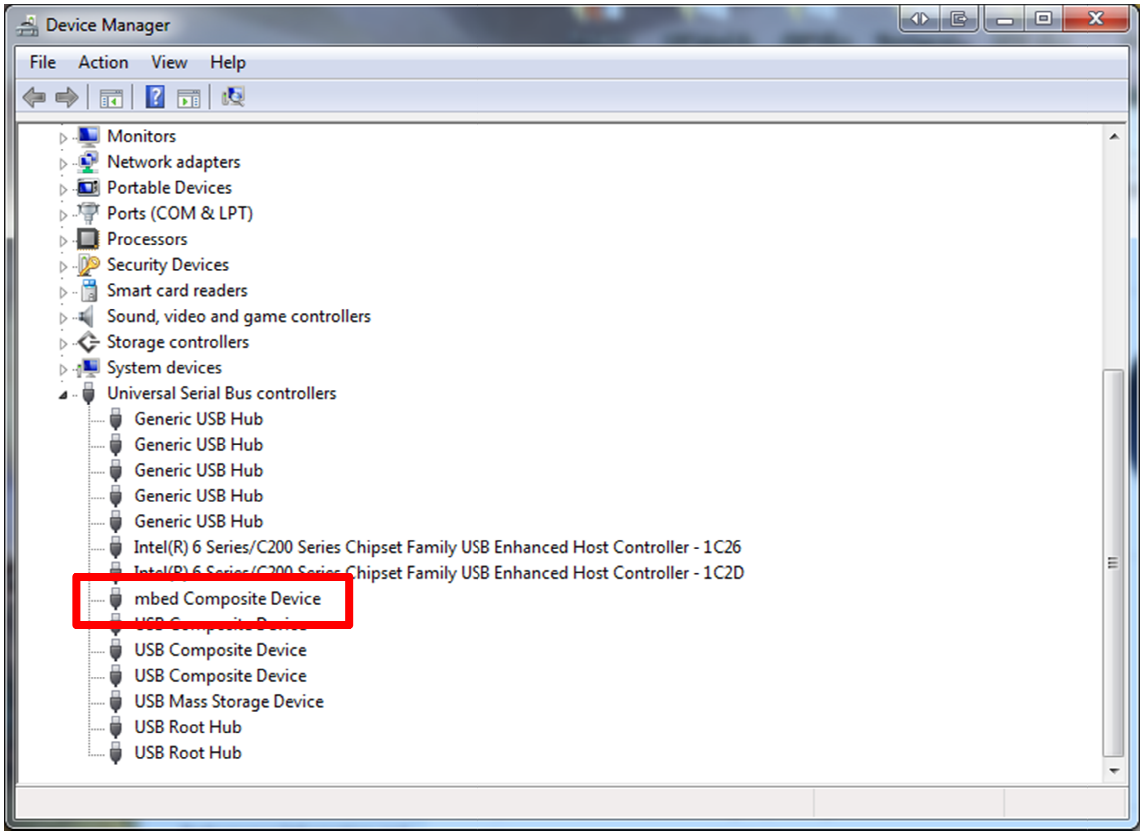
- KDS works with devices which support OpenSDAv2 connection.
- You can find Open SDA User's Guide here: http://www.freescale.com/files/32bit/doc/user_guide/OPENSDAUG.pdf
- You can learn more about Open SDA in the following link: https://community.freescale.com/docs/DOC-100720

Identify your Connection Type

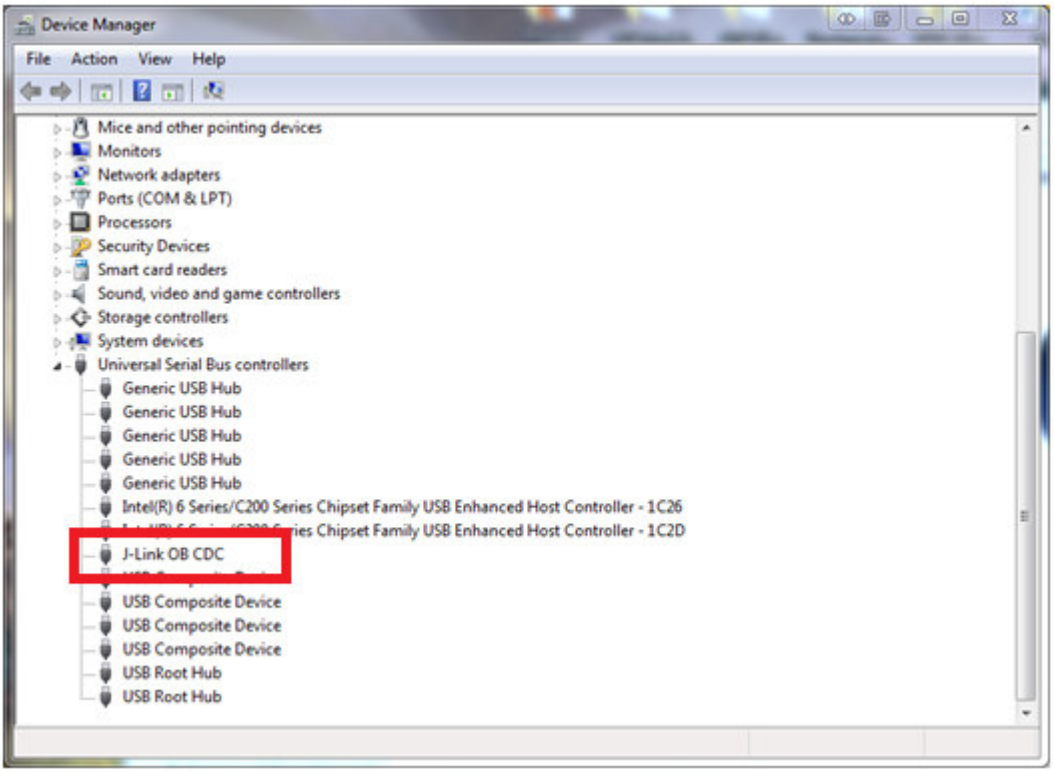
To find out which your connection type is you must connect your device to your computer and go to Windows Device Manager, here you can see the connection used by your device. You can see how to open Windows device manager in the link below:
http://windows.microsoft.com/en-us/windows/open-device-manager#1TC=windows-7

MBED Connection

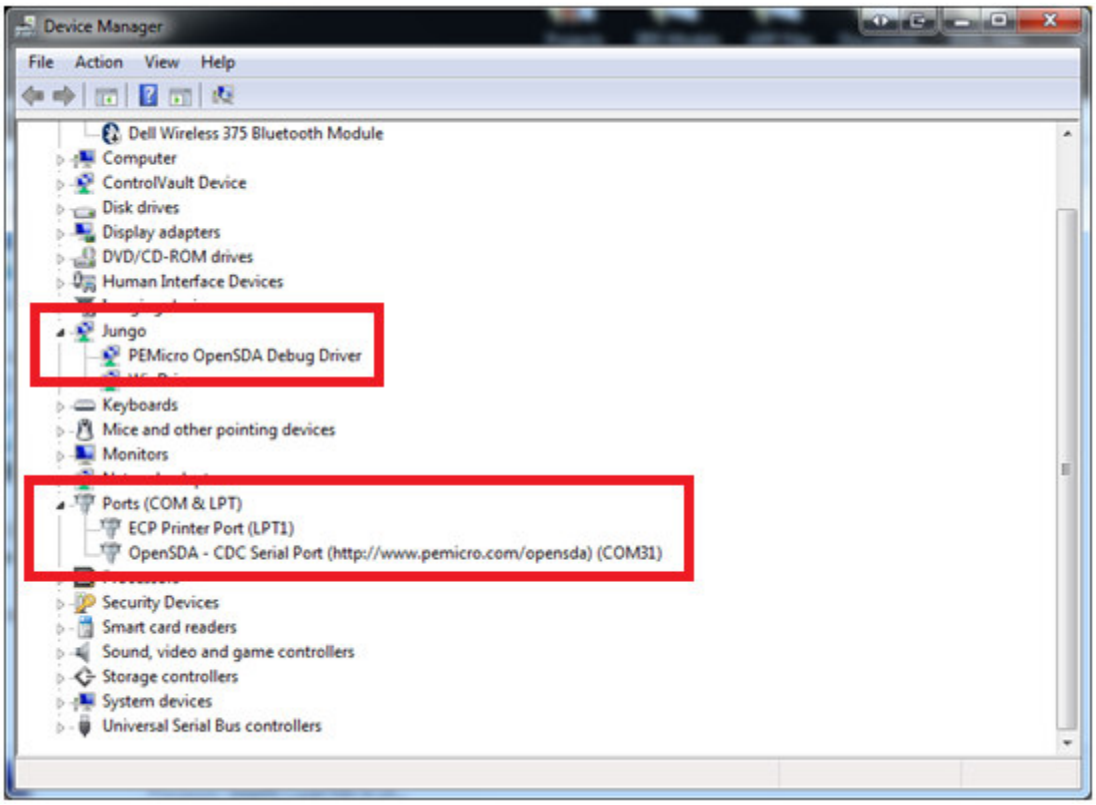
Please note that this connection is not supported in KDS yet.



Segger J-link connection



P&E Micro Connection

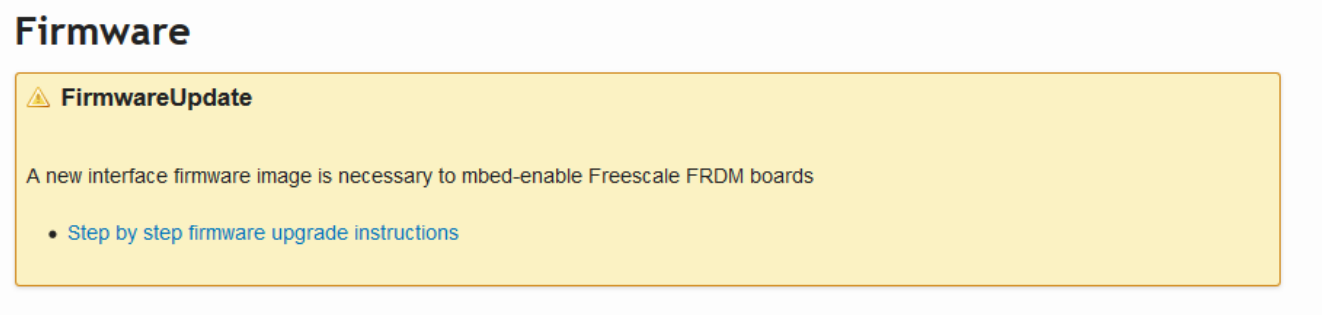


Switching or updating your connection firmware

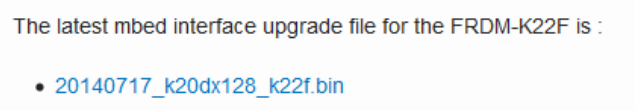
You can download different versions of OpenSDA form our partners' web sites.

MBED

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

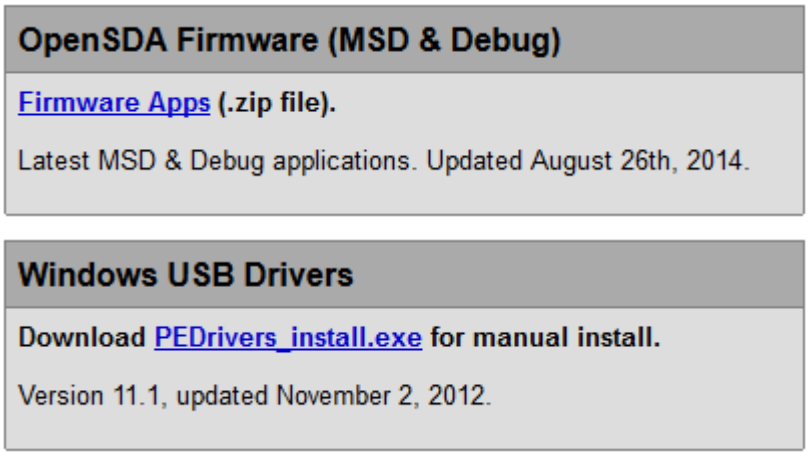


4) Save the latest firmware and follow the instructions to do the update



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/