NXP Semiconductor

Adding CMSIS-DSP Library to a KSDK 2.x project in Kinetis Design Studio

By: Technical Information Center

SECURE CONNECTIONS FOR A SMARTER WORLD

About this document

This document explains how to link the CMSIS-DSP library to a KSDK 2.x project in KDS.

The steps described in the document were done using the MK64FN1M0VLL12 MCU like the one in the FRDM-K64F board, but the same principles are applicable to any Kinetis MCU.

Software versions

The steps described in this document are valid for the following versions of the software tools:

- KDS v3.2.0
- KSDK 2.x

Contents

1. Glossary

KDS	***Kinetis Design Studio*: Integrated Development Environment (IDE) software for Kinetis MCUs.
KSDK	***Kinetis Software Development Kit*: Set of peripheral drivers, stacks and middleware layers for Kinetis microcontrollers.
CMSIS	**Cortex Microcontroller Software Interface Standard.** Hardware abstraction layer for the Cortex-M processor series.

2. Overview and concepts

2.1 CMSIS - Cortex Microcontroller Software Interface Standard

The ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and specifies debugger interfaces. Creation of software is a major cost factor in the embedded industry. By standardizing the software interfaces across all Cortex-M silicon vendor products, especially when creating new projects or migrating existing software to a new device, means significant cost reductions.
The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices

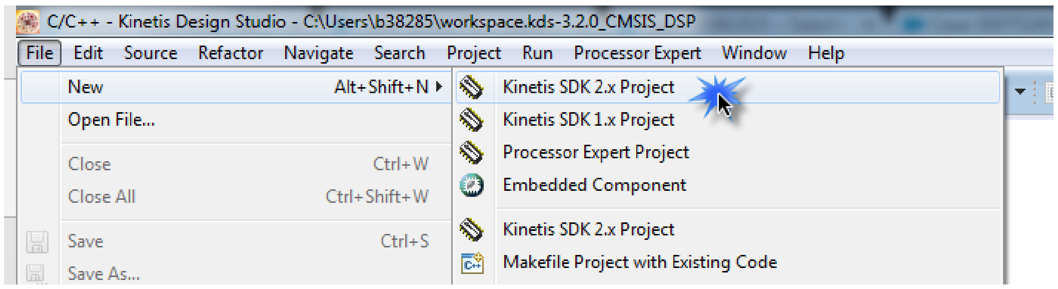
2.1.1 CMSIS Components

- **CMSIS-CORE**: API for the Cortex-M processor core and peripherals. It provides at standardized interface for Cortex-M0, Cortex-M3, Cortex-M4, SC000, and SC300. Included are also SIMD intrinsic functions for Cortex-M4 SIMD instructions.
- **CMSIS-Driver**: defines generic peripheral driver interfaces for middleware making it reusable across supported devices. The API is RTOS independent and connects microcontroller peripherals with middleware that implements for example communication stacks, file systems, or graphic user interfaces.
- **CMSIS-DSP**: DSP Library Collection with over 60 Functions for various data types: fix-point (fractional q7, q15, q31) and single precision floating-point (32-bit). The library is available for Cortex-M0, Cortex-M3, and Cortex-M4. The Cortex-M4 implementation is optimized for the SIMD instruction set.
- **CMSIS-RTOS API**: Common API for Real-Time operating systems. It provides a standardized programming interface that is portable to many RTOS and enables therefore software templates, middleware, libraries, and other components that can work across supported the RTOS systems.
- **CMSIS-Pack**: describes with a XML based package description (PDSC) file the user and device relevant parts of a file collection (called software pack) that includes source, header, and library files, documentation, Flash programming algorithms, source code templates, and example projects. Development tools and web infrastructures use the PDSC file to extract device parameters, software components, and evaluation board configurations.
- **CMSIS-SVD**: System View Description for Peripherals. Describes the peripherals of a device in an XML file and can be used to create peripheral awareness in debuggers or header files with peripheral register and interrupt definitions.
- **CMSIS-DAP**: Debug Access Port. Standardized firmware for a Debug Unit that connects to the CoreSight Debug Access Port. CMSIS-DAP is distributed as separate package and well suited for integration on evaluation boards. This component is provided as separate download.

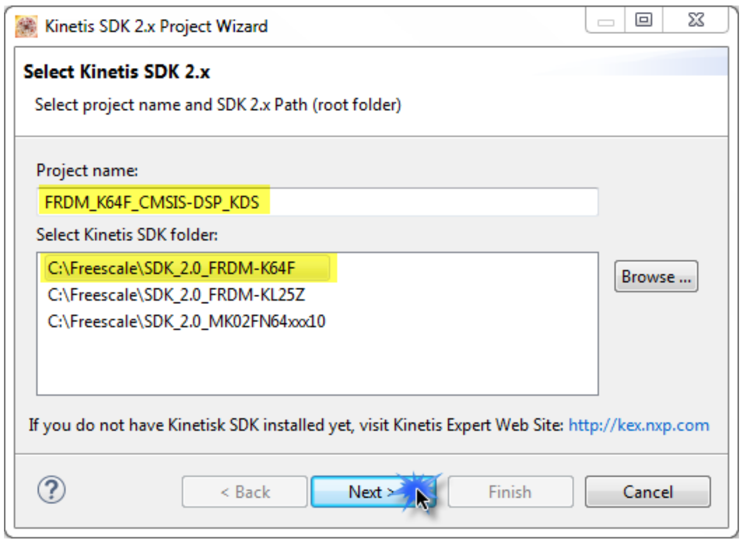
3. DSP example application

3.1 Creating KSDK 2.x Project

- Open **KDS** and click on **File > New > Kinetis SDK 2.x Project**:


C/C++ - Kinetis Design Studio - C:\Users\b38285\workspace.kds-3.2.0_CMSIS_DSP
File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Kinetis SDK 2.x Project
Kinetis SDK 1.x Project
Processor Expert Project
Embedded Component
Kinetis SDK 2.x Project
Makefile Project with Existing Code

- Give a name to the project, select the corresponding Kinetis SDK folder and click on **Next**:


Kinetis SDK 2.x Project Wizard

Select Kinetis SDK 2.x

Select project name and SDK 2.x Path (root folder)

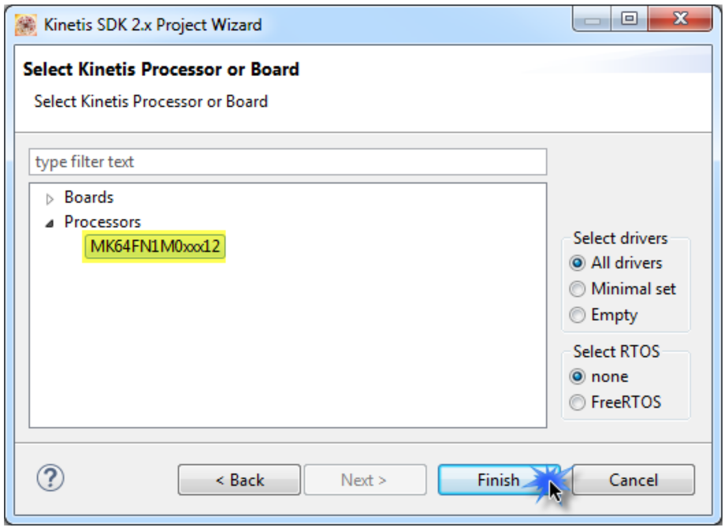
Project name: FRDM_K64F_CMSIS-DSP_KDS

Select Kinetis SDK folder:

C:\Freescale\SDK_2.0_FRDM-K64F
C:\Freescale\SDK_2.0_FRDM-KL25Z
C:\Freescale\SDK_2.0_MK02FN64xxx10

If you do not have Kinetisk SDK installed yet, visit Kinetis Expert Web Site: http://kex.nxp.com

– Select the **processor** to be used and click on **Finish**:


Kinetis SDK 2.x Project Wizard

Select Kinetis Processor or Board

Select Kinetis Processor or Board

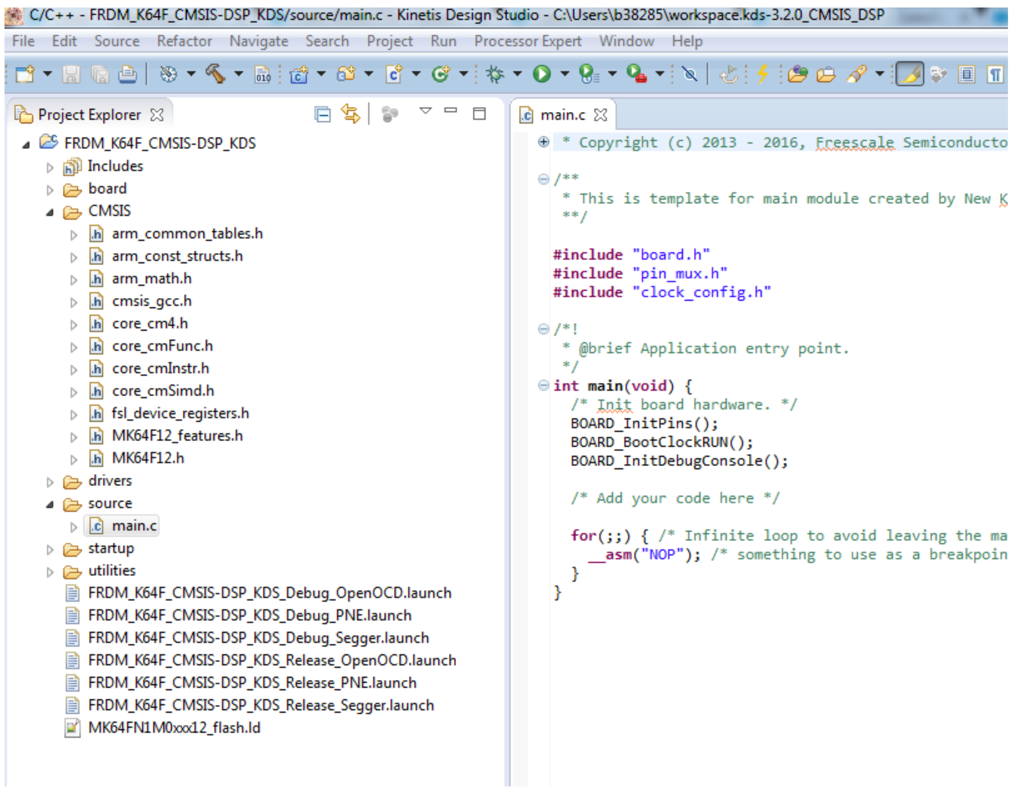
type filter text

- Boards
- Processors
  - MK64FN1M0xxx12

Select drivers: [✓ All drivers] Minimal set Empty

Select RTOS: [✓ none] FreeRTOS

– The new project should now appear on your workspace:


C/C++ - FRDM_K64F_CMSIS-DSP_KDS/source/main.c - Kinetis Design Studio - C:\Users\b38285\workspace.kds-3.2.0_CMSIS_DSP

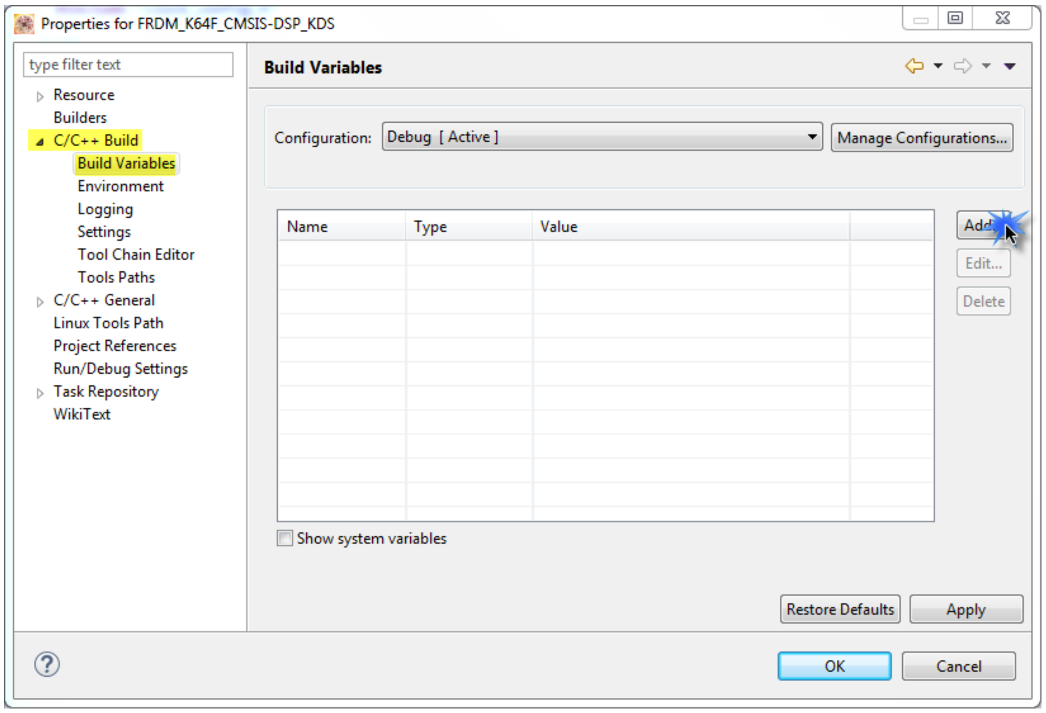
File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Project Explorer
main.c

```
FRDM_K64F_CMSIS-DSP_KDS
  Includes
  board
  CMSIS
    arm_common_tables.h
    arm_const_structs.h
    arm_math.h
    cmsis_gcc.h
    core_cm4.h
    core_cmFunc.h
    core_cmInstr.h
    core_cmSimd.h
    fsl_device_registers.h
    MK64F12_features.h
    MK64F12.h
  drivers
  source
    main.c
  startup
  utilities
  FRDM_K64F_CMSIS-DSP_KDS_Debug_OpenOCD.launch
  FRDM_K64F_CMSIS-DSP_KDS_Debug_PNE.launch
  FRDM_K64F_CMSIS-DSP_KDS_Debug_Segger.launch
  FRDM_K64F_CMSIS-DSP_KDS_Release_OpenOCD.launch
  FRDM_K64F_CMSIS-DSP_KDS_Release_PNE.launch
  FRDM_K64F_CMSIS-DSP_KDS_Release_Segger.launch
  MK64FN1M0xxx12_flash.ld
```

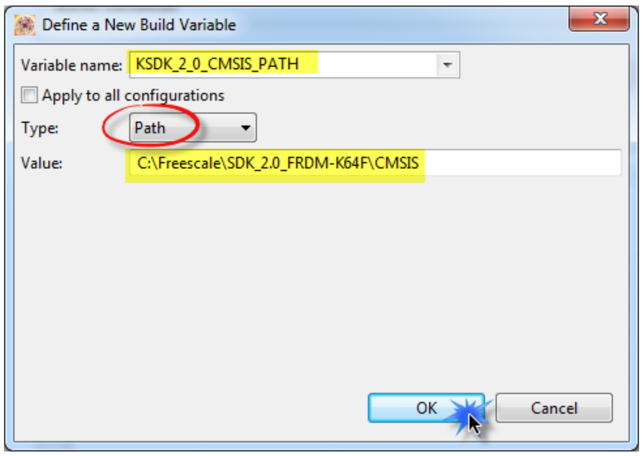
```c
 * Copyright (c) 2013 - 2016, Freescale Semiconducto

/**
 * This is template for main module created by New K
 **/

#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"

/*!
 * @brief Application entry point.
 */
int main(void) {
  /* Init board hardware. */
  BOARD_InitPins();
  BOARD_BootClockRUN();
  BOARD_InitDebugConsole();

  /* Add your code here */

  for(;;) { /* Infinite loop to avoid leaving the ma
    __asm("NOP"); /* something to use as a breakpoin
  }
}
```

3.2 Linking CMSIS-DSP Library

– The first step is to create a build variable that will be used to specify the path of the DSP library and its included folders. Go to **Project > Properties and under C/C++ Build** select **Build Variables** and click on **Add**:



– A new window will open, specify the **name** of the build variable, its **type** and **value**, the Value is the location of your CMSIS folder:

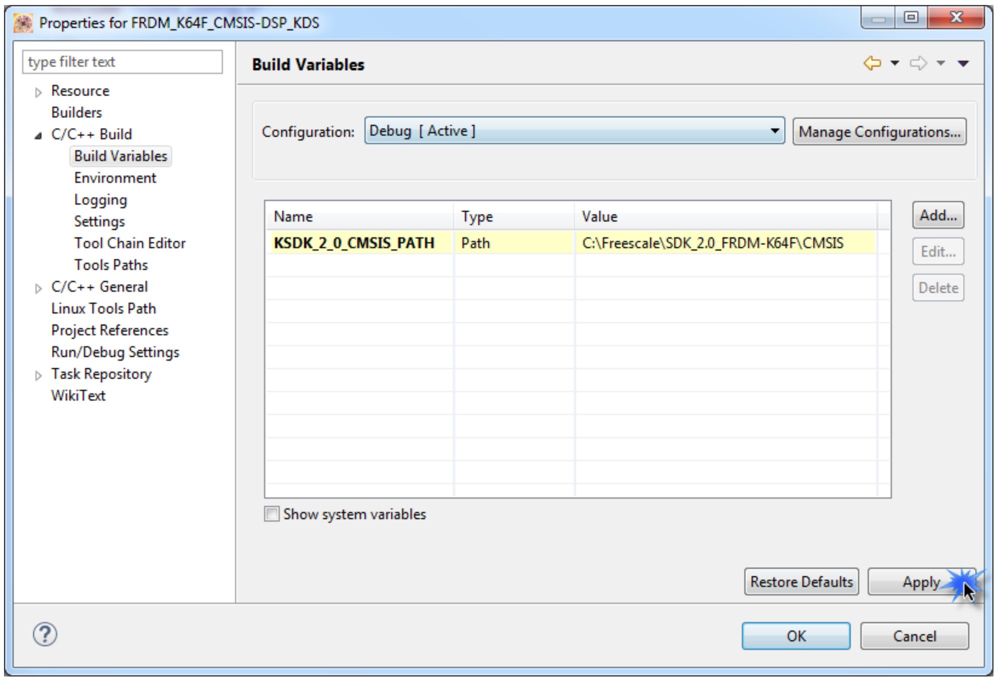


Variable name: KSDK_2_0_CMSIS_PATH

Type: Path

Value: C:\Freescale\SDK_2.0_FRDM-K64F\CMSIS

– The new variable should be listed as in the image below, click on **Apply**:



Configuration: Debug [Active]

Name	Type	Value
KSDK_2_0_CMSIS_PATH	Path	C:\Freescale\SDK_2.0_FRDM-K64F\CMSIS

☐ Show system variables

– The next step is to include the CMSIS-DSP library paths. Go to **C/C++ Build > Settings > Cross ARM C Compiler > Includes** and add the following paths then click on **Apply**:

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\BasicMathFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\CommonTables"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\ComplexMathFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\ControllerFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\FastMathFunctions"

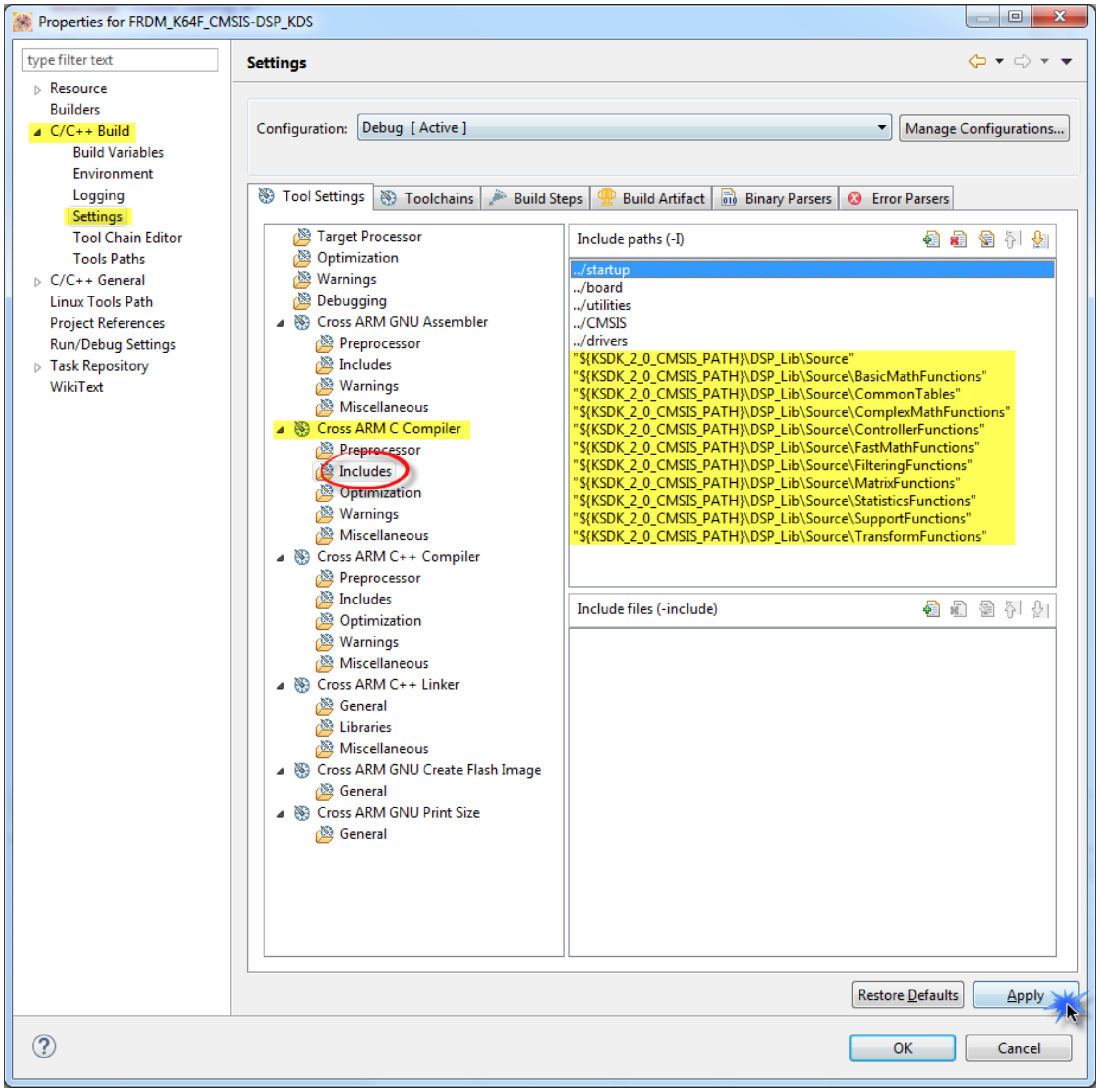
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\FilteringFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\MatrixFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\StatisticsFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\SupportFunctions"

"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\TransformFunctions"

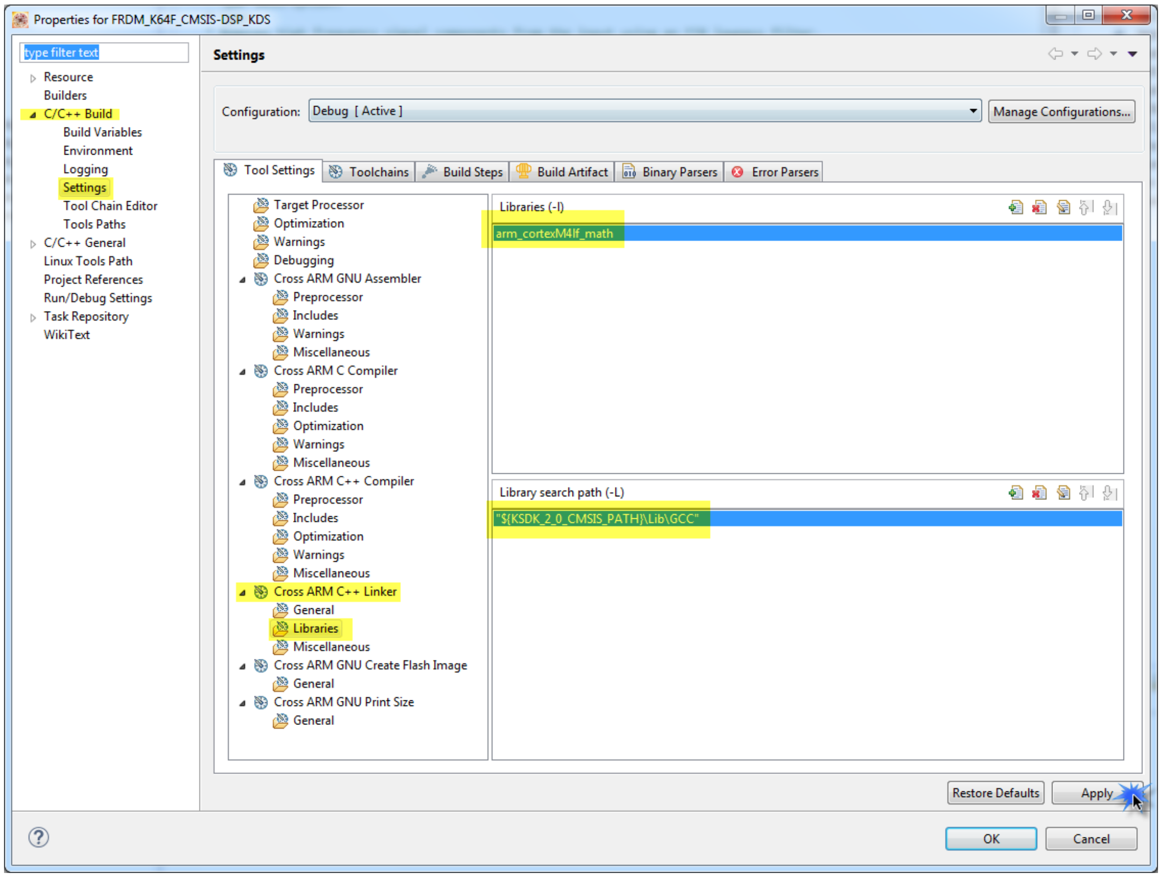


Configuration: Debug [Active]

Include paths (-I)

```
../startup
../board
../utilities
../CMSIS
../drivers
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\BasicMathFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\CommonTables"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\ComplexMathFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\ControllerFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\FastMathFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\FilteringFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\MatrixFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\StatisticsFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\SupportFunctions"
"${KSDK_2_0_CMSIS_PATH}\DSP_Lib\Source\TransformFunctions"
```

– After that, go to **C/C++ Build > Settings > Cross ARM C++ Linker > Libraries** and specify the precompiled library to be used and its path:



Configuration: Debug [Active]

Libraries (-l): arm_cortexM4lf_math

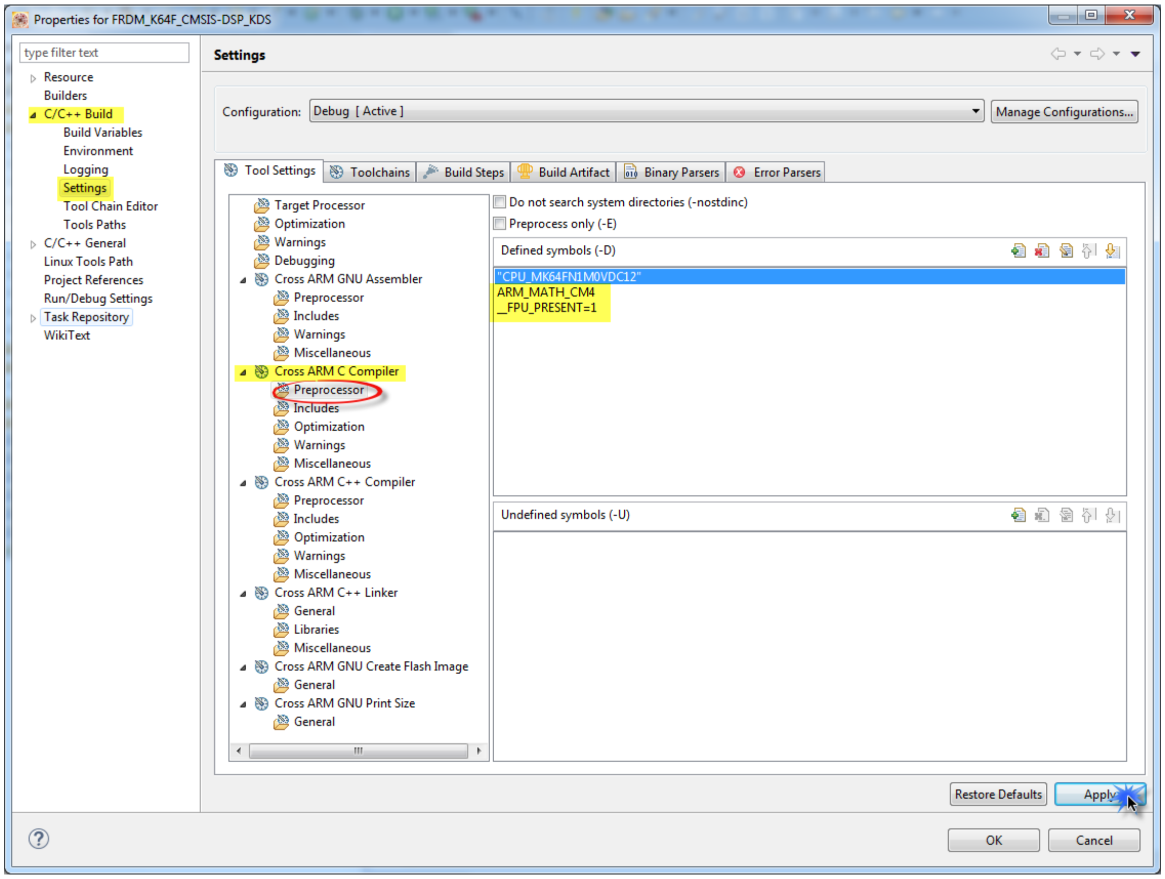
Library search path (-L): "${KSDK_2_0_CMSIS_PATH}\Lib\GCC"

Library name: *arm_cortexM4lf_math.* The M denotes the ARM core, while the 'l' means 'little endian'. The 'f' means an ARM core with Harware Floating Point Unit.

Path: *${KSDK_2_0_CMSIS_PATH}\Lib\GCC*

– Now go to **C/C++ Build > Settings > Cross ARM C Compiler > Preprocessor** and specify the following macros:

ARM_MATH_CM4: Tells the CMSIS library which ARM Cortex core I'm using.
__FPU_PRESENT=1: Needs to be specified when building on FPU supported Targets.



Configuration: Debug [Active]

Do not search system directories (-nostdinc)

Preprocess only (-E)

Defined symbols (-D)

```
"CPU_MK64FN1M0VDC12"
ARM_MATH_CM4
__FPU_PRESENT=1
```

Undefined symbols (-U)

– Finally click on **Apply** then **OK**.

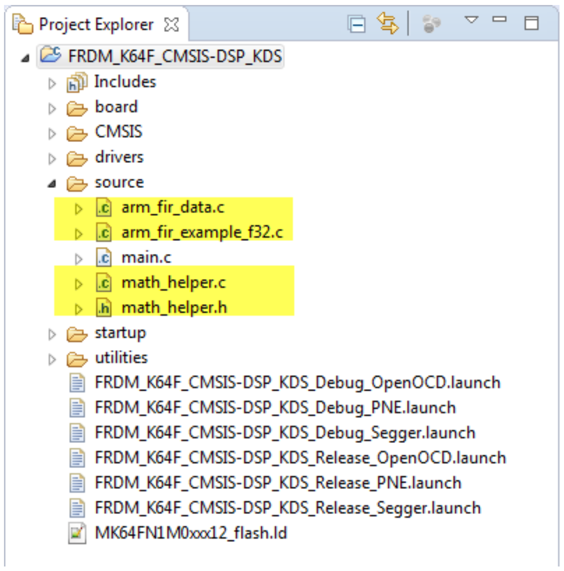
3.2 Importing DSP example source files

– For this project the "**FIR Lowpass Filter**" example will be used, it can be found on the following path:

  "{Your KSDK installation folder}\CMSIS\DSP_Lib\Examples\arm_fir_example\ARM"

– The first step is to copy the source files of the example to the project, the files that need to be copied are:

  arm_fir_example_f32.c
  arm_fir_data.c
  math_helper.c
  math_helper.h


Project Explorer
FRDM_K64F_CMSIS-DSP_KDS
Includes
board
CMSIS
drivers
source
arm_fir_data.c
arm_fir_example_f32.c
main.c
math_helper.c
math_helper.h
startup
utilities
FRDM_K64F_CMSIS-DSP_KDS_Debug_OpenOCD.launch
FRDM_K64F_CMSIS-DSP_KDS_Debug_PNE.launch
FRDM_K64F_CMSIS-DSP_KDS_Debug_Segger.launch
FRDM_K64F_CMSIS-DSP_KDS_Release_OpenOCD.launch
FRDM_K64F_CMSIS-DSP_KDS_Release_PNE.launch
FRDM_K64F_CMSIS-DSP_KDS_Release_Segger.launch
MK64FN1M0xxx12_flash.ld

- The next step is to delete the **main.c** file of our project but first we need to copy the SDK include files and initialization functions to the FIR example file:

  #include "board.h"

  #include "pin_mux.h"

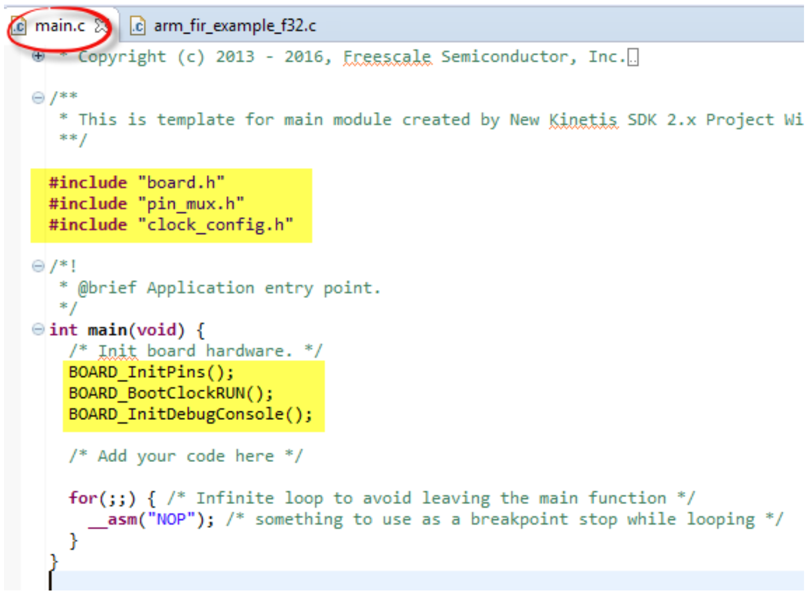
  #include "clock_config.h"

  BOARD_InitPins();

  BOARD_BootClockRUN();

  BOARD_InitDebugConsole();

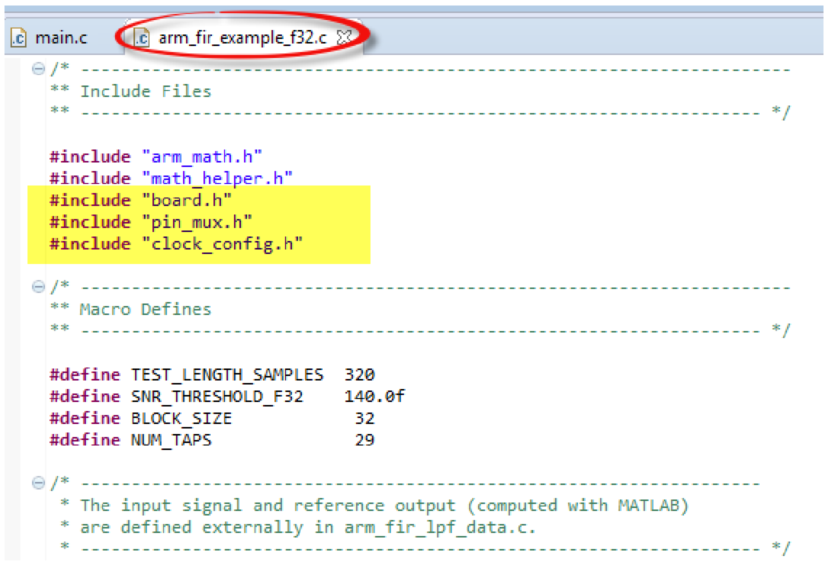
- Include files and initialization functions on the **main.c** file:



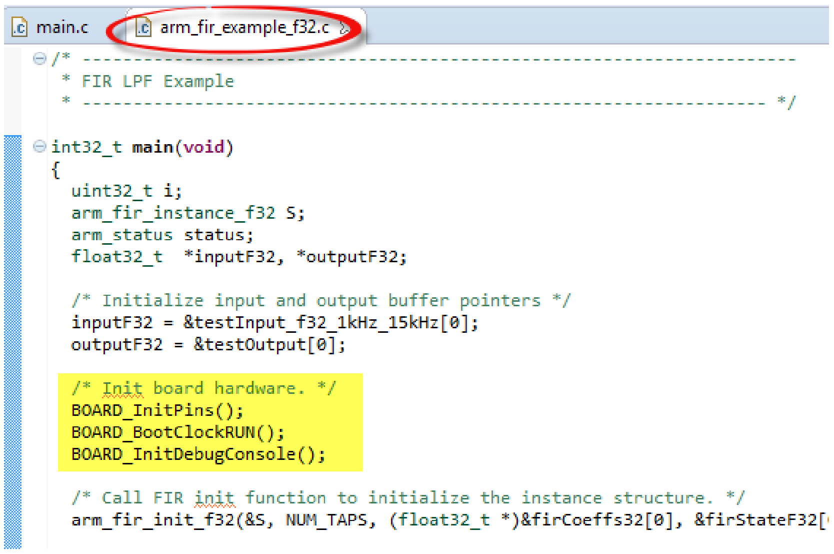
main.c | arm_fir_example_f32.c

```
 * Copyright (c) 2013 - 2016, Freescale Semiconductor, Inc.

/**
 * This is template for main module created by New Kinetis SDK 2.x Project Wi
 **/

#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"

/*!
 * @brief Application entry point.
 */
int main(void) {
  /* Init board hardware. */
  BOARD_InitPins();
  BOARD_BootClockRUN();
  BOARD_InitDebugConsole();

  /* Add your code here */

  for(;;) { /* Infinite loop to avoid leaving the main function */
    __asm("NOP"); /* something to use as a breakpoint stop while looping */
  }
}
```

– Modified **FIR example** with SDK include files and initialization functions:

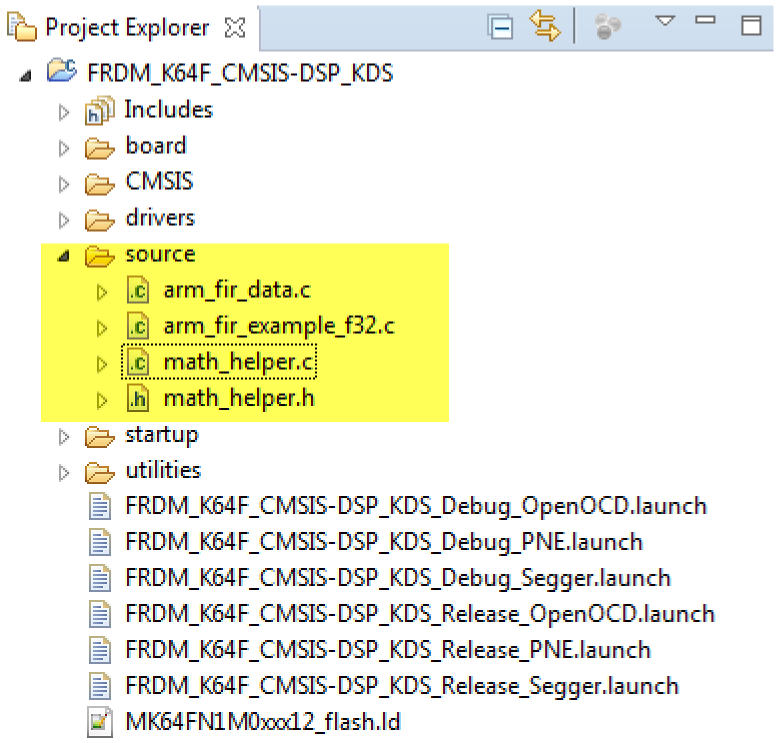

main.c | arm_fir_example_f32.c

```
/* ----------------------------------------------------------------------
** Include Files
** ------------------------------------------------------------------- */

#include "arm_math.h"
#include "math_helper.h"
#include "board.h"
#include "pin_mux.h"
#include "clock_config.h"

/* ----------------------------------------------------------------------
** Macro Defines
** ------------------------------------------------------------------- */

#define TEST_LENGTH_SAMPLES  320
#define SNR_THRESHOLD_F32    140.0f
#define BLOCK_SIZE            32
#define NUM_TAPS              29

/* -------------------------------------------------------------------
 * The input signal and reference output (computed with MATLAB)
 * are defined externally in arm_fir_lpf_data.c.
 * ------------------------------------------------------------------- */
```


main.c | arm_fir_example_f32.c

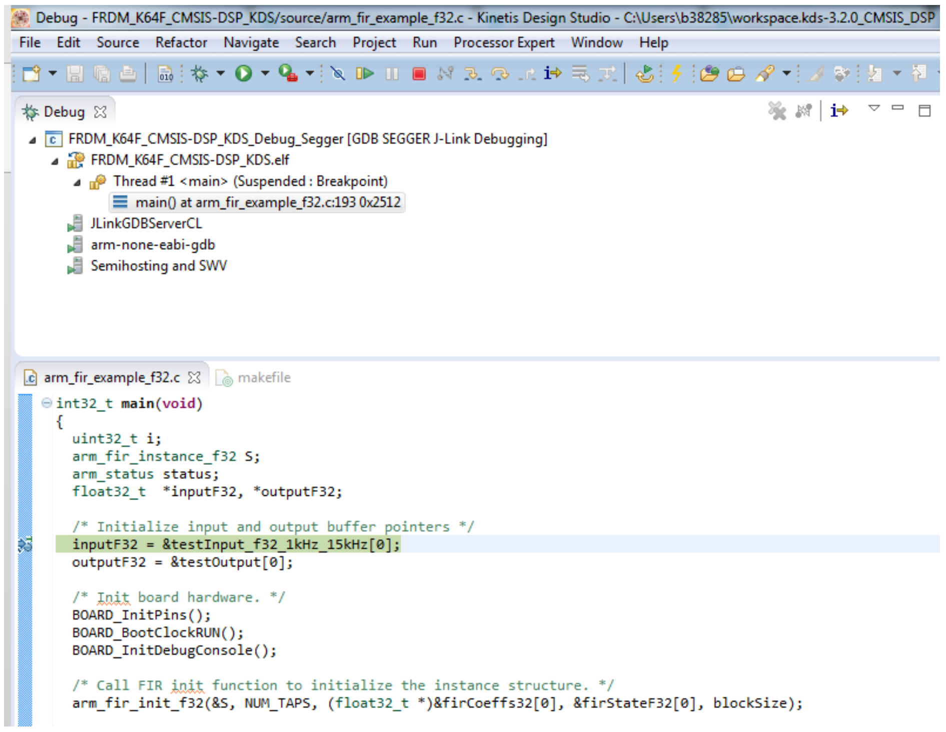
```
/* ----------------------------------------------------------------------
 * FIR LPF Example
 * ------------------------------------------------------------------- */

int32_t main(void)
{
  uint32_t i;
  arm_fir_instance_f32 S;
  arm_status status;
  float32_t  *inputF32, *outputF32;

  /* Initialize input and output buffer pointers */
  inputF32 = &testInput_f32_1kHz_15kHz[0];
  outputF32 = &testOutput[0];

  /* Init board hardware. */
  BOARD_InitPins();
  BOARD_BootClockRUN();
  BOARD_InitDebugConsole();

  /* Call FIR init function to initialize the instance structure. */
  arm_fir_init_f32(&S, NUM_TAPS, (float32_t *)&firCoeffs32[0], &firStateF32[
```

– Finally the **main.c** file can be deleted from the project:



```
FRDM_K64F_CMSIS-DSP_KDS
    Includes
    board
    CMSIS
    drivers
    source
        arm_fir_data.c
        arm_fir_example_f32.c
        math_helper.c
        math_helper.h
    startup
    utilities
    FRDM_K64F_CMSIS-DSP_KDS_Debug_OpenOCD.launch
    FRDM_K64F_CMSIS-DSP_KDS_Debug_PNE.launch
    FRDM_K64F_CMSIS-DSP_KDS_Debug_Segger.launch
    FRDM_K64F_CMSIS-DSP_KDS_Release_OpenOCD.launch
    FRDM_K64F_CMSIS-DSP_KDS_Release_PNE.launch
    FRDM_K64F_CMSIS-DSP_KDS_Release_Segger.launch
    MK64FN1M0xxx12_flash.ld
```

– Now you should be able to **compile** and **debug** the project:


Debug - FRDM_K64F_CMSIS-DSP_KDS/source/arm_fir_example_f32.c - Kinetis Design Studio - C:\Users\b38285\workspace.kds-3.2.0_CMSIS_DSP
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Processor Expert
Window
Help
Debug
arm_fir_example_f32.c
makefile

```
FRDM_K64F_CMSIS-DSP_KDS_Debug_Segger [GDB SEGGER J-Link Debugging]
    FRDM_K64F_CMSIS-DSP_KDS.elf
        Thread #1 <main> (Suspended : Breakpoint)
            main() at arm_fir_example_f32.c:193 0x2512
    JLinkGDBServerCL
    arm-none-eabi-gdb
    Semihosting and SWV
```

```c
int32_t main(void)
{
  uint32_t i;
  arm_fir_instance_f32 S;
  arm_status status;
  float32_t  *inputF32, *outputF32;

  /* Initialize input and output buffer pointers */
  inputF32 = &testInput_f32_1kHz_15kHz[0];
  outputF32 = &testOutput[0];

  /* Init board hardware. */
  BOARD_InitPins();
  BOARD_BootClockRUN();
  BOARD_InitDebugConsole();

  /* Call FIR init function to initialize the instance structure. */
  arm_fir_init_f32(&S, NUM_TAPS, (float32_t *)&firCoeffs32[0], &firStateF32[0], blockSize);
```

Appendix A - References

- KDS webpage:
  www.nxp.com/kds

- KSDK webpage:
  www.nxp.com/ksdk

- MCU on Eclipse - Tutorial: Using the ARM CMSIS Library:
  https://mcuoneclipse.com/2013/02/14/tutorial-using-the-arm-cmsis-library/

- CMSIS - Cortex Microcontroller Software Interface Standard:
  http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php