eIQ: CMSIS-NN Porting Guide for MCUs

Revision 2

NXP Semiconductors
www.nxp.com

Contents

1 Document Overview 3

2 Considerations 3

3 Prerequisites 3

4 MCUXpresso IDE Porting for CIFAR-10 Demo 3

4.1 Cloning a Project 3
4.2 Source Files 5
4.3 MCUXpresso IDE Project Settings 9

5 MCUXpresso IDE Porting for Keyword Spotting Demo 11

5.1 Create a C++ Project 11
5.2 Source Files 13
5.3 Compiler IDE Settings 16

6 IAR Porting 19

6.1 Cloning a Project 19
6.2 Copying Source Files 19
6.3 IAR Compiler Settings 25

1 Document Overview

This document describes how to port eIQ CMSIS-NN examples found in the MCUXpresso SDK to a new MCU, such as the LPC55S69. It covers porting for both MCUXpresso IDE and IAR Embedded Workbench for ARM.

2 Considerations

Inferencing of a model can theoretically be done on almost any MCU, as the majority of operations simply consist of doing multiple and accumulate math calculations. There's no special hardware or module required to do inferencing. However high core clock speeds and fast memory can drastically reduce inference time. There also needs to be enough flash and RAM to store the model. It's these memory and performance constraints that will often be a limiting factor on which MCUs a particular model can be ported to and is very dependent on the particular model being used.

This document will cover porting eIQ for CMSIS-NN to the LPC55S69. The instructions in this document can be used to port eIQ for CMSIS-NN to other devices as well.

3 Prerequisites

The following items will be needed:

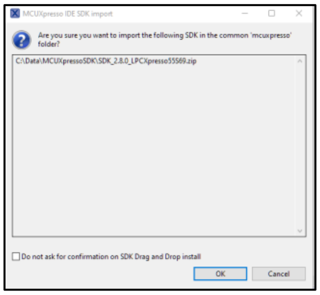
a) MCUXpresso SDK for LPC55S69
b) MCUXpresso SDK for RT1060 - make sure to include the "eIQ" middleware option.
c) MCUXpresso IDE – if using MCUXpresso IDE
d) MCUXpresso Config Tools – if using IAR

4 MCUXpresso IDE Porting for CIFAR-10 Demo

4.1 Cloning a Project

The first step is to create a new LPC55S69 project using the "Import SDK Example" feature in MCUXpresso IDE. The basic Hello World example will be used as a template to copy the eIQ CMSIS-NN source into.

1. Unzip both the LPC55S69 and the RT1060 SDK zip files into a directory path that does not contain spaces. The files inside will be used in later steps.
2. Open up MCUXpresso IDE and select a new workspace.
3. Install the LPC55S69 SDK into the "Installed SDKs" tab by dragging and dropping the zipped **SDK_2.8.0_LPCXpresso55S69.zip** file into the Installed SDKs window. This dialog box will come up, and click OK to continue the import

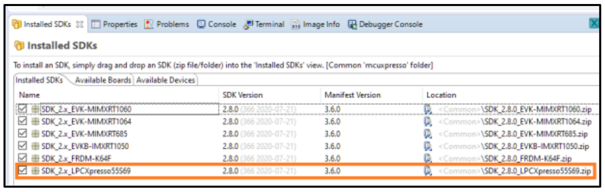


Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

C:\Data\MCUXpressoSDK\SDK_2.8.0_LPCXpresso55S69.zip

□ Do not ask for confirmation on SDK Drag and Drop install

4. It will look something like the following when complete:

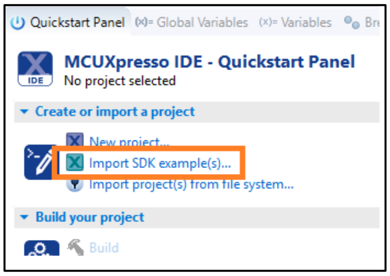


Installed SDKs

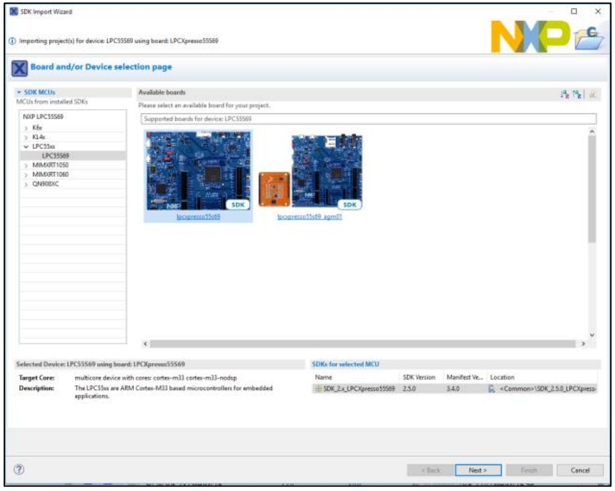
To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Common 'mcuxpresso' folder]

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVK-MIMXRT1060	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_EVK-MIMXRT1060.zip
☑ SDK_2.x_EVK-MIMXRT1064	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_EVK-MIMXRT1064.zip
☑ SDK_2.x_EVK-MIMXRT685	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_EVK-MIMXRT685.zip
☑ SDK_2.x_EVKB-IMXRT1050	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_EVKB-IMXRT1050.zip
☑ SDK_2.x_FRDM-K64F	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_FRDM-K64F.zip
☑ SDK_2.x_LPCXpresso55S69	2.8.0 (366 2020-07-21)	3.6.0	<Common>\SDK_2.8.0_LPCXpresso55S69.zip

5. Next import the desired project. In the Quickstart Panel, select **Import SDK examples(s)…**



6. Select the **lpcxpresso55s69** board and click on Next



Importing project(s) for device: LPC55S69 using board: LPCXpresso55S69

Please select an available board for your project.

Supported boards for device: LPC55S69

Selected Device: LPC55S69 using board: LPCXpresso55S69

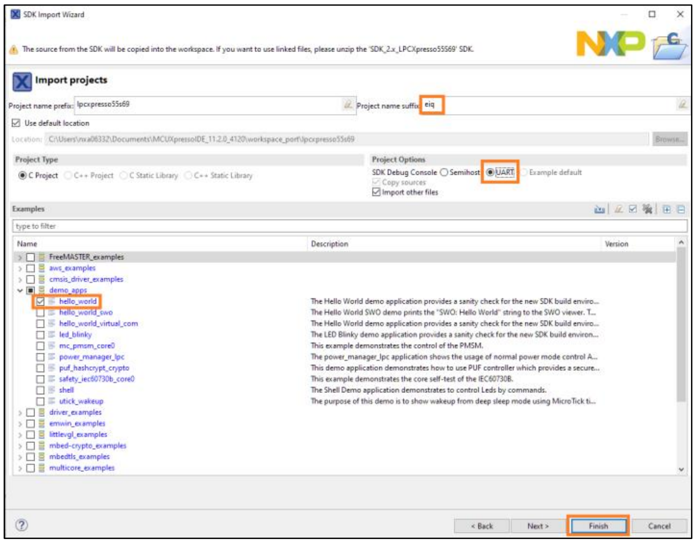
Target Core: multicore device with cores: cortex-m33 cortex-m33-nodsp

Description: The LPC55xx are ARM Cortex-M33 based microcontrollers for embedded applications.

SDKs for selected MCU

Name	SDK Version	Manifest Ve...	Location
SDK_2.x_LPCXpresso55S69	2.5.0	3.4.0	<Common>\SDK_2.5.0_LPCXpress

7. Open up the **demo_apps** category and select the **hello_world** project. In the **Project name suffix** box, change it to **eiq** to make it unique. Change the **SDK Debug Console** to **UART**. Then click on **Finish**.



The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_LPCXpresso55S69' SDK.

Project name prefix: lpcxpresso55s69 Project name suffix: eiq

☑ Use default location

Location: C:\Users\nxa06332\Documents\MCUXpressoIDE_11.2.0_4120\workspace_port\lpcxpresso55s69

Project Type: ● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options: SDK Debug Console ○ Semihost ● UART ○ Example default

☑ Copy sources

☑ Import other files

Examples

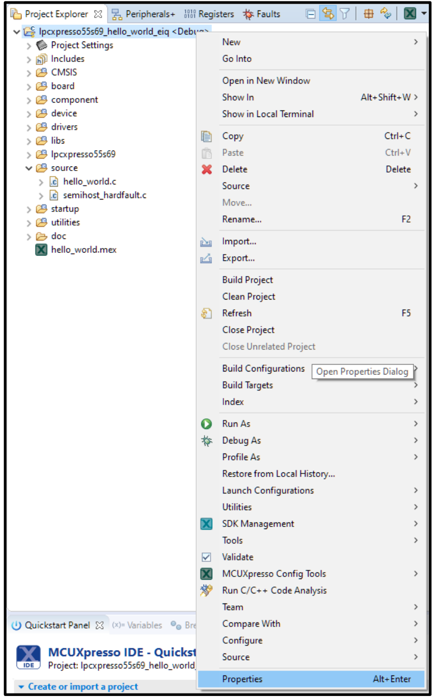
type to filter

Name	Description	Version
☐ FreeMASTER_examples		
☐ aws_examples		
☐ cmsis_driver_examples		
■ demo_apps		
☑ hello_world	The Hello World demo application provides a sanity check for the new SDK build enviro...	
☐ hello_world_swo	The Hello World SWO demo prints the "SWO: Hello World" string to the SWO viewer. T...	
☐ hello_world_virtual_com	The Hello World demo application provides a sanity check for the new SDK build enviro...	
☐ led_blinky	The LED Blinky demo application provides a sanity check for the new SDK build environ...	
☐ mc_pmsm_core0	This example demonstrates the control of the PMSM.	
☐ power_manager_lpc	The power_manager_lpc application shows the usage of normal power mode control A...	
☐ puf_hashcrypt_crypto	This demo application demonstrates how to use PUF controller which provides a secure...	
☐ safety_iec60730b_core0	This example demonstrates the core self-test of the IEC60730B.	
☐ shell	The Shell Demo application demonstrates to control Leds by commands.	
☐ utick_wakeup	The purpose of this demo is to show wakeup from deep sleep mode using MicroTick ti...	
☐ driver_examples		
☐ emwin_examples		
☐ littlevgl_examples		
☐ mbed-crypto_examples		
☐ mbedtls_examples		
☐ multicore_examples		

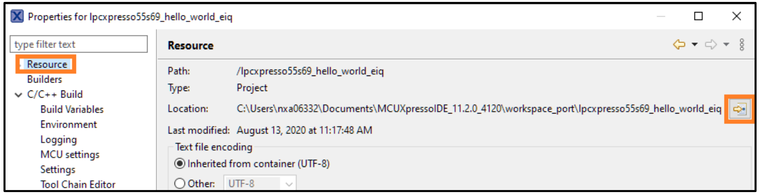
8. This will create a new project in your workspace that you can see in the project view

4.2 Source Files

9. If not done already, unzip the RT1060 SDK. The eIQ source files will be used in the next steps.
10. In MCUXpresso IDE, open the directory location of the project by right click on the project name and selecting Properties.



11. Then go to the **Resource** category and by the **Location**, click on the arrow to open that location in Windows Explorer



Resource

Path: /lpcxpresso55s69_hello_world_eiq

Type: Project

Location: C:\Users\nxa06332\Documents\MCUXpressoIDE_11.2.0_4120\workspace_port\lpcxpresso55s69_hello_world_eiq

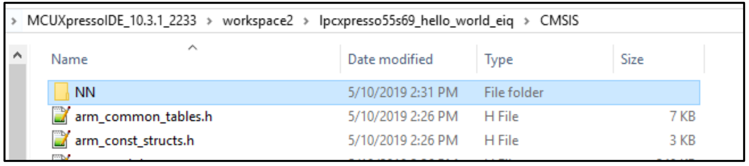
Last modified: August 13, 2020 at 11:17:48 AM

Text file encoding

● Inherited from container (UTF-8)

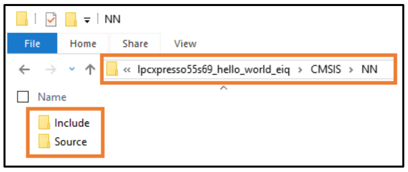
○ Other: UTF-8

12. In Windows Explorer, navigate to the **lpcxpresso55s69_hello_world_eiq** folder, and then the **CMSIS** folder. Then create a new folder inside the CMSIS folder named "**NN**". It will look like the following when complete:

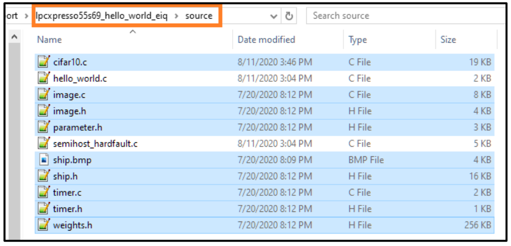

MCUXpressoIDE_10.3.1_2233 › workspace2 › lpcxpresso55s69_hello_world_eiq › CMSIS

Name	Date modified	Type	Size
NN	5/10/2019 2:31 PM	File folder	
arm_common_tables.h	5/10/2019 2:26 PM	H File	7 KB
arm_const_structs.h	5/10/2019 2:26 PM	H File	3 KB

13. If not done already, unzip the RT1060 SDK. The eIQ source files will be used in the next steps.
14. Inside the RT1060 SDK go to **\middleware\eiq\cmsis-nn\** Copy the "**Source**" and "**Include**" folders into the **NN** folder created in the previous step.
15. When finished it should look like the following:



16. Take the files in the RT1060 SDK **\middleware\eiq\cmsis-nn\Examples\cifar10** folder and copy them into the "Source" folder found at <project_location>.
17. Then also take the files in **\middleware\eiq\cmsis-nn\Examples\common\source** and copy them into the same "Source" folder as well at <project_location>. It will look like the following when complete:

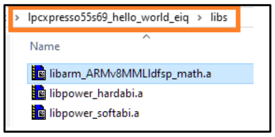


ort › lpcxpresso55s69_hello_world_eiq › source

Search source

Name	Date modified	Type	Size
cifar10.c	8/11/2020 3:46 PM	C File	19 KB
hello_world.c	8/11/2020 3:04 PM	C File	2 KB
image.c	7/20/2020 8:12 PM	C File	8 KB
image.h	7/20/2020 8:12 PM	H File	4 KB
parameter.h	7/20/2020 8:12 PM	H File	3 KB
semihost_hardfault.c	8/11/2020 3:04 PM	C File	5 KB
ship.bmp	7/20/2020 8:09 PM	BMP File	4 KB
ship.h	7/20/2020 8:12 PM	H File	16 KB
timer.c	7/20/2020 8:12 PM	C File	2 KB
timer.h	7/20/2020 8:12 PM	H File	1 KB
weights.h	7/20/2020 8:12 PM	H File	256 KB

18. Then inside the unzipped LPC55S69 SDK folder, copy the **libarm_ARMv8MMLldfsp_math.a** library file found at **\SDK_2.8.0_LPCXpresso55S69\CMSIS\DSP\Lib\GCC** and paste it into the "**libs**" folder that already exists at <project_location>

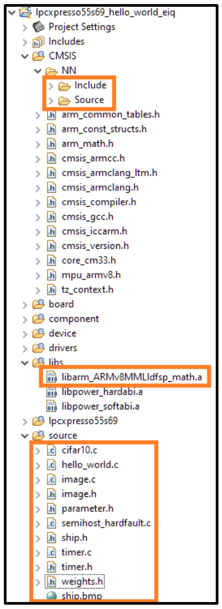


› lpcxpresso55s69_hello_world_eiq › libs

Name

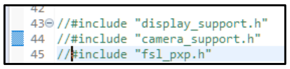
libarm_ARMv8MMLldfsp_math.a
libpower_hardabi.a
libpower_softabi.a

19. Back in the MCUXpresso IDE, click on the project name, and then hit F5 to refresh the file list. You should see the new files show up and look like the following:



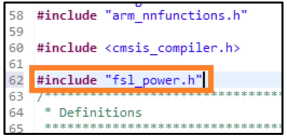
```
lpcxpresso55s69_hello_world_eiq
  Project Settings
  Includes
  CMSIS
    NN
      Include
      Source
    arm_common_tables.h
    arm_const_structs.h
    arm_math.h
    cmsis_armcc.h
    cmsis_armclang_ltm.h
    cmsis_armclang.h
    cmsis_compiler.h
    cmsis_gcc.h
    cmsis_iccarm.h
    cmsis_version.h
    core_cm33.h
    mpu_armv8.h
    tz_context.h
  board
  component
  device
  drivers
  libs
    libarm_ARMv8MMLldfsp_math.a
    libpower_hardabi.a
    libpower_softabi.a
  lpcxpresso55s69
  source
    cifar10.c
    hello_world.c
    image.c
    image.h
    parameter.h
    semihost_hardfault.c
    ship.h
    timer.c
    timer.h
    weights.h
    ship.bmp
```

20. Next, the code in **cifar10.c** needs to be updated to work with the LPC55S69. The biggest change will be that the RT1060 SDK demo includes camera and LCD support, which needs to be removed when running on other boards.
    a. Remove or comment out the 3 camera and display header files includes at the top of the file near line 43:



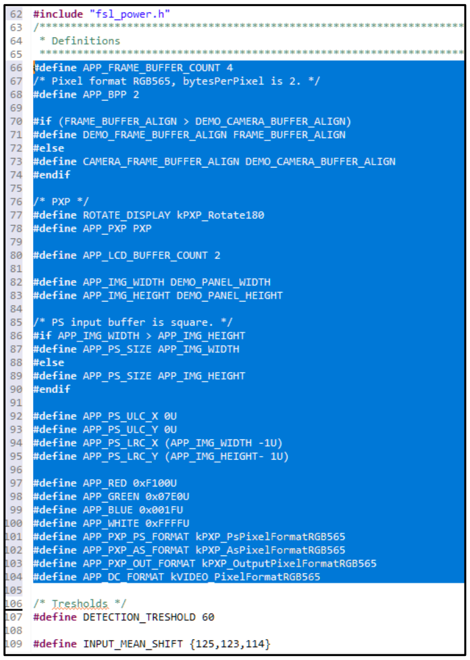
```
42
43 //#include "display_support.h"
44 //#include "camera_support.h"
45 //#include "fsl_pxp.h"
```

    b. Add **#define "fsl_power.h"** to the list of includes



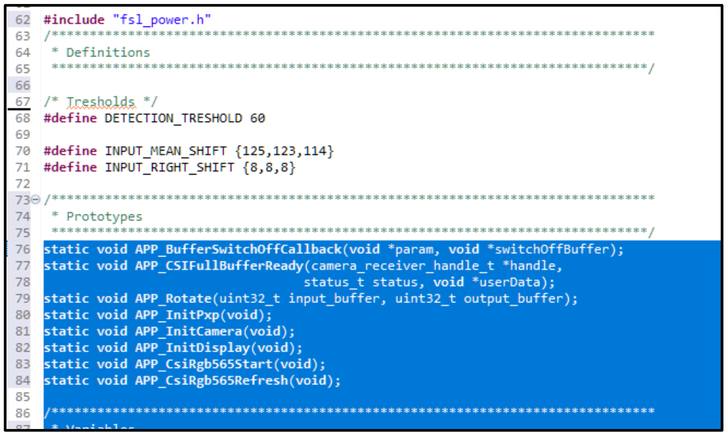
```
#include "arm_nnfunctions.h"

#include <cmsis_compiler.h>

#include "fsl_power.h"
/*************************************
 * Definitions
 *************************************
```

c. Then remove the camera and LCD defines from lines 66 to 104

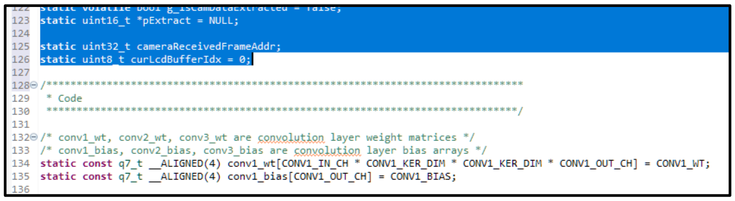


```
62  #include "fsl_power.h"
63  /*****************************************************************************
64   * Definitions
65   *****************************************************************************
66  #define APP_FRAME_BUFFER_COUNT 4
67  /* Pixel format RGB565, bytesPerPixel is 2. */
68  #define APP_BPP 2
69  
70  #if (FRAME_BUFFER_ALIGN > DEMO_CAMERA_BUFFER_ALIGN)
71  #define DEMO_FRAME_BUFFER_ALIGN FRAME_BUFFER_ALIGN
72  #else
73  #define CAMERA_FRAME_BUFFER_ALIGN DEMO_CAMERA_BUFFER_ALIGN
74  #endif
75  
76  /* PXP */
77  #define ROTATE_DISPLAY kPXP_Rotate180
78  #define APP_PXP PXP
79  
80  #define APP_LCD_BUFFER_COUNT 2
81  
82  #define APP_IMG_WIDTH DEMO_PANEL_WIDTH
83  #define APP_IMG_HEIGHT DEMO_PANEL_HEIGHT
84  
85  /* PS input buffer is square. */
86  #if APP_IMG_WIDTH > APP_IMG_HEIGHT
87  #define APP_PS_SIZE APP_IMG_WIDTH
88  #else
89  #define APP_PS_SIZE APP_IMG_HEIGHT
90  #endif
91  
92  #define APP_PS_ULC_X 0U
93  #define APP_PS_ULC_Y 0U
94  #define APP_PS_LRC_X (APP_IMG_WIDTH -1U)
95  #define APP_PS_LRC_Y (APP_IMG_HEIGHT- 1U)
96  
97  #define APP_RED 0xF100U
98  #define APP_GREEN 0x07E0U
99  #define APP_BLUE 0x001FU
100 #define APP_WHITE 0xFFFFU
101 #define APP_PXP_PS_FORMAT kPXP_PsPixelFormatRGB565
102 #define APP_PXP_AS_FORMAT kPXP_AsPixelFormatRGB565
103 #define APP_PXP_OUT_FORMAT kPXP_OutputPixelFormatRGB565
104 #define APP_DC_FORMAT kVIDEO_PixelFormatRGB565
105 
106 /* Tresholds */
107 #define DETECTION_TRESHOLD 60
108 
109 #define INPUT_MEAN_SHIFT {125,123,114}
```

d. Then remove the LCD and camera defines, function prototypes, and variables from line 76 to 126:

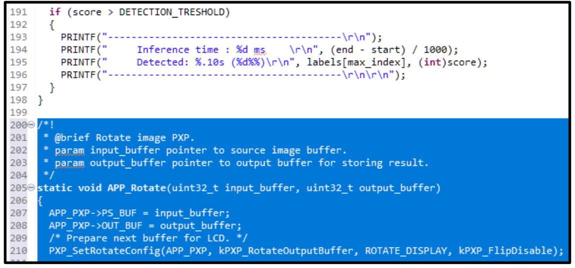


```
62  #include "fsl_power.h"
63  /*******************************************************************************
64   * Definitions
65   ******************************************************************************/
66  
67  /* Tresholds */
68  #define DETECTION_TRESHOLD 60
69  
70  #define INPUT_MEAN_SHIFT {125,123,114}
71  #define INPUT_RIGHT_SHIFT {8,8,8}
72  
73⊖ /*******************************************************************************
74   * Prototypes
75   ******************************************************************************/
76  static void APP_BufferSwitchOffCallback(void *param, void *switchOffBuffer);
77  static void APP_CSIFullBufferReady(camera_receiver_handle_t *handle,
78                                     status_t status, void *userData);
79  static void APP_Rotate(uint32_t input_buffer, uint32_t output_buffer);
80  static void APP_InitPxp(void);
81  static void APP_InitCamera(void);
82  static void APP_InitDisplay(void);
83  static void APP_CsiRgb565Start(void);
84  static void APP_CsiRgb565Refresh(void);
85  
86  /*******************************************************************************
87   * Variables
```

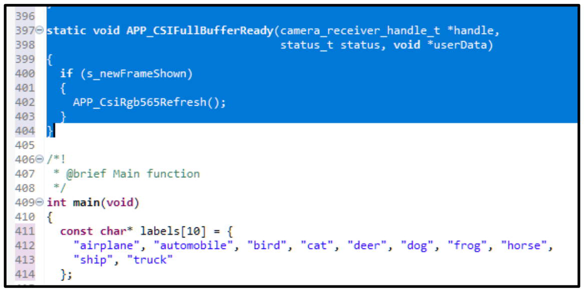


```
122 static volatile bool g_isCamDataExtracted = false;
123 static uint16_t *pExtract = NULL;
124 
125 static uint32_t cameraReceivedFrameAddr;
126 static uint8_t curLcdBufferIdx = 0;
127 
128⊖ /*******************************************************************************
129  * Code
130  ******************************************************************************/
131 
132⊖ /* conv1_wt, conv2_wt, conv3_wt are convolution layer weight matrices */
133 /* conv1_bias, conv2_bias, conv3_bias are convolution layer bias arrays */
134 static const q7_t __ALIGNED(4) conv1_wt[CONV1_IN_CH * CONV1_KER_DIM * CONV1_KER_DIM * CONV1_OUT_CH] = CONV1_WT;
135 static const q7_t __ALIGNED(4) conv1_bias[CONV1_OUT_CH] = CONV1_BIAS;
136 
```

e. Then remove the functions starting around line 200 to line 404 (after deleting the code from the previous steps). It starts at **APP_Rotate()** and goes all the way to **main()**:

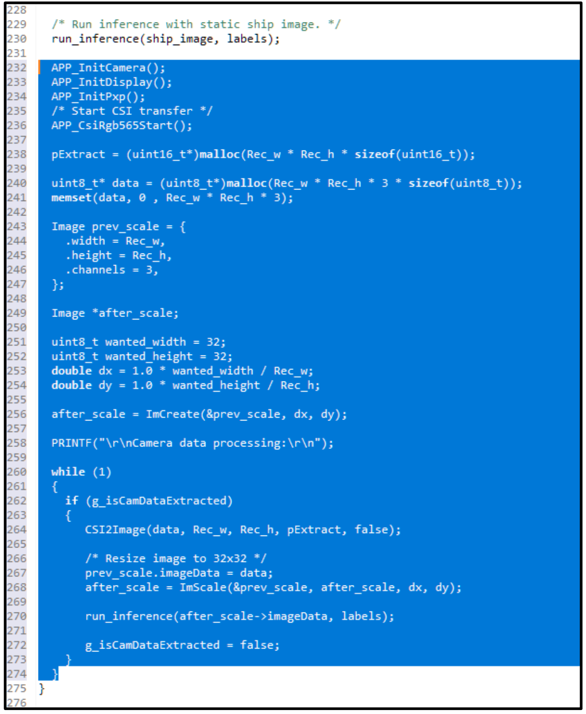


```
191   if (score > DETECTION_TRESHOLD)
192   {
193     PRINTF("-------------------------------------\r\n");
194     PRINTF("     Inference time : %d ms    \r\n", (end - start) / 1000);
195     PRINTF("     Detected: %.10s (%d%%)\r\n", labels[max_index], (int)score);
196     PRINTF("-------------------------------------\r\n\r\n");
197   }
198 }
199 
200⊖/*!
201  * @brief Rotate image PXP.
202  * param input_buffer pointer to source image buffer.
203  * param output_buffer pointer to output buffer for storing result.
204  */
205⊖static void APP_Rotate(uint32_t input_buffer, uint32_t output_buffer)
206 {
207   APP_PXP->PS_BUF = input_buffer;
208   APP_PXP->OUT_BUF = output_buffer;
209   /* Prepare next buffer for LCD. */
210   PXP_SetRotateConfig(APP_PXP, kPXP_RotateOutputBuffer, ROTATE_DISPLAY, kPXP_FlipDisable);
```



```
static void APP_CSIFullBufferReady(camera_receiver_handle_t *handle,
                                    status_t status, void *userData)
{
  if (s_newFrameShown)
  {
    APP_CsiRgb565Refresh();
  }
}

/*!
 * @brief Main function
 */
int main(void)
{
  const char* labels[10] = {
    "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse",
    "ship", "truck"
  };
```

f. Finally in main(), remove the code in main() that is after the function call to run_inference(), leaving the last closing bracket at the bottom. Remove lines 232 to 274:



```
228
229    /* Run inference with static ship image. */
230    run_inference(ship_image, labels);
231
232    APP_InitCamera();
233    APP_InitDisplay();
234    APP_InitPxp();
235    /* Start CSI transfer */
236    APP_CsiRgb565Start();
237
238    pExtract = (uint16_t*)malloc(Rec_w * Rec_h * sizeof(uint16_t));
239
240    uint8_t* data = (uint8_t*)malloc(Rec_w * Rec_h * 3 * sizeof(uint8_t));
241    memset(data, 0 , Rec_w * Rec_h * 3);
242
243    Image prev_scale = {
244      .width = Rec_w,
245      .height = Rec_h,
246      .channels = 3,
247    };
248
249    Image *after_scale;
250
251    uint8_t wanted_width = 32;
252    uint8_t wanted_height = 32;
253    double dx = 1.0 * wanted_width / Rec_w;
254    double dy = 1.0 * wanted_height / Rec_h;
255
256    after_scale = ImCreate(&prev_scale, dx, dy);
257
258    PRINTF("\r\nCamera data processing:\r\n");
259
260    while (1)
261    {
262      if (g_isCamDataExtracted)
263      {
264         CSI2Image(data, Rec_w, Rec_h, pExtract, false);
265
266         /* Resize image to 32x32 */
267         prev_scale.imageData = data;
268         after_scale = ImScale(&prev_scale, after_scale, dx, dy);
269
270         run_inference(after_scale->imageData, labels);
271
272         g_isCamDataExtracted = false;
273      }
274    }
275  }
276
```

21. Then the main function in cifar10.c will need to be updated for setting up the clock and board:

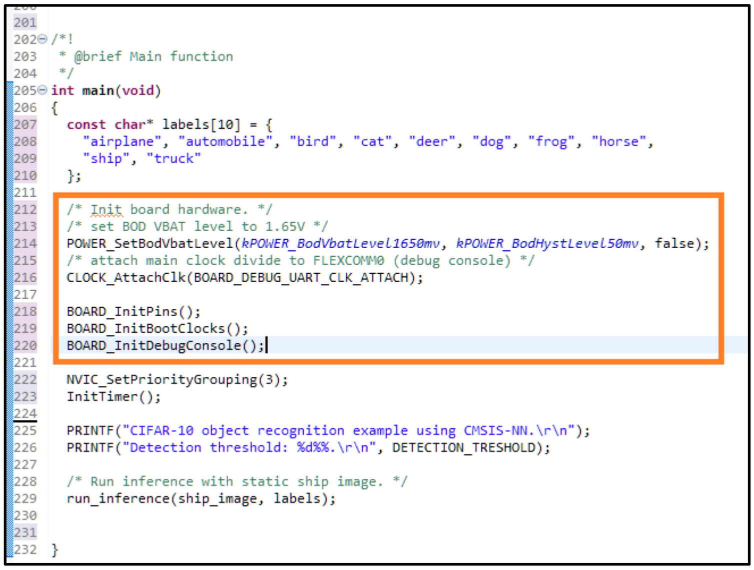
a. Open **hello_world.c** by double clicking on the file name

b. Copy the following lines from **hello_world.c** starting from about line 36 which sets up the clock and board hardware.

```
/* Init board hardware. */
/* set BOD VBAT level to 1.65V */
POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
/* attach main clock divide to FLEXCOMM0 (debug console) */
CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

BOARD_InitPins();
BOARD_InitBootClocks();
BOARD_InitDebugConsole();
```

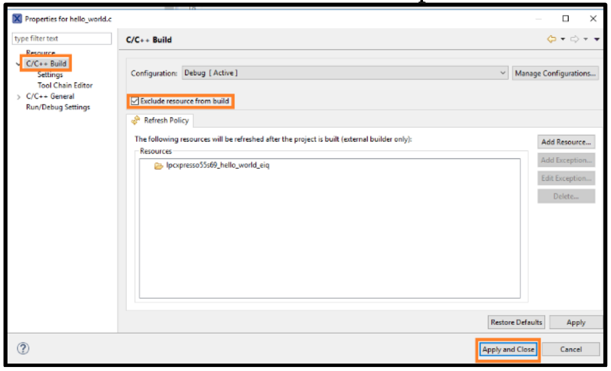
c. Overwrite the similar board setup code in **cifar10.c** near the beginning of **main(void)**.

d. **main() in cifar10.c** will look like the following when complete:



```
/*!
 * @brief Main function
 */
int main(void)
{
  const char* labels[10] = {
    "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse",
    "ship", "truck"
  };

  /* Init board hardware. */
  /* set BOD VBAT level to 1.65V */
  POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
  /* attach main clock divide to FLEXCOMM0 (debug console) */
  CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

  BOARD_InitPins();
  BOARD_InitBootClocks();
  BOARD_InitDebugConsole();

  NVIC_SetPriorityGrouping(3);
  InitTimer();

  PRINTF("CIFAR-10 object recognition example using CMSIS-NN.\r\n");
  PRINTF("Detection threshold: %d%%.\r\n", DETECTION_TRESHOLD);

  /* Run inference with static ship image. */
  run_inference(ship_image, labels);


}
```

22. Finally, prevent the **hello_world.c** file from compiling by right clicking on that file and selecting **Properties**. In the dialog box that comes up, select the **C/C++ Build** category, and check the **Exclude resource from build** option. Then click **Apply and Close**.



Configuration: Debug [Active]

☑ Exclude resource from build

The following resources will be refreshed after the project is built (external builder only):

Resources

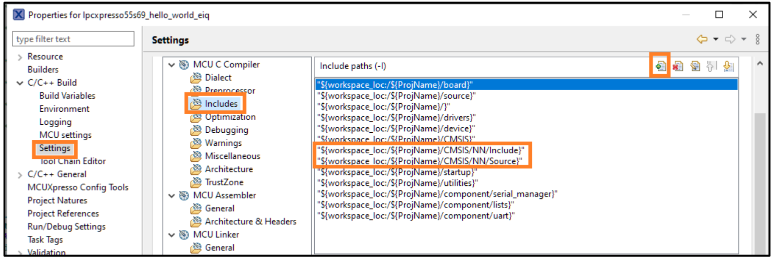
lpcxpresso55s69_hello_world_eiq

4.3 MCUXpresso IDE Project Settings

To get the project to properly compile, some changes to the project settings must be made in the IDE:

23. Open the Properties dialog box by right clicking on the project name and selecting Properties (like done in the previous section).
24. Add the include path for the CMSIS-NN files by going to the **C/C++ Build->Settings** category and then in **Tool Settings** tab, in **MCU C Compiler->Includes**, click on the **Add** button and add the following:
"${workspace_loc:/${ProjName}/CMSIS/NN/Source}"
"${workspace_loc:/${ProjName}/CMSIS/NN/Include}"

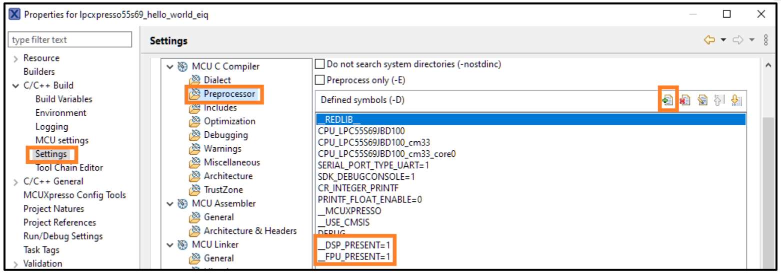
It will look like the following when complete:



Include paths (-I)

```
"${workspace_loc:/${ProjName}/board}"
"${workspace_loc:/${ProjName}/source}"
"${workspace_loc:/${ProjName}/}"
"${workspace_loc:/${ProjName}/drivers}"
"${workspace_loc:/${ProjName}/device}"
"${workspace_loc:/${ProjName}/CMSIS}"
"${workspace_loc:/${ProjName}/CMSIS/NN/Include}"
"${workspace_loc:/${ProjName}/CMSIS/NN/Source}"
"${workspace_loc:/${ProjName}/startup}"
"${workspace_loc:/${ProjName}/utilities}"
"${workspace_loc:/${ProjName}/component/serial_manager}"
"${workspace_loc:/${ProjName}/component/lists}"
"${workspace_loc:/${ProjName}/component/uart}"
```

25. Then add **__DSP_PRESENT=1** and **__FPU_PRESENT=1** to the precompile options by using the Add button. Note the **two** underscore lines before each option.



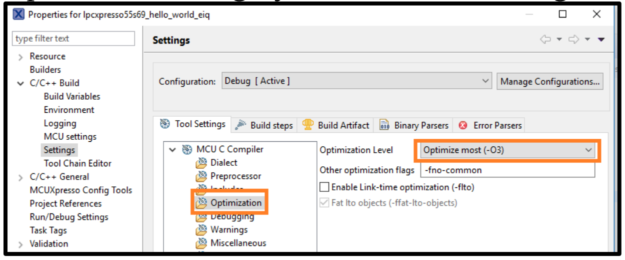
□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

```
__REDLIB__
CPU_LPC55S69JBD100
CPU_LPC55S69JBD100_cm33
CPU_LPC55S69JBD100_cm33_core0
SERIAL_PORT_TYPE_UART=1
SDK_DEBUGCONSOLE=1
CR_INTEGER_PRINTF
PRINTF_FLOAT_ENABLE=0
__MCUXPRESSO
__USE_CMSIS
DEBUG
__DSP_PRESENT=1
__FPU_PRESENT=1
```

26. eIQ inference time is highly sensitive to code optimization settings. This can be adjusted in the Optimization category. Set to O3 for the highest optimization.



Configuration: Debug [Active]

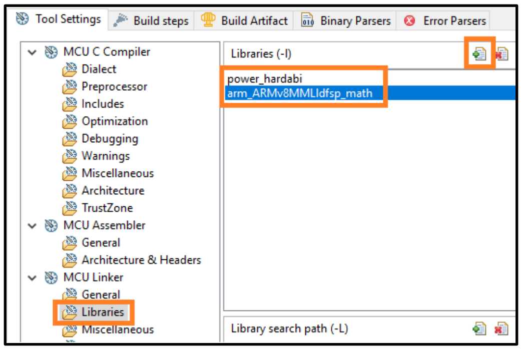
Optimization Level: Optimize most (-O3)

Other optimization flags: -fno-common

□ Enable Link-time optimization (-flto)

☑ Fat lto objects (-ffat-lto-objects)

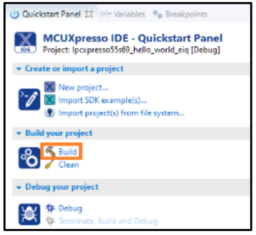
27. Then in the MCU Linker->Libraries category, add the math library by using the Add button to add:
arm_ARMv8MMLldfsp_math



Libraries (-l)

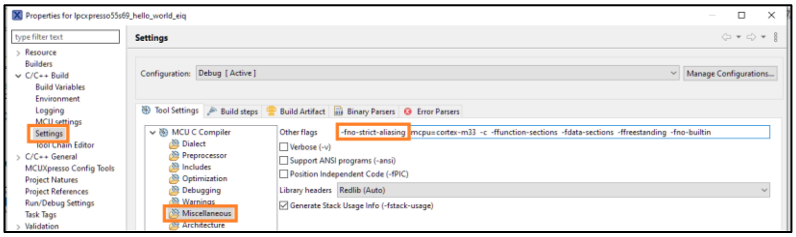
```
power_hardabi
arm_ARMv8MMLldfsp_math
```

28. Click on **Apply and Close** to save these settings and close the project options dialog box.
29. Build the project using the Quickstart Panel



Project: lpcxpresso55s69_hello_world_eiq [Debug]

30. You may get some warning messages. This is due to a CMSIS-NN bug. These can be disabled by adding **-fno-strict-aliasing** to the compiler flag settings under **Miscellaneous** in the project settings.



Configuration: Debug [Active]

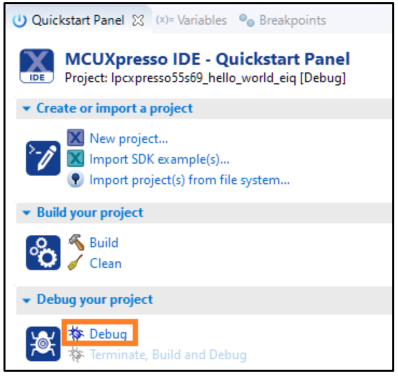
Other flags: -fno-strict-aliasing mcpu=cortex-m33 -c -ffunction-sections -fdata-sections -ffreestanding -fno-builtin

□ Verbose (-v)
□ Support ANSI programs (-ansi)
□ Position Independent Code (-fPIC)

Library headers: Redlib (Auto)

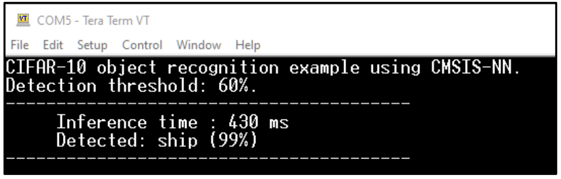
☑ Generate Stack Usage Info (-fstack-usage)

31. Flash and debug the project from the Quickstart Panel



Project: lpcxpresso55s69_hello_world_eiq [Debug]

32. Open a terminal program, and you will see it identify the ship image loaded in ship.h:



```
CIFAR-10 object recognition example using CMSIS-NN.
Detection threshold: 60%.
----------------------------------------
     Inference time : 430 ms
     Detected: ship (99%)
----------------------------------------
```

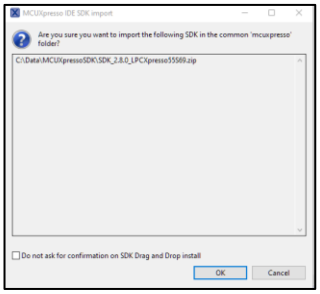
5 MCUXpresso IDE Porting for Keyword Spotting Demo

5.1 Create a C++ Project

The first step is to create a new project using the "New Project" wizard in MCUXpresso IDE. This new project will be used as a template to copy the eIQ source into.

1. Unzip both the LPC55S69 and the RT1060 SDK zip files into a directory path that does not contain spaces. The files inside will be used in later steps.

2. Open up MCUXpresso IDE and select a new workspace
3. Install the LPC55S69 SDK into the "Installed SDKs" tab by dragging and dropping the zipped **SDK_2.8.0_LPCXpresso55S69.zip** file into the Installed SDKs window. This dialog box will come up, and click OK to continue the import

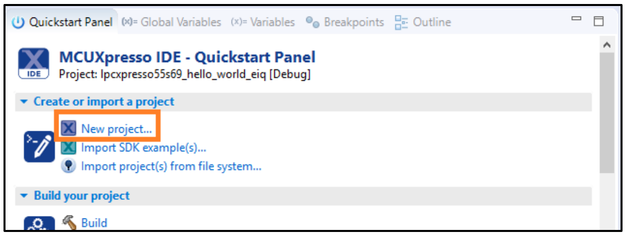


Are you sure you want to import the following SDK in the common 'mcuxpresso' folder?

C:\Data\MCUXpressoSDK\SDK_2.8.0_LPCXpresso55S69.zip

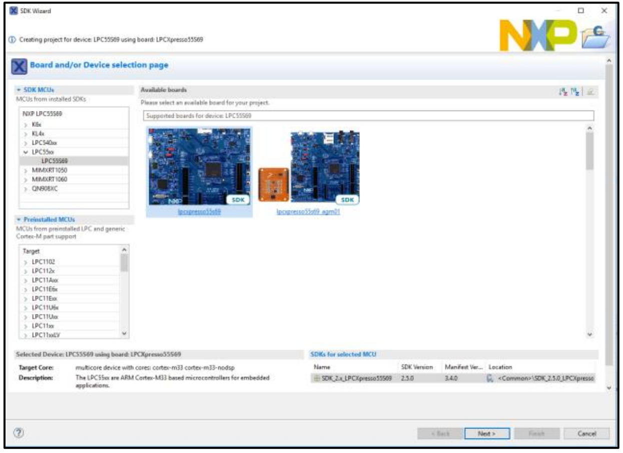
□ Do not ask for confirmation on SDK Drag and Drop install

4. Next, create a new LPC55S69 project. In the Quickstart Panel, select **New Project…**



Project: lpcxpresso55s69_hello_world_eiq [Debug]

5. Select the lpcxpresso55s69 board and click on Next



Creating project for device: LPC55S69 using board: LPCXpresso55S69

Selected Device: LPC55S69 using board: LPCXpresso55S69

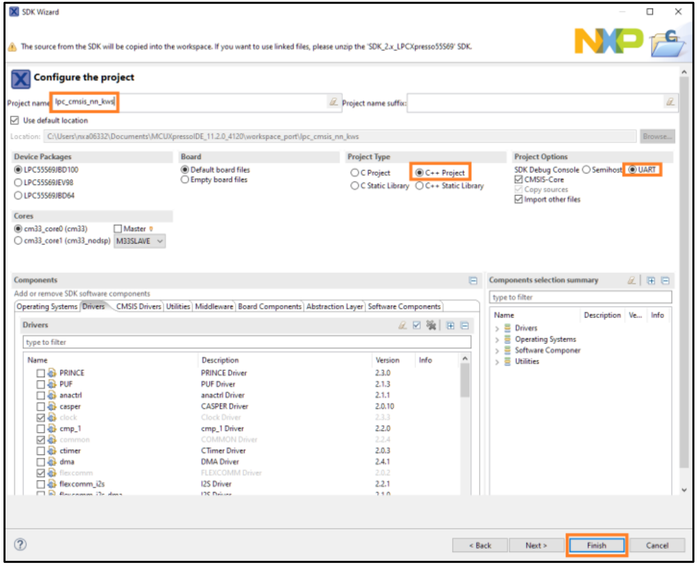
Target Core: multicore device with cores: cortex-m33 cortex-m33-nodsp

Description: The LPC55xx are ARM Cortex-M33 based microcontrollers for embedded applications.

SDKs for selected MCU

Name	SDK Version	Manifest Ver...	Location
SDK_2.x_LPCXpresso55S69	2.5.0	3.4.0	<Common>\SDK_2.5.0_LPCXpresso

6. Give it a unique project name like **lpc_cmsis_nn_kws**, then make sure to select **C++ Project** in the Project Type. Also, in the Project Options, select **UART**. Click on **Finish**.



The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_LPCXpresso55S69' SDK.

Project name: lpc_cmsis_nn_kws

Project name suffix:

☑ Use default location

Location: C:\Users\nxa06332\Documents\MCUXpressoIDE_11.2.0_4120\workspace_port\lpc_cmsis_nn_kws

Device Packages: ● LPC55S69JBD100 ○ LPC55S69JEV98 ○ LPC55S69JBD64

Board: ● Default board files ○ Empty board files

Project Type: ○ C Project ● C++ Project ○ C Static Library ○ C++ Static Library

Project Options: SDK Debug Console ○ Semihost ● UART; ☑ CMSIS-Core; ☑ Copy sources; ☑ Import other files

Cores: ● cm33_core0 (cm33) □ Master; ○ cm33_core1 (cm33_nodsp) M33SLAVE

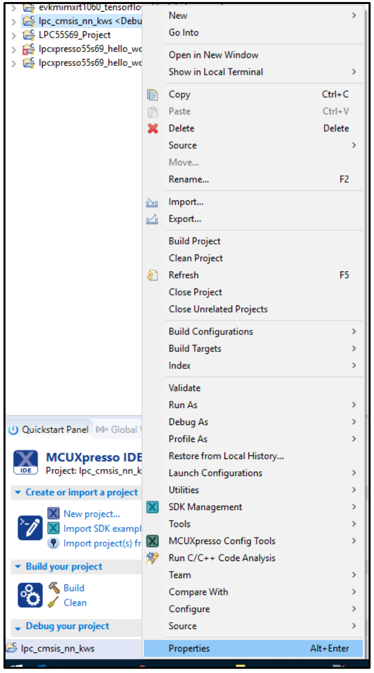
Name	Description	Version	Info
□ PRINCE	PRINCE Driver	2.3.0	
□ PUF	PUF Driver	2.1.3	
□ anactrl	anactrl Driver	2.1.1	
□ casper	CASPER Driver	2.0.10	
☑ clock	Clock Driver	2.3.3	
□ cmp_1	cmp_1 Driver	2.2.0	
☑ common	COMMON Driver	2.2.4	
□ ctimer	CTimer Driver	2.0.3	
□ dma	DMA Driver	2.4.1	
☑ flexcomm	FLEXCOMM Driver	2.0.2	
□ flexcomm_i2s	I2S Driver	2.2.1	
□ flexcomm_i2s_dma	I2S Driver	2.1.0	

Name	Description	Ve...	Info
Drivers			
Operating Systems			
Software Componer			
Utilities			

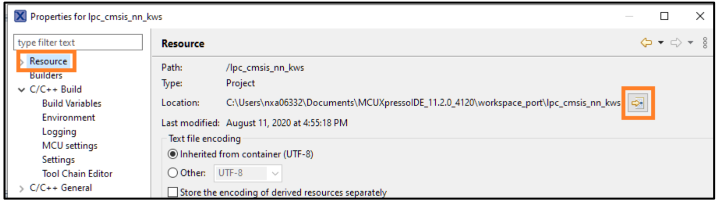
7. This will create a new project in your workspace that you can see in the project view

5.2 Source Files

8. Once the project is created, open the project location by right click on the project name and selecting Properties.

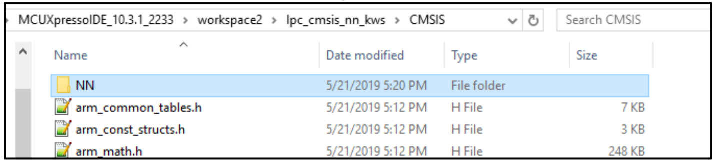


9. Then go to the **Resource** category and by the **Location**, click on the arrow to open that location in Windows Explorer



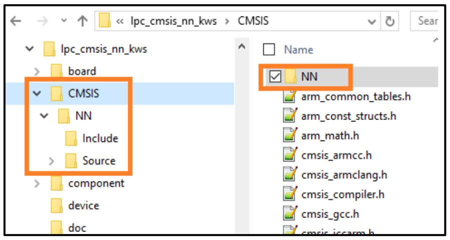
Path: /lpc_cmsis_nn_kws
Type: Project
Location: C:\Users\nxa06332\Documents\MCUXpressoIDE_11.2.0_4120\workspace_port\lpc_cmsis_nn_kws
Last modified: August 11, 2020 at 4:55:18 PM
Text file encoding
● Inherited from container (UTF-8)
○ Other: UTF-8
□ Store the encoding of derived resources separately

10. In Windows Explorer, navigate to the **lpc_cmsis_nn_kws** folder (or what is was named above), and then the **CMSIS** folder. Then create a new folder inside the CMSIS folder named "**NN**". It will look like the following when complete:

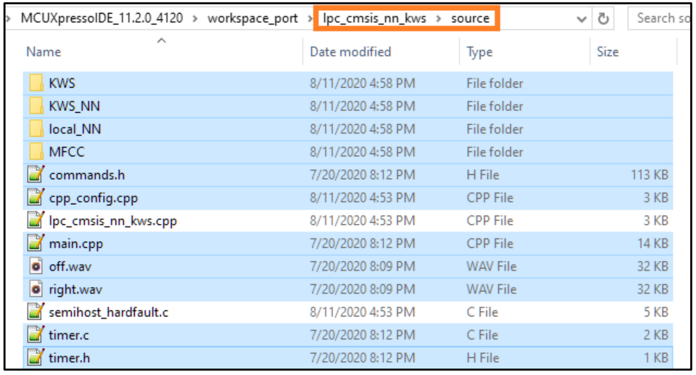

MCUXpressoIDE_10.3.1_2233 › workspace2 › lpc_cmsis_nn_kws › CMSIS
Search CMSIS

Name	Date modified	Type	Size
NN	5/21/2019 5:20 PM	File folder	
arm_common_tables.h	5/21/2019 5:12 PM	H File	7 KB
arm_const_structs.h	5/21/2019 5:12 PM	H File	3 KB
arm_math.h	5/21/2019 5:12 PM	H File	248 KB

11. If not done already, unzip the RT1060 SDK. The eIQ source files will be used in the next steps.
12. Inside the RT1060 SDK go to **\middleware\eiq\cmsis-nn\** and copy the Source and Include folders into the **NN** folder created in the previous step.
13. When finished it should look like the following:

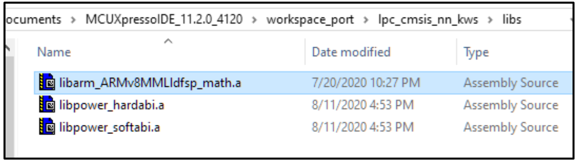


14. Then take the folders and files in the RT1050 or RT1060 **\middleware\eiq\cmsis-nn\Examples\kws** folder and copy them into the “Source” folder found at <project_location>.
15. Then also take the files in **\middleware\eiq\cmsis-nn\Examples\common\source** and copy them into the same “Source” folder as well at <project_location>. It will look like the following when complete:


MCUXpressoIDE_11.2.0_4120 › workspace_port › lpc_cmsis_nn_kws › source
Search so

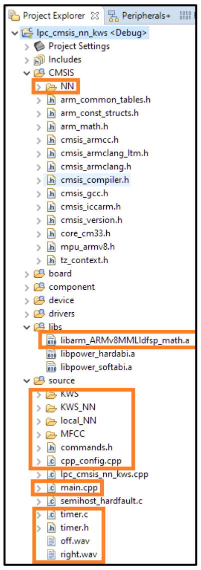
Name	Date modified	Type	Size
KWS	8/11/2020 4:58 PM	File folder	
KWS_NN	8/11/2020 4:58 PM	File folder	
local_NN	8/11/2020 4:58 PM	File folder	
MFCC	8/11/2020 4:58 PM	File folder	
commands.h	7/20/2020 8:12 PM	H File	113 KB
cpp_config.cpp	8/11/2020 4:53 PM	CPP File	3 KB
lpc_cmsis_nn_kws.cpp	8/11/2020 4:53 PM	CPP File	3 KB
main.cpp	7/20/2020 8:12 PM	CPP File	14 KB
off.wav	7/20/2020 8:09 PM	WAV File	32 KB
right.wav	7/20/2020 8:09 PM	WAV File	32 KB
semihost_hardfault.c	8/11/2020 4:53 PM	C File	5 KB
timer.c	7/20/2020 8:12 PM	C File	2 KB
timer.h	7/20/2020 8:12 PM	H File	1 KB

16. Then inside the LPC55S69 SDK folder, copy the **libarm_ARMv8MMLldfsp_math.a** library file found at **\SDK_2.8.0_LPCXpresso55S69\CMSIS\DSP\Lib\GCC** and paste it into the “libs” folder that already exists at <project_location>


ocuments › MCUXpressoIDE_11.2.0_4120 › workspace_port › lpc_cmsis_nn_kws › libs

Name	Date modified	Type
libarm_ARMv8MMLldfsp_math.a	7/20/2020 10:27 PM	Assembly Source
libpower_hardabi.a	8/11/2020 4:53 PM	Assembly Source
libpower_softabi.a	8/11/2020 4:53 PM	Assembly Source

17. Back in the MCUXpresso IDE, click on the project name, and then hit F5 to refresh the file list. You should see the new files show up and look like the following:

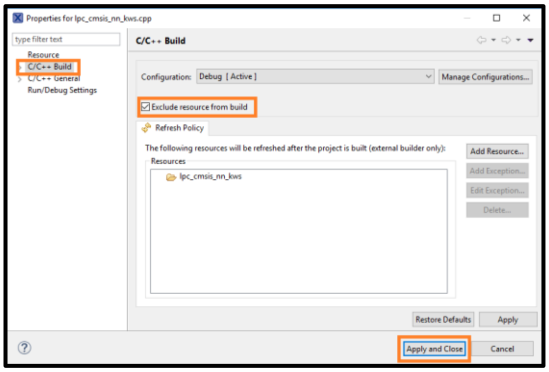


```
lpc_cmsis_nn_kws <Debug>
  Project Settings
  Includes
  CMSIS
    NN
    arm_common_tables.h
    arm_const_structs.h
    arm_math.h
    cmsis_armcc.h
    cmsis_armclang_ltm.h
    cmsis_armclang.h
    cmsis_compiler.h
    cmsis_gcc.h
    cmsis_iccarm.h
    cmsis_version.h
    core_cm33.h
    mpu_armv8.h
    tz_context.h
  board
  component
  device
  drivers
  libs
    libarm_ARMv8MMLldfsp_math.a
    libpower_hardabi.a
    libpower_softabi.a
  source
    KWS
    KWS_NN
    local_NN
    MFCC
    commands.h
    cpp_config.cpp
    lpc_cmsis_nn_kws.cpp
    main.cpp
    semihost_hardfault.c
    timer.c
    timer.h
    off.wav
    right.wav
```

18. Next, the code in **main.cpp** needs to be updated to work with the LPC55S69.
    a. Open **main.cpp** by double clicking on the file name
    b. All the SAI and microphone code will need to be removed, as the inference will just be done on pre-recorded files. To simplify this process, delete everything in the file and then replace it all with the code below.

```
#include "board.h"
#include "fsl_debug_console.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "timer.h"
#include "kws_ds_cnn.h"
#include "commands.h"

#define DETECTION_TRESHOLD 55

void run_inference(KWS *kws, int16_t* buf, char output_class[12][8])
{
 int detection_threshold = DETECTION_TRESHOLD; //in percent
 kws->audio_buffer = buf;
 int start = 0;
 int end = 0;
 start = GetTimeInUS();
 kws->extract_features(); //extract mfcc features
 kws->classify(); //classify using dnn
 kws->average_predictions();
 end = GetTimeInUS();
 int max_ind = kws->get_top_class(kws->output);
 if (kws->averaged_output[max_ind] > detection_threshold * 128 / 100)
 {
  PRINTF("----------------------------------------\r\n");
  PRINTF("    Inference time: %d ms\r\n", (end - start) / 1000);
  PRINTF("    Detected: %.10s (%d%%)\r\n", output_class[max_ind], ((int)kws->averaged_output[max_ind] * 100 / 128));
  PRINTF("----------------------------------------\r\n\r\n");
 }
}

int main(void)
{

 /* (recording_win x frame_shift) is the actual recording window size. */
 int recording_win = 49;
 /* Averaging window for smoothing out the output predictions. */
 int averaging_window_len = 1;

 /* Create new instance for static audio files with averaging window len = 1. */
 KWS_DS_CNN *kws = new KWS_DS_CNN(recording_win, averaging_window_len);

 char output_class[12][8] = {"Silence", "Unknown", "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go"};

 BOARD_InitBootPins();
 BOARD_InitBootClocks();
 BOARD_InitDebugConsole();
 InitTimer();

 PRINTF("Keyword spotting example using CMSIS-NN.\r\n");
 PRINTF("Detection threshold: %d%%\r\n", DETECTION_TRESHOLD);
 PRINTF("\r\nStatic data processing:\r\n");

 run_inference(kws, (int16_t *)OFF, output_class);
 run_inference(kws, (int16_t *)RIGHT, output_class);
 while(1)
 {}
}
```

19. Finally, prevent the **lpc_cmsis_nn_kws.cpp** file from compiling by right clicking on that file and selecting **Properties**. In the dialog box that comes up, select the **C/C++ Build** category, and check the **Exclude resource from build** option. Then click **Apply and Close**.



Configuration: Debug [Active] Manage Configurations...

☑ Exclude resource from build

The following resources will be refreshed after the project is built (external builder only):

Resources

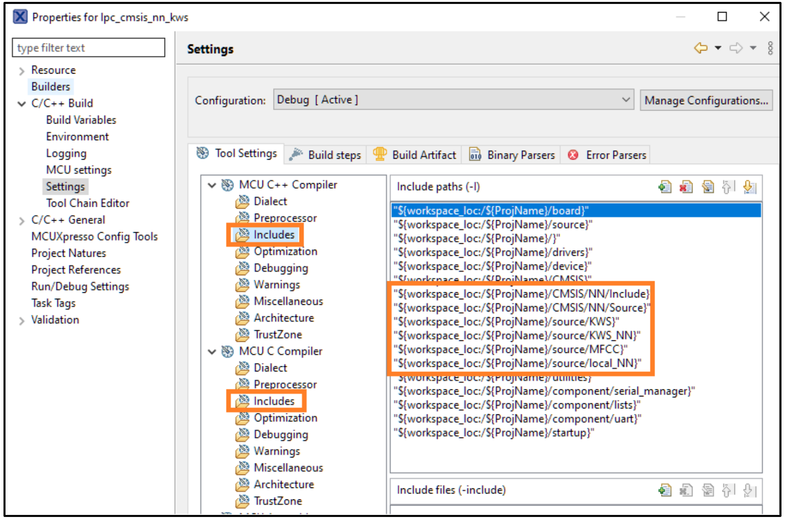
lpc_cmsis_nn_kws

5.3 Compiler IDE Settings

To get the project to properly compile, some changes to the project settings must be made in the IDE:

20. Open the Properties dialog box by right clicking on the project name and selecting Properties (like done in the previous section).
21. The following settings will need to be updated for both the "MCU C++ Compiler" and the "MCU C Compiler"
22. Add the include path for the CMSIS-NN files by going to the **C/C++ Build->Settings** category and then in **Tool Settings** tab, in **MCU C++ Compiler->Includes**, click on the Add button and add the following:
"${workspace_loc:/${ProjName}/CMSIS/NN/Source}"
"${workspace_loc:/${ProjName}/CMSIS/NN/Include}"
"${workspace_loc:/${ProjName}/source/KWS}"
"${workspace_loc:/${ProjName}/source/KWS_NN}"
"${workspace_loc:/${ProjName}/source/MFCC}"
"${workspace_loc:/${ProjName}/source/local_NN}"

Do this for both the C++ and C compiler categories.
It will look like the following when complete:

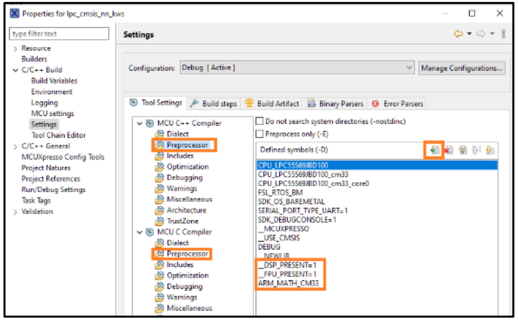


Configuration: Debug [Active] Manage Configurations...

Include paths (-I)

```
"${workspace_loc:/${ProjName}/board}"
"${workspace_loc:/${ProjName}/source}"
"${workspace_loc:/${ProjName}/}"
"${workspace_loc:/${ProjName}/drivers}"
"${workspace_loc:/${ProjName}/device}"
"${workspace_loc:/${ProjName}/CMSIS}"
"${workspace_loc:/${ProjName}/CMSIS/NN/Include}
"${workspace_loc:/${ProjName}/CMSIS/NN/Source}"
"${workspace_loc:/${ProjName}/source/KWS}"
"${workspace_loc:/${ProjName}/source/KWS_NN}"
"${workspace_loc:/${ProjName}/source/MFCC}"
"${workspace_loc:/${ProjName}/source/local_NN}"
"${workspace_loc:/${ProjName}/utilities}"
"${workspace_loc:/${ProjName}/component/serial_manager}"
"${workspace_loc:/${ProjName}/component/lists}"
"${workspace_loc:/${ProjName}/component/uart}"
"${workspace_loc:/${ProjName}/startup}"
```

23. Then add **__DSP_PRESENT=1** and **__FPU_PRESENT=1** and **ARM_MATH_CM33** to the precompile options by using the Add button. Note the two underscore lines before **__DSP_PRESENT** and **__FPU_PRESENT**. And remember that this is needed for both the C++ and C compiler options.



Configuration: Debug [Active]

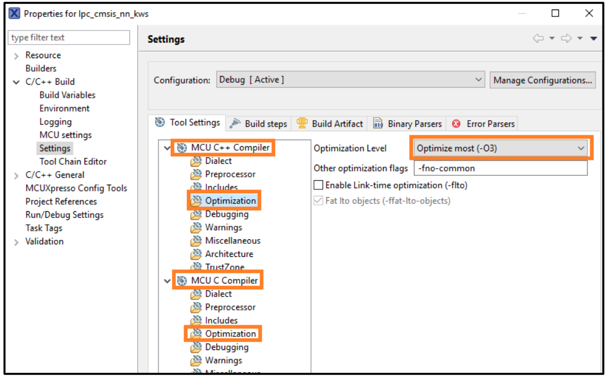
□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

```
CPU_LPC55S69JBD100
CPU_LPC55S69JBD100_cm33
CPU_LPC55S69JBD100_cm33_core0
FSL_RTOS_BM
SDK_OS_BAREMETAL
SERIAL_PORT_TYPE_UART=1
SDK_DEBUGCONSOLE=1
__MCUXPRESSO
__USE_CMSIS
DEBUG
__NEWLIB__
__DSP_PRESENT=1
__FPU_PRESENT=1
ARM_MATH_CM33
```

24. eIQ inference time is highly sensitive to code optimization settings. This can be adjusted in the **Optimization** category. Set to O3 for the highest optimization. Again, this needs to be done for both the C++ and C compiler options.



Configuration: Debug [Active]

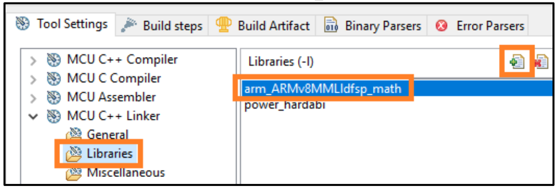
Optimization Level: Optimize most (-O3)

Other optimization flags: -fno-common

□ Enable Link-time optimization (-flto)

☑ Fat lto objects (-ffat-lto-objects)

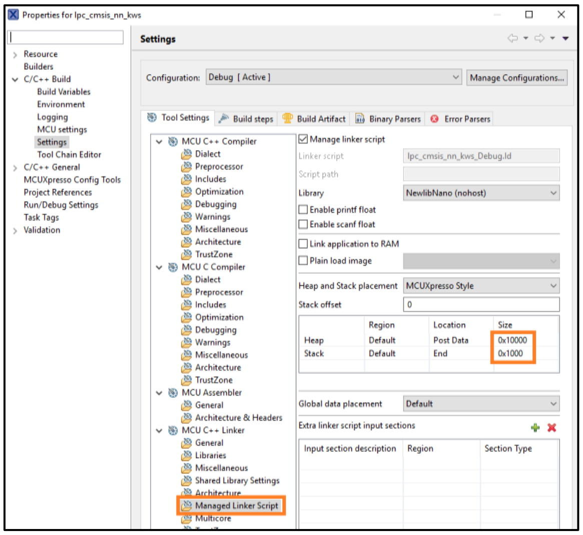
25. Then in the MCU Linker->Libraries category, add the math library by using the Add button to add:
arm_ARMv8MMLldfsp_math



Libraries (-l)

```
arm_ARMv8MMLldfsp_math
power_hardabi
```

26. In the **Managed Linker Script** category, adjust the heap and stack sizes to be 0x10000 and 0x1000 respectively.



Configuration: Debug [Active]

☑ Manage linker script

Linker script: lpc_cmsis_nn_kws_Debug.ld

Script path:

Library: NewlibNano (nohost)

☐ Enable printf float

☐ Enable scanf float

☐ Link application to RAM

☐ Plain load image

Heap and Stack placement: MCUXpresso Style

Stack offset: 0

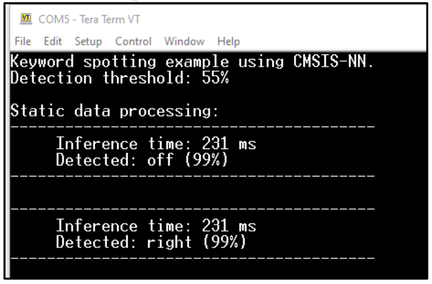
	Region	Location	Size
Heap	Default	Post Data	0x10000
Stack	Default	End	0x1000

Global data placement: Default

Extra linker script input sections

Input section description	Region	Section Type

27. Click on **Apply and Close** to save these settings and close the project options dialog box.
28. Build and debug as usual, and on the terminal, you will see it correctly identify the pre-recorded voice samples found in commands.h:

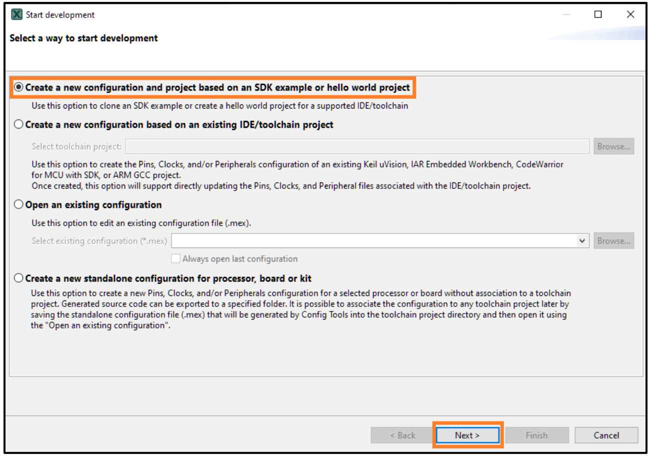


```
Keyword spotting example using CMSIS-NN.
Detection threshold: 55%

Static data processing:
----------------------------------------
     Inference time: 231 ms
     Detected: off (99%)
----------------------------------------

----------------------------------------
     Inference time: 231 ms
     Detected: right (99%)
----------------------------------------
```

6 IAR Porting

6.1 Cloning a Project

1. Extract the LPC55S69 SDK by unzipping the .zip file if not done already.
2. Open up MCUXpresso Config Tools
3. The following dialog box will pop up. If it does not, you can also go to File->New to open the dialog box. Click on **Create a new configuration based on an SDK example or hello world project** and then click on Next.



Start development

Select a way to start development

● Create a new configuration and project based on an SDK example or hello world project

Use this option to clone an SDK example or create a hello world project for a supported IDE/toolchain

○ Create a new configuration based on an existing IDE/toolchain project

Select toolchain project:

Use this option to create the Pins, Clocks, and/or Peripherals configuration of an existing Keil uVision, IAR Embedded Workbench, CodeWarrior for MCU with SDK, or ARM GCC project.
Once created, this option will support directly updating the Pins, Clocks, and Peripheral files associated with the IDE/toolchain project.

○ Open an existing configuration

Use this option to edit an existing configuration file (.mex).

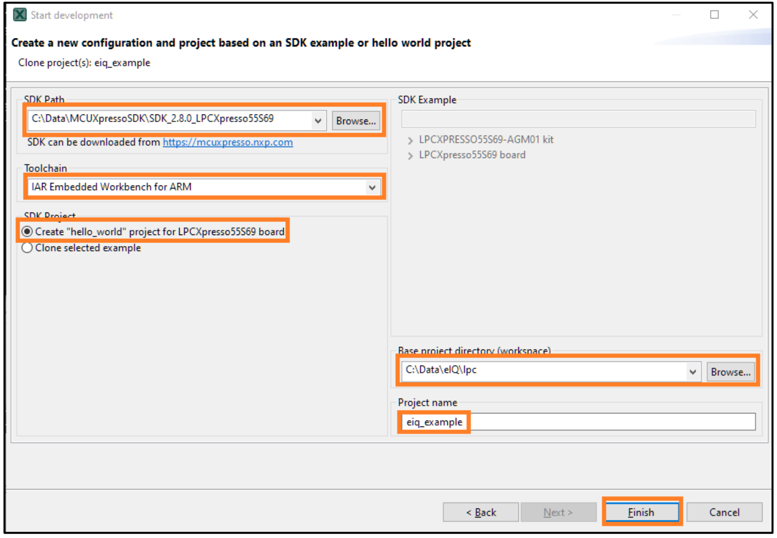
Select existing configuration (*.mex)

□ Always open last configuration

○ Create a new standalone configuration for processor, board or kit

Use this option to create a new Pins, Clocks, and/or Peripherals configuration for a selected processor or board without association to a toolchain project. Generated source code can be exported to a specified folder. It is possible to associate the configuration to any toolchain project later by saving the standalone configuration file (.mex) that will be generated by Config Tools into the toolchain project directory and then open it using the "Open an existing configuration".

4. On the next screen, there are several items to configure before clicking on Finish:
   c. Select the path to the LPC55S69 SDK unzipped earlier
   d. Select the toolchain (IAR in this case)
   e. Select to create a **hello_world** project
   f. Select a directory to put this new project into
   g. Select a name to give the project (**eiq_example** for this particular example)



Start development

Create a new configuration and project based on an SDK example or hello world project

Clone project(s): eiq_example

SDK Path: C:\Data\MCUXpressoSDK\SDK_2.8.0_LPCXpresso55S69

SDK can be downloaded from https://mcuxpresso.nxp.com

Toolchain: IAR Embedded Workbench for ARM

SDK Project

● Create "hello_world" project for LPCXpresso55S69 board

○ Clone selected example

SDK Example

› LPCXPRESSO55S69-AGM01 kit
› LPCXpresso55S69 board

Base project directory (workspace): C:\Data\eIQ\lpc

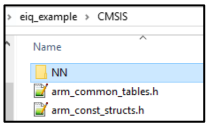
Project name: eiq_example

5. A new project will be created in the directory you specified.

6.2 Copying Source Files

6. With the project opened, the next steps are to copy in the CMSIS-NN files and example code into the project directory, and then import those files into IAR.

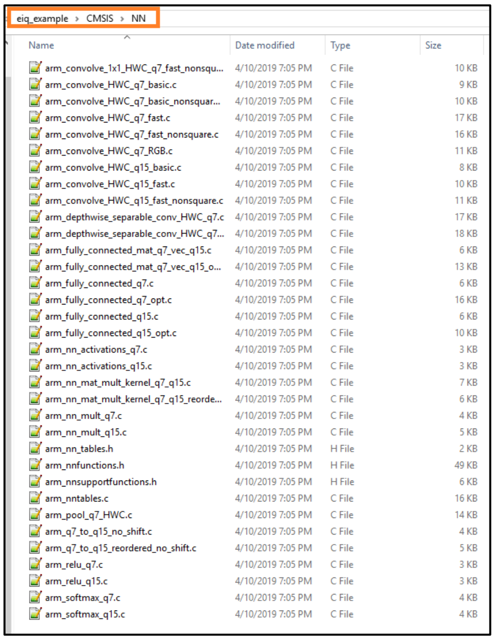
7. Go to the project directory created in the last section (ie **C:\Data\eIQ\lpc\eiq_example**), and inside the **CMSIS** folder, add a new folder named **NN**.



eiq_example › CMSIS

Name
NN
arm_common_tables.h
arm_const_structs.h

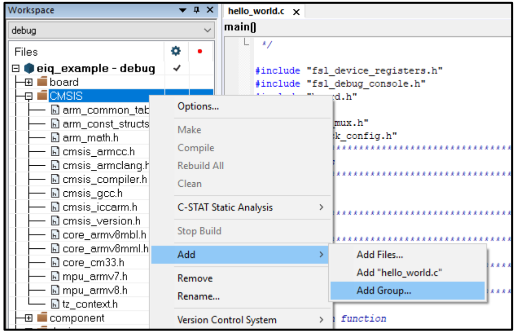
8. Inside the unzipped RT1060 SDK go to **\middleware\eiq\cmsis-nn\Source**. Inside each sub-directory copy and paste the .c and .h files into the NN folder created in the previous step.
9. Do the same for the files in the **\middleware\eiq\cmsis-nn\Include** folder so that they are also included in the NN folder.
10. When finished it should look like the following:



eiq_example › CMSIS › NN

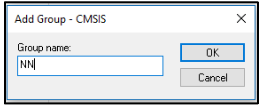
Name	Date modified	Type	Size
arm_convolve_1x1_HWC_q7_fast_nonsqu...	4/10/2019 7:05 PM	C File	10 KB
arm_convolve_HWC_q7_basic.c	4/10/2019 7:05 PM	C File	9 KB
arm_convolve_HWC_q7_basic_nonsquar...	4/10/2019 7:05 PM	C File	10 KB
arm_convolve_HWC_q7_fast.c	4/10/2019 7:05 PM	C File	17 KB
arm_convolve_HWC_q7_fast_nonsquare.c	4/10/2019 7:05 PM	C File	16 KB
arm_convolve_HWC_q7_RGB.c	4/10/2019 7:05 PM	C File	11 KB
arm_convolve_HWC_q15_basic.c	4/10/2019 7:05 PM	C File	8 KB
arm_convolve_HWC_q15_fast.c	4/10/2019 7:05 PM	C File	10 KB
arm_convolve_HWC_q15_fast_nonsquare.c	4/10/2019 7:05 PM	C File	11 KB
arm_depthwise_separable_conv_HWC_q7.c	4/10/2019 7:05 PM	C File	17 KB
arm_depthwise_separable_conv_HWC_q7...	4/10/2019 7:05 PM	C File	18 KB
arm_fully_connected_mat_q7_vec_q15.c	4/10/2019 7:05 PM	C File	6 KB
arm_fully_connected_mat_q7_vec_q15_o...	4/10/2019 7:05 PM	C File	13 KB
arm_fully_connected_q7.c	4/10/2019 7:05 PM	C File	6 KB
arm_fully_connected_q7_opt.c	4/10/2019 7:05 PM	C File	16 KB
arm_fully_connected_q15.c	4/10/2019 7:05 PM	C File	6 KB
arm_fully_connected_q15_opt.c	4/10/2019 7:05 PM	C File	10 KB
arm_nn_activations_q7.c	4/10/2019 7:05 PM	C File	3 KB
arm_nn_activations_q15.c	4/10/2019 7:05 PM	C File	3 KB
arm_nn_mat_mult_kernel_q7_q15.c	4/10/2019 7:05 PM	C File	7 KB
arm_nn_mat_mult_kernel_q7_q15_reorde...	4/10/2019 7:05 PM	C File	6 KB
arm_nn_mult_q7.c	4/10/2019 7:05 PM	C File	4 KB
arm_nn_mult_q15.c	4/10/2019 7:05 PM	C File	5 KB
arm_nn_tables.h	4/10/2019 7:05 PM	H File	2 KB
arm_nnfunctions.h	4/10/2019 7:05 PM	H File	49 KB
arm_nnsupportfunctions.h	4/10/2019 7:05 PM	H File	6 KB
arm_nntables.c	4/10/2019 7:05 PM	C File	16 KB
arm_pool_q7_HWC.c	4/10/2019 7:05 PM	C File	14 KB
arm_q7_to_q15_no_shift.c	4/10/2019 7:05 PM	C File	4 KB
arm_q7_to_q15_reordered_no_shift.c	4/10/2019 7:05 PM	C File	5 KB
arm_relu_q7.c	4/10/2019 7:05 PM	C File	3 KB
arm_relu_q15.c	4/10/2019 7:05 PM	C File	3 KB
arm_softmax_q7.c	4/10/2019 7:05 PM	C File	4 KB
arm_softmax_q15.c	4/10/2019 7:05 PM	C File	4 KB

11. Open IAR and open the new project created in the last section by going to File->Open Workspace and navigating to the directory to open the .eww file
12. Add a new group in the project view by right clicking on the CMSIS-NN folder and going to Add->Add Group..



```
 */

#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
```

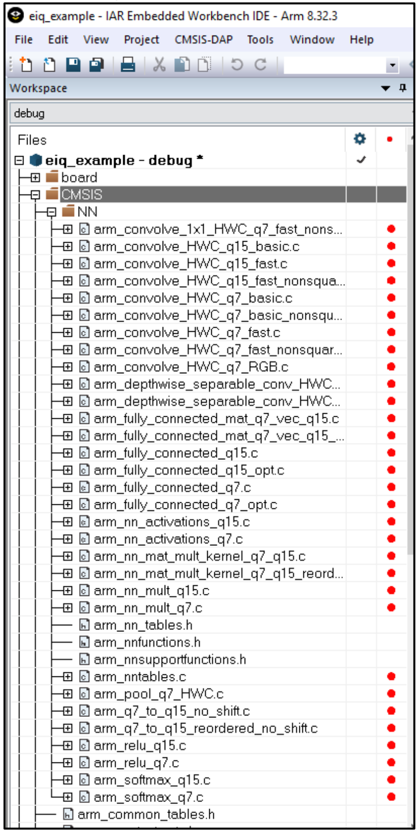
13. In the dialog box that appears, make the group name **NN** and click on OK



Add Group - CMSIS

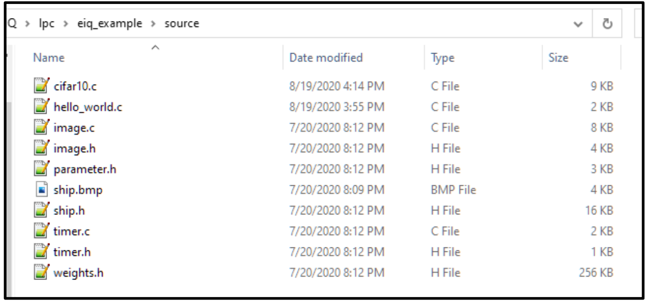
Group name: NN

14. Now drag-and-drop all the files in the <project_location>\CMSIS\NN folder into the NN group that was just created in IAR. It will look like the following when complete:



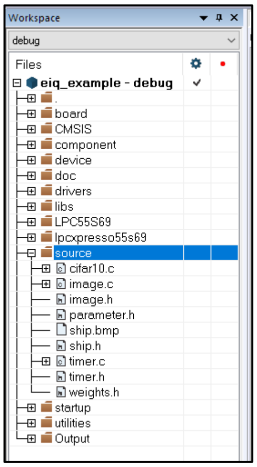
```
eiq_example - debug *
├ board
├ CMSIS
│ ├ NN
│ │ ├ arm_convolve_1x1_HWC_q7_fast_nons...
│ │ ├ arm_convolve_HWC_q15_basic.c
│ │ ├ arm_convolve_HWC_q15_fast.c
│ │ ├ arm_convolve_HWC_q15_fast_nonsqua...
│ │ ├ arm_convolve_HWC_q7_basic.c
│ │ ├ arm_convolve_HWC_q7_basic_nonsqu...
│ │ ├ arm_convolve_HWC_q7_fast.c
│ │ ├ arm_convolve_HWC_q7_fast_nonsquar...
│ │ ├ arm_convolve_HWC_q7_RGB.c
│ │ ├ arm_depthwise_separable_conv_HWC...
│ │ ├ arm_depthwise_separable_conv_HWC...
│ │ ├ arm_fully_connected_mat_q7_vec_q15.c
│ │ ├ arm_fully_connected_mat_q7_vec_q15_...
│ │ ├ arm_fully_connected_q15.c
│ │ ├ arm_fully_connected_q15_opt.c
│ │ ├ arm_fully_connected_q7.c
│ │ ├ arm_fully_connected_q7_opt.c
│ │ ├ arm_nn_activations_q15.c
│ │ ├ arm_nn_activations_q7.c
│ │ ├ arm_nn_mat_mult_kernel_q7_q15.c
│ │ ├ arm_nn_mat_mult_kernel_q7_q15_reord...
│ │ ├ arm_nn_mult_q15.c
│ │ ├ arm_nn_mult_q7.c
│ │ ├ arm_nn_tables.h
│ │ ├ arm_nnfunctions.h
│ │ ├ arm_nnsupportfunctions.h
│ │ ├ arm_nntables.c
│ │ ├ arm_pool_q7_HWC.c
│ │ ├ arm_q7_to_q15_no_shift.c
│ │ ├ arm_q7_to_q15_reordered_no_shift.c
│ │ ├ arm_relu_q15.c
│ │ ├ arm_relu_q7.c
│ │ ├ arm_softmax_q15.c
│ │ └ arm_softmax_q7.c
│ ├ arm_common_tables.h
```

15. With the CMSIS-NN library files added, the next step is to copy in the CMSIS-NN CIFAR10 example.
16. Take the files in the RT1060 SDK **\middleware\eiq\cmsis-nn\Examples\cifar10** folder and copy them into the "source" folder found at <project_location>.
17. Then also take the files in **\middleware\eiq\cmsis-nn\Examples\common\source** and copy them into the same "Source" folder as well at <project_location>. It will look like the following when complete:

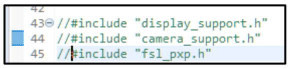


Name	Date modified	Type	Size
cifar10.c	8/19/2020 4:14 PM	C File	9 KB
hello_world.c	8/19/2020 3:55 PM	C File	2 KB
image.c	7/20/2020 8:12 PM	C File	8 KB
image.h	7/20/2020 8:12 PM	H File	4 KB
parameter.h	7/20/2020 8:12 PM	H File	3 KB
ship.bmp	7/20/2020 8:09 PM	BMP File	4 KB
ship.h	7/20/2020 8:12 PM	H File	16 KB
timer.c	7/20/2020 8:12 PM	C File	2 KB
timer.h	7/20/2020 8:12 PM	H File	1 KB
weights.h	7/20/2020 8:12 PM	H File	256 KB

18. Next, drag-and-drop the files from the last step into the group named **source** in the project view. It should look like the following when complete:

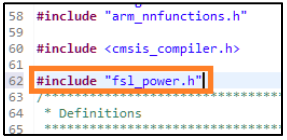


33. Next, the code in **cifar10.c** needs to be updated to work with the LPC55S69. The biggest change will be that the RT1060 has camera and LCD support, which needs to be removed when running on other boards.
    a. Remove or comment out the 3 camera and display header files includes at the top of the file near line 43:



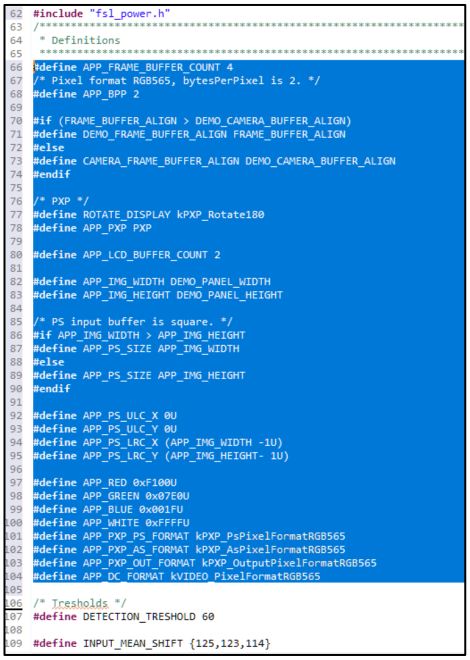
```
43  //#include "display_support.h"
44  //#include "camera_support.h"
45  //#include "fsl_pxp.h"
```

    b. Add **#define "fsl_power.h"** to the list of includes



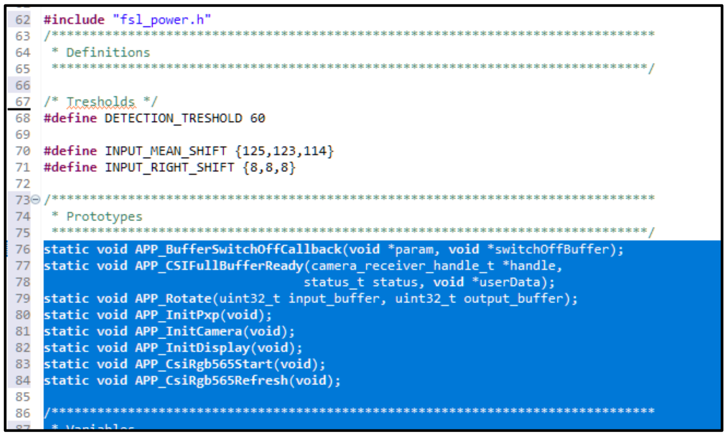
```
58  #include "arm_nnfunctions.h"
59
60  #include <cmsis_compiler.h>
61
62  #include "fsl_power.h"
63  /**********************************
64   * Definitions
65   **********************************
```

    c. Then remove the camera and LCD defines from lines 66 to 104

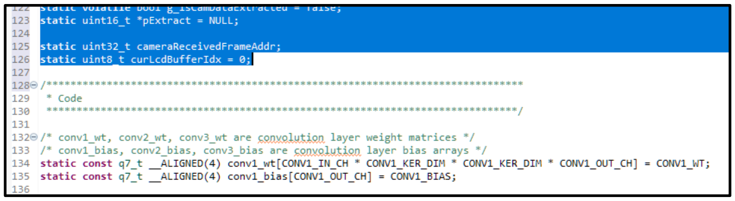


```
62  #include "fsl_power.h"
63  /*****************************************************************
64   * Definitions
65   *****************************************************************
66  #define APP_FRAME_BUFFER_COUNT 4
67  /* Pixel format RGB565, bytesPerPixel is 2. */
68  #define APP_BPP 2
69
70  #if (FRAME_BUFFER_ALIGN > DEMO_CAMERA_BUFFER_ALIGN)
71  #define DEMO_FRAME_BUFFER_ALIGN FRAME_BUFFER_ALIGN
72  #else
73  #define CAMERA_FRAME_BUFFER_ALIGN DEMO_CAMERA_BUFFER_ALIGN
74  #endif
75
76  /* PXP */
77  #define ROTATE_DISPLAY kPXP_Rotate180
78  #define APP_PXP PXP
79
80  #define APP_LCD_BUFFER_COUNT 2
81
82  #define APP_IMG_WIDTH DEMO_PANEL_WIDTH
83  #define APP_IMG_HEIGHT DEMO_PANEL_HEIGHT
84
85  /* PS input buffer is square. */
86  #if APP_IMG_WIDTH > APP_IMG_HEIGHT
87  #define APP_PS_SIZE APP_IMG_WIDTH
88  #else
89  #define APP_PS_SIZE APP_IMG_HEIGHT
90  #endif
91
92  #define APP_PS_ULC_X 0U
93  #define APP_PS_ULC_Y 0U
94  #define APP_PS_LRC_X (APP_IMG_WIDTH -1U)
95  #define APP_PS_LRC_Y (APP_IMG_HEIGHT- 1U)
96
97  #define APP_RED 0xF100U
98  #define APP_GREEN 0x07E0U
99  #define APP_BLUE 0x001FU
100 #define APP_WHITE 0xFFFFU
101 #define APP_PXP_PS_FORMAT kPXP_PsPixelFormatRGB565
102 #define APP_PXP_AS_FORMAT kPXP_AsPixelFormatRGB565
103 #define APP_PXP_OUT_FORMAT kPXP_OutputPixelFormatRGB565
104 #define APP_DC_FORMAT kVIDEO_PixelFormatRGB565
105
106 /* Tresholds */
107 #define DETECTION_TRESHOLD 60
108
109 #define INPUT_MEAN_SHIFT {125,123,114}
```

d. Then remove the LCD and camera defines, function prototypes, and variables from line 76 to 126:

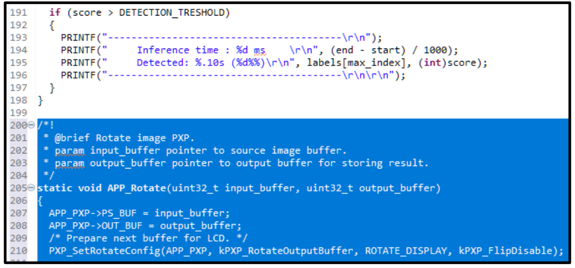


```
62  #include "fsl_power.h"
63  /*******************************************************************************
64   * Definitions
65   ******************************************************************************/
66
67  /* Tresholds */
68  #define DETECTION_TRESHOLD 60
69
70  #define INPUT_MEAN_SHIFT {125,123,114}
71  #define INPUT_RIGHT_SHIFT {8,8,8}
72
73⊖ /*******************************************************************************
74   * Prototypes
75   ******************************************************************************/
76  static void APP_BufferSwitchOffCallback(void *param, void *switchOffBuffer);
77  static void APP_CSIFullBufferReady(camera_receiver_handle_t *handle,
78                                     status_t status, void *userData);
79  static void APP_Rotate(uint32_t input_buffer, uint32_t output_buffer);
80  static void APP_InitPxp(void);
81  static void APP_InitCamera(void);
82  static void APP_InitDisplay(void);
83  static void APP_CsiRgb565Start(void);
84  static void APP_CsiRgb565Refresh(void);
85
86  /*******************************************************************************
87   * Variables
```

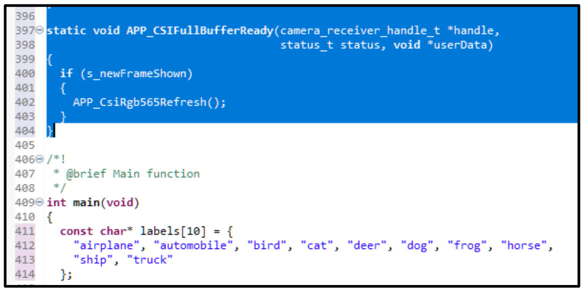


```
122 static volatile bool g_isCamDataExtracted = false;
123 static uint16_t *pExtract = NULL;
124
125 static uint32_t cameraReceivedFrameAddr;
126 static uint8_t curLcdBufferIdx = 0;
127
128⊖/*******************************************************************************
129  * Code
130  ******************************************************************************/
131
132⊖/* conv1_wt, conv2_wt, conv3_wt are convolution layer weight matrices */
133 /* conv1_bias, conv2_bias, conv3_bias are convolution layer bias arrays */
134 static const q7_t __ALIGNED(4) conv1_wt[CONV1_IN_CH * CONV1_KER_DIM * CONV1_KER_DIM * CONV1_OUT_CH] = CONV1_WT;
135 static const q7_t __ALIGNED(4) conv1_bias[CONV1_OUT_CH] = CONV1_BIAS;
136
```

e. Then remove the LCD and camera functions starting around line 200 to line 404 (after deleting the code from the previous steps). It starts at **APP_Rotate()** and goes all the way to **main()**:

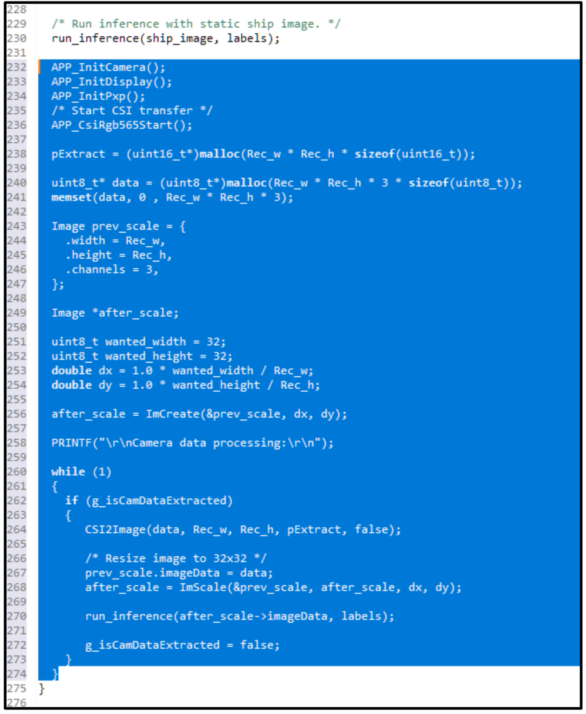


```
191   if (score > DETECTION_TRESHOLD)
192   {
193     PRINTF("---------------------------------------\r\n");
194     PRINTF("     Inference time : %d ms    \r\n", (end - start) / 1000);
195     PRINTF("     Detected: %.10s (%d%%)\r\n", labels[max_index], (int)score);
196     PRINTF("---------------------------------------\r\n\r\n");
197   }
198 }
199
200⊖/*!
201  * @brief Rotate image PXP.
202  * param input_buffer pointer to source image buffer.
203  * param output_buffer pointer to output buffer for storing result.
204  */
205⊖static void APP_Rotate(uint32_t input_buffer, uint32_t output_buffer)
206 {
207   APP_PXP->PS_BUF = input_buffer;
208   APP_PXP->OUT_BUF = output_buffer;
209   /* Prepare next buffer for LCD. */
210   PXP_SetRotateConfig(APP_PXP, kPXP_RotateOutputBuffer, ROTATE_DISPLAY, kPXP_FlipDisable);
```



```
396
397⊖static void APP_CSIFullBufferReady(camera_receiver_handle_t *handle,
398                                    status_t status, void *userData)
399 {
400   if (s_newFrameShown)
401   {
402     APP_CsiRgb565Refresh();
403   }
404 }
405
406⊖/*!
407  * @brief Main function
408  */
409⊖int main(void)
410 {
411   const char* labels[10] = {
412     "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse",
413     "ship", "truck"
414   };
```

f. Finally in main(), remove the code in main() that is after the function call to run_inference(), leaving the last closing bracket at the bottom. Remove lines 232 to 274:



```
228
229   /* Run inference with static ship image. */
230   run_inference(ship_image, labels);
231
232   APP_InitCamera();
233   APP_InitDisplay();
234   APP_InitPxp();
235   /* Start CSI transfer */
236   APP_CsiRgb565Start();
237
238   pExtract = (uint16_t*)malloc(Rec_w * Rec_h * sizeof(uint16_t));
239
240   uint8_t* data = (uint8_t*)malloc(Rec_w * Rec_h * 3 * sizeof(uint8_t));
241   memset(data, 0 , Rec_w * Rec_h * 3);
242
243   Image prev_scale = {
244     .width = Rec_w,
245     .height = Rec_h,
246     .channels = 3,
247   };
248
249   Image *after_scale;
250
251   uint8_t wanted_width = 32;
252   uint8_t wanted_height = 32;
253   double dx = 1.0 * wanted_width / Rec_w;
254   double dy = 1.0 * wanted_height / Rec_h;
255
256   after_scale = ImCreate(&prev_scale, dx, dy);
257
258   PRINTF("\r\nCamera data processing:\r\n");
259
260   while (1)
261   {
262     if (g_isCamDataExtracted)
263     {
264        CSI2Image(data, Rec_w, Rec_h, pExtract, false);
265
266        /* Resize image to 32x32 */
267        prev_scale.imageData = data;
268        after_scale = ImScale(&prev_scale, after_scale, dx, dy);
269
270        run_inference(after_scale->imageData, labels);
271
272        g_isCamDataExtracted = false;
273     }
274   }
275 }
276
```

34. Then the main function in cifar10.c will need to be updated for setting up the clock and board:

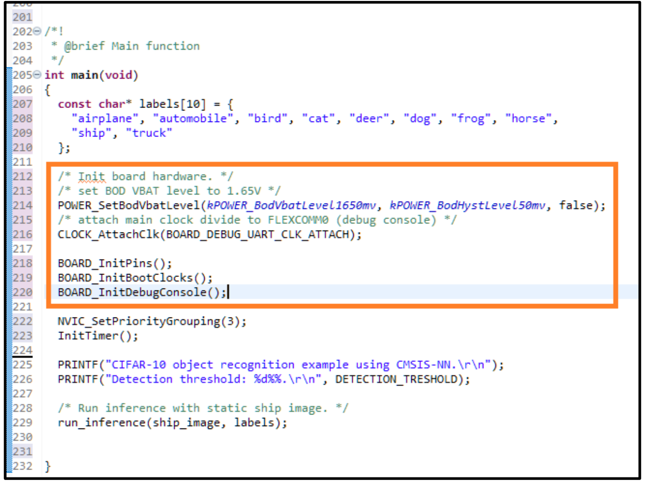
e. Open **hello_world.c** by double clicking on the file name

f. Copy the following lines from **hello_world.c** starting from about line 36 which sets up the clock and board hardware.

```
/* Init board hardware. */
/* set BOD VBAT level to 1.65V */
POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
/* attach main clock divide to FLEXCOMM0 (debug console) */
CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

BOARD_InitPins();
BOARD_InitBootClocks();
BOARD_InitDebugConsole();
```

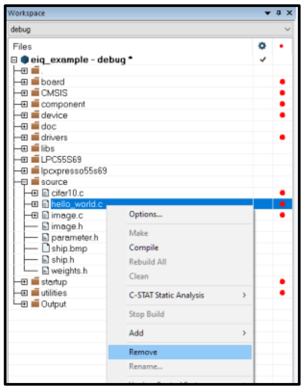
g. Overwrite the similar board setup code in **cifar10.c** near the beginning of **main(void)**.

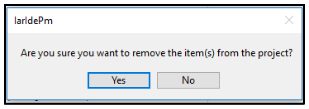
h. **main() in cifar10.c** will look like the following when complete:



```
/*!
 * @brief Main function
 */
int main(void)
{
  const char* labels[10] = {
    "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse",
    "ship", "truck"
  };

  /* Init board hardware. */
  /* set BOD VBAT level to 1.65V */
  POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
  /* attach main clock divide to FLEXCOMM0 (debug console) */
  CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

  BOARD_InitPins();
  BOARD_InitBootClocks();
  BOARD_InitDebugConsole();

  NVIC_SetPriorityGrouping(3);
  InitTimer();

  PRINTF("CIFAR-10 object recognition example using CMSIS-NN.\r\n");
  PRINTF("Detection threshold: %d%%.\r\n", DETECTION_TRESHOLD);

  /* Run inference with static ship image. */
  run_inference(ship_image, labels);


}
```

19. Finally, remove the **hello_world.c** from the IAR project by right clicking on that file and selecting **Remove**. IAR will pop up a dialog asking if you are sure, so click on Yes to remove the file.



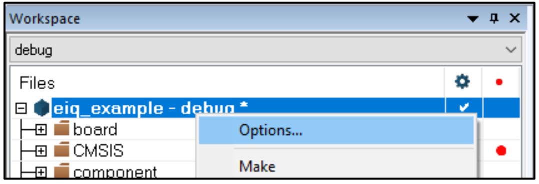

IarIdePm

Are you sure you want to remove the item(s) from the project?

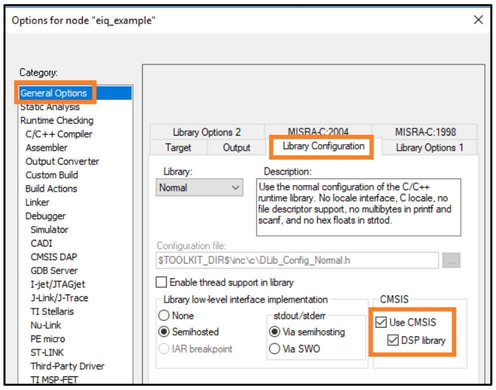
6.3 IAR Compiler Settings

The final section is to adjust the IAR project settings.

20. Right click on the project name and select Options



21. In the **General Options** category, in **Library Configuration** tab, check "**Use CMSIS**" and check "**DSP Library**". This adds CMSIS math support to the project which is used by CMSIS-NN.


Options for node "eiq_example"

Category: General Options

Library: Normal

Description: Use the normal configuration of the C/C++ runtime library. No locale interface, C locale, no file descriptor support, no multibytes in printf and scanf, and no hex floats in strtod.

Configuration file: $TOOLKIT_DIR$\inc\c\DLib_Config_Normal.h

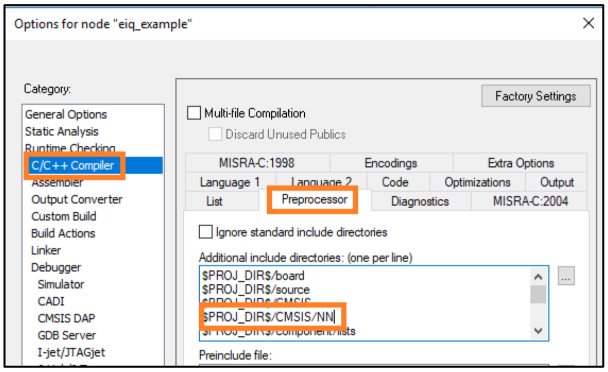
☐ Enable thread support in library

Library low-level interface implementation: ○ None ● Semihosted ○ IAR breakpoint

stdout/stderr: ● Via semihosting ○ Via SWO

CMSIS: ☑ Use CMSIS ☑ DSP library

22. The NN directory that was created must be added to the include directory list. In the **C/C++ Compiler** category, go the **Preprocessor** tab and in the **Additional include directories** box, add a line for **$PROJ_DIR$/CMSIS/NN**



☐ Multi-file Compilation

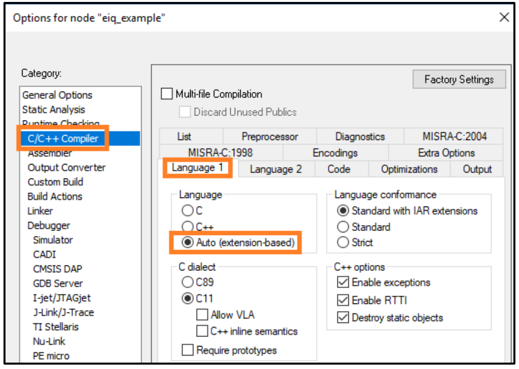
☐ Discard Unused Publics

☐ Ignore standard include directories

Additional include directories: (one per line)

```
$PROJ_DIR$/board
$PROJ_DIR$/source
$PROJ_DIR$/CMSIS
$PROJ_DIR$/CMSIS/NN
$PROJ_DIR$/component/lists
```

23. Also in the **C/C++ Compiler** category, in the **Language 1** tab in the **Language** box, select **Auto**



☐ Multi-file Compilation

☐ Discard Unused Publics

Language: ○ C ○ C++ ● Auto (extension-based)

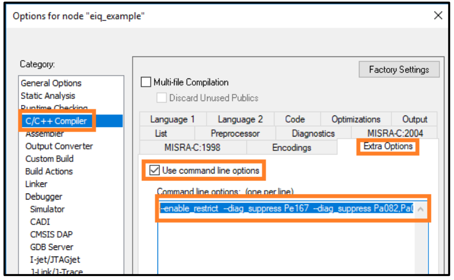
Language conformance: ● Standard with IAR extensions ○ Standard ○ Strict

C dialect: ○ C89 ● C11 ☐ Allow VLA ☐ C++ inline semantics

C++ options: ☑ Enable exceptions ☑ Enable RTTI ☑ Destroy static objects

☐ Require prototypes

24. Also in the **C/C++ Compiler** category, in the **Extra Options** tab, check "**Use command line options**" and add the following text. This is not needed for CIFAR-10 demo but would be needed if using the KWS demo. (Note the double dashes before each option)
--enable_restrict --diag_suppress Pe167 --diag_suppress Pa082,Pa050



☐ Multi-file Compilation

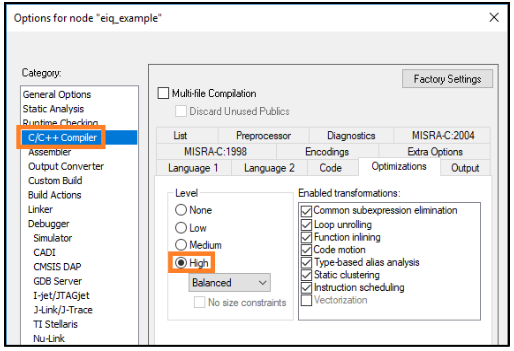
☐ Discard Unused Publics

☑ Use command line options

Command line options: (one per line)

```
--enable_restrict --diag_suppress Pe167 --diag_suppress Pa082,Pa
```

25. eIQ inference time is highly sensitive to code optimization settings. This can be adjusted in the C/C++ Compiler category in the **Optimizations** tab. Set to **High** for the highest optimization.



☐ Multi-file Compilation

☐ Discard Unused Publics

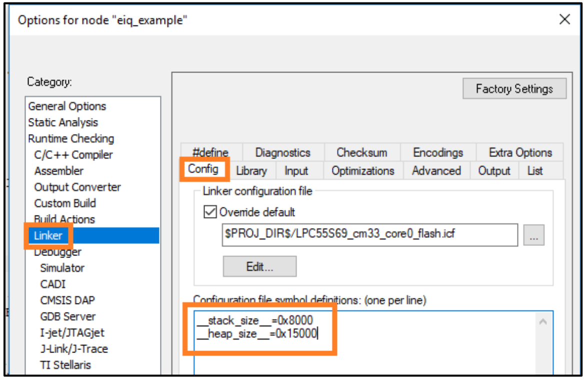
Level: ○ None ○ Low ○ Medium ● High

Balanced

☐ No size constraints

Enabled transformations:
☑ Common subexpression elimination
☑ Loop unrolling
☑ Function inlining
☑ Code motion
☑ Type-based alias analysis
☑ Static clustering
☑ Instruction scheduling
☐ Vectorization

26. In the **Linker** category, under the **Config** tab, add the following to the linker options (double underscore on both sides of the stack_size and heap_size. This increases the default stack size, as otherwise it will run out of stack space:
__stack_size__=0x8000
__heap_size__=0x15000



Options for node "eiq_example"

Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
Simulator
CADI
CMSIS DAP
GDB Server
I-jet/JTAGjet
J-Link/J-Trace
TI Stellaris

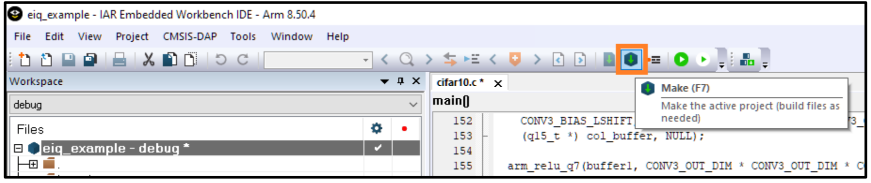
Factory Settings

#define | Diagnostics | Checksum | Encodings | Extra Options
Config | Library | Input | Optimizations | Advanced | Output | List

Linker configuration file
☑ Override default
$PROJ_DIR$/LPC55S69_cm33_core0_flash.icf
Edit...

Configuration file symbol definitions: (one per line)
```
__stack_size__=0x8000
__heap_size__=0x15000
```

27. Click on OK to save these settings.
28. Then click on the Make button to build the project. It should compile without errors.

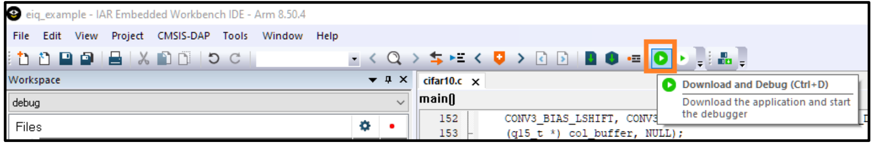


eiq_example - IAR Embedded Workbench IDE - Arm 8.50.4

Make (F7)
Make the active project (build files as needed)

```
152    CONV3_BIAS_LSHIFT
153      (q15_t *) col_buffer, NULL);
154
155    arm_relu_q7(buffer1, CONV3_OUT_DIM * CONV3_OUT_DIM * C
```

29. Connect a USB cable to the board and open a terminal program and connect to the COM port that the LPC board enumerated as.
30. Run and debug the project as normal.

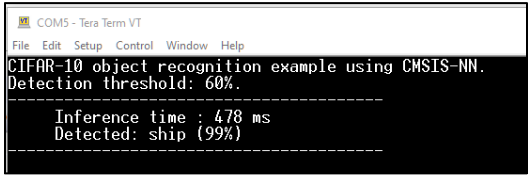


eiq_example - IAR Embedded Workbench IDE - Arm 8.50.4

Download and Debug (Ctrl+D)
Download the application and start the debugger

```
152    CONV3_BIAS_LSHIFT, CONV3
153      (q15_t *) col_buffer, NULL);
```

31. You should see the following output on the terminal:



COM5 - Tera Term VT

```
CIFAR-10 object recognition example using CMSIS-NN.
Detection threshold: 60%.
----------------------------------------
      Inference time : 478 ms
      Detected: ship (99%)
----------------------------------------
```