eIQ: Glow Porting for MCUs

Revision 1

Contents

1 Document Overview

This document describes how to port eIQ for Glow to a new MCU, such as the LPC55S69. It covers porting with MCUXpresso IDE, IAR Embedded Workbench for ARM, and Keil MDK.

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 Glow to the LPC55S69. The instructions in this document can also be used to port eIQ for Glow to other devices as well, including other devices in the i.MX RT family.

3 Prerequisites

The following items will be needed:

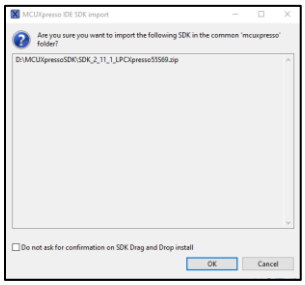
a) MCUXpresso SDK for LPC55S69
b) MCUXpresso SDK for RT1060 - make sure to include the "eIQ" middleware option.
c) MCUXpresso IDE – if using MCUXpresso IDE
d) MCUXpresso Config Tools – if using IAR or Keil

4 MCUXpresso IDE Porting Glow using CIFAR-10 Demo

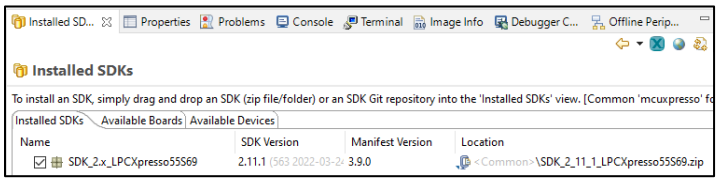
4.1 Cloning a Project

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 Glow source into.

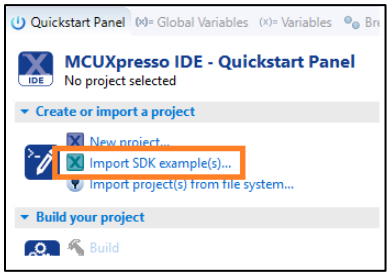
1. Unzip both the LPC55S69 and the RT1060 SDK zip files into a directory path that does not contain spaces. The files inside both zip files will be used in later steps.
2. Open up MCUXpresso IDE and select a new workspace
3. Install the LPC55S69 SDK into the "Installed SDKs" tab by dragging and dropping the zipped **SDK_2_11_1_LPCXpresso55S69.zip** file into the Installed SDKs window. This dialog box will come up, and click OK to continue the import



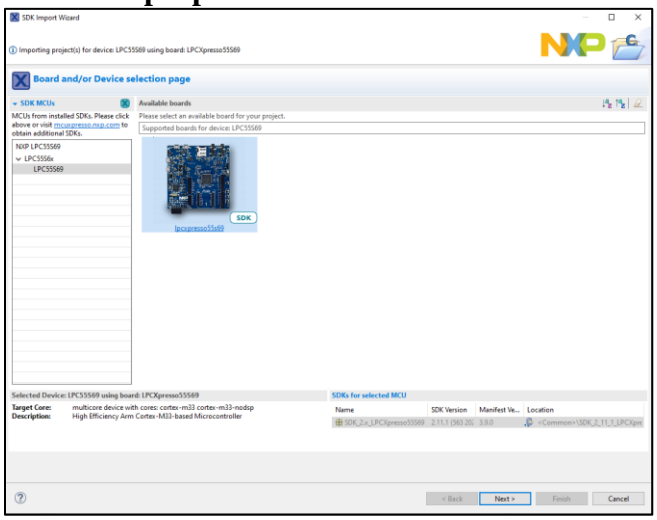
4. It will look something like the following when complete:



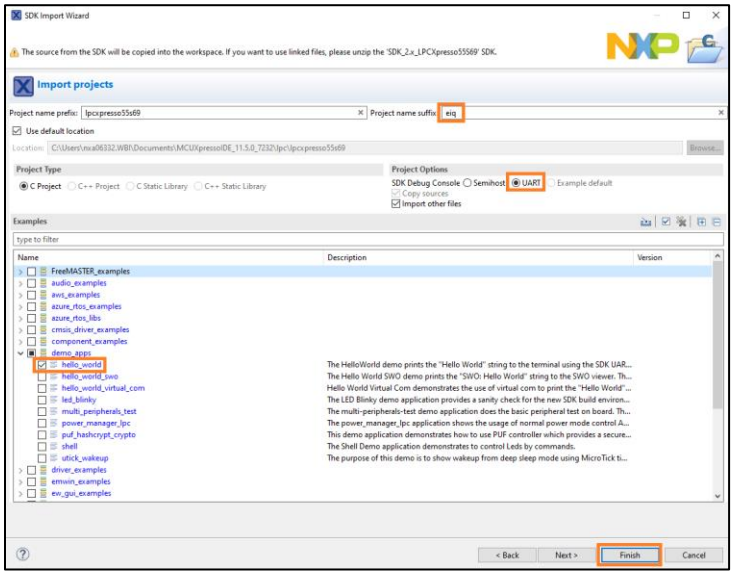
5. Next import the desired project. In the Quickstart Panel, select **Import SDK examples(s)…**



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



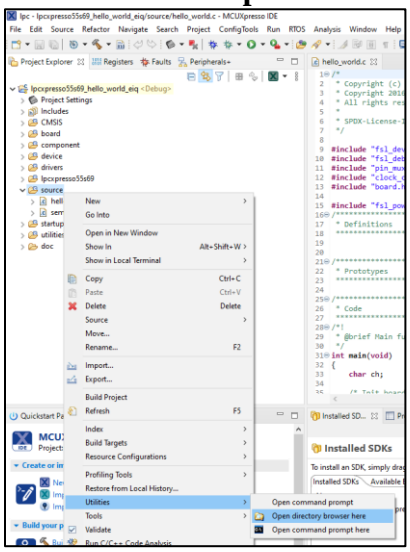
7. Open up the **demo_apps** category and select the **hello_world** project. In the **Project name suffix** box, change it to **eiq** to make it unique. Change the **SDK Debug Console** to **UART**. Then click on **Finish**.



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

4.2 Source Files

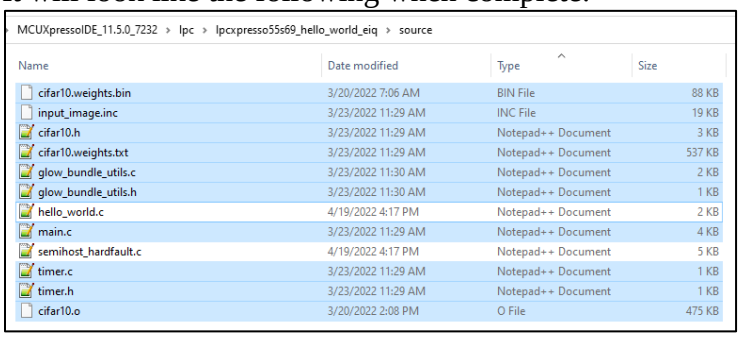
1. If not done already, unzip the RT1060 SDK. The eIQ source files will be used in the next steps.
2. In MCUXpresso IDE, right click on the source direcotyr in the Project Explorer window and select **Utilities->Open directory browser here**



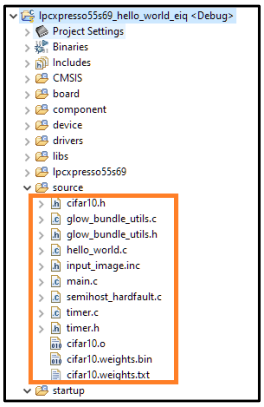
3. We will copy some files into the Windows Explorer window that comes up. It should be something like **C:\Users\username\Documents\MCUXpressoIDE_11.5.0_7232\workspace\lpcxpresso55s69_hello_world_eiq\source**
4. Open a new Windows Explorer, and inside the unzipped RT1060 SDK go to **\boards\evkmimxrt1060\eiq_examples\glow_cifar10\source** and copy all the files in that directory to the **lpcxpresso55s69_hello_world_eiq\source** directory from the previous step.

   Note that the default RT1060 glow bundle that was just copied into the LPC project was compiled for a Cortex M7 core, but it will run on the LPC55S69 M33 core with MCUXpresso IDE. However when compiling a model with Glow for a LPC55S69 device it is recommend to use the *-mcpu=cortex-m33* Glow compile option.

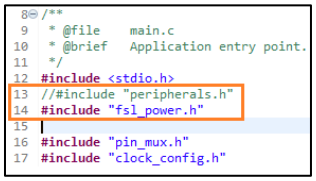
5. Then still inside the unzipped RT1060 SDK go to **\middleware\eiq\glow\bundle_utils** and copy both **glow_bundle_utils** .c and .h files in that directory to that same **lpcxpresso55s69_hello_world_eiq\source** directory from the previous step.
6. It will look like the following when complete:



7. 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:



8. Next the code in **main.c** needs to be updated to work with the LPC55S69.
   a. Open up main.c by double clicking on the file name.
   b. Comment out line 13 which #includes the peripherals.h file
   c. Add a line below it with **#include "fsl_power.h"**
   d. It should look like the following when done:

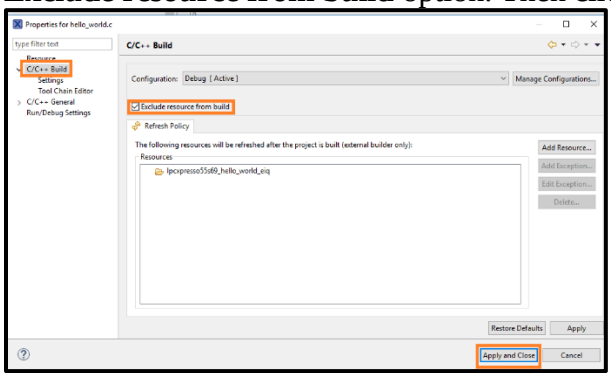


   e. Next, open **hello_world.c** by double clicking on the file name
   f. Copy the following lines from **hello_world.c** starting from about line 35 which sets up the clock and board hardware.

```
/* 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. Then open up **main.c** to overwrite the similar board setup code in **main.c** near the beginning of **main(void)** up to and including the **BOARD_InitDebugConsole()** code. Make sure to not modify the init_timer() function call though.
   h. It should look like the following when added:

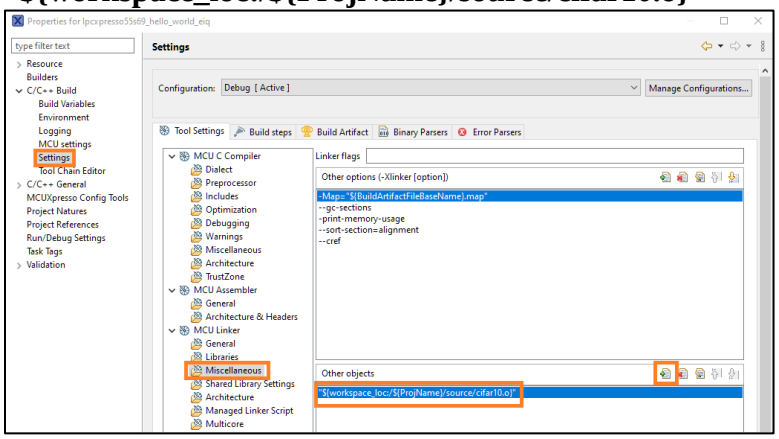
9. Finally, prevent the **hello_world.c** file from compiling by right clicking on that file and selecting **Properties**. In the dialog box that comes up, select the **C/C++ Build** category, and check the **Exclude resource from build** option. Then click **Apply and Close**.



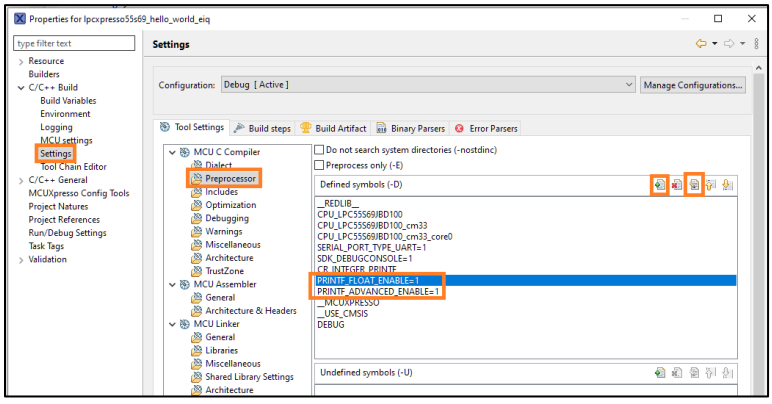
4.3 MCUXpresso IDE Project Settings

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

1. Open the Properties dialog box by right clicking on the project name and selecting Properties (like done in the previous section).
2. Add the cifar.o compiled code from the Glow compiler output by going to the **Miscellaneous** settings and under **Other objects** use the Add button to include the object file **"${workspace_loc:/${ProjName}/source/cifar10.o}"**

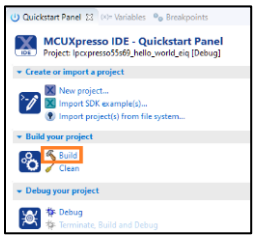


3. Next enable floating point printf support by going to the Preprocessor settings and edit **PRINTF_FLOAT_ENABLE=1**, and then also add **PRINTF_ADVANCED_ENABLE=1**.



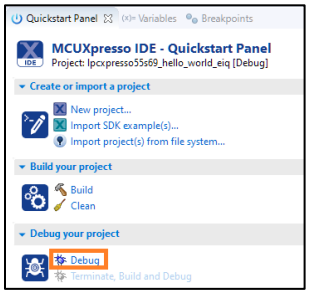
4. Click on **Apply and Close** to save these settings and close the project options dialog box.

5. Build the project using the Quickstart Panel and make sure there are no compile errors in the Console tab in the center bottom.

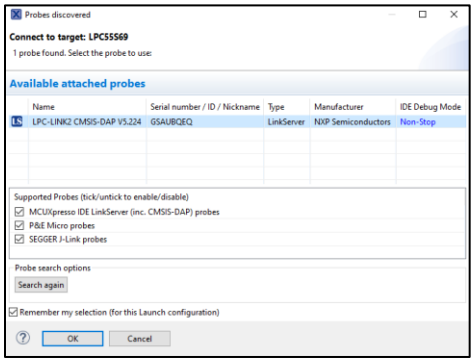


4.4 Run Example

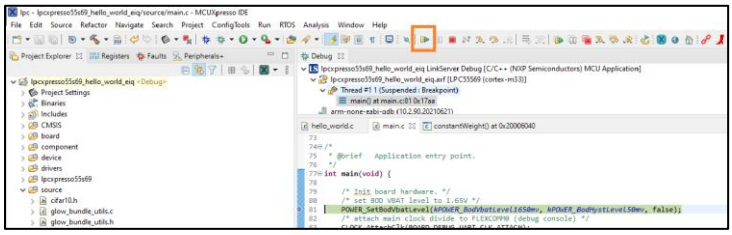
6. Plug the micro-B USB cable into the board at J11 on the i.MXRT1170 board.
7. Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity.
8. Debug the project by clicking on “Debug” in the Quickstart Panel.



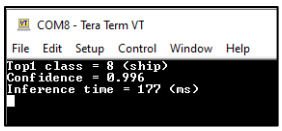
9. It will ask what interface to use. Select **LPC-LINK2 CMSIS-DAP**.



10. You may get an additional dialog box about which core to use. Leave it as default.
11. The debugger will download the firmware and open up the debug view. It may take some time to download the firmware. Click on the Resume button to start running.



12. You should see the following output in the terminal:

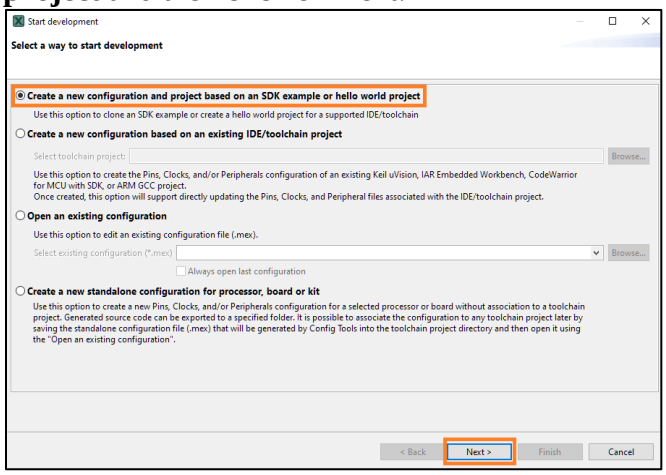


5 IAR Porting Glow using CIFAR-10 Demo

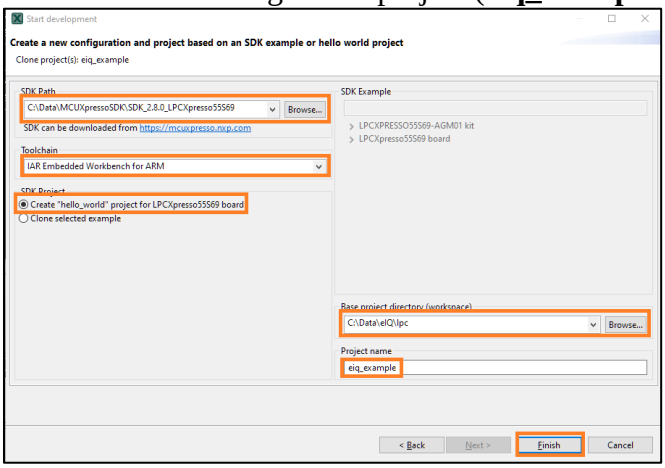
This section will cover how to port Glow in IAR. A CIFAR-10 image classification model will be used as an example, and it must be compiled by Glow specifically for the Cortex-M33 core found on the LPC55S69. The RT685 SDK Glow bundle for Cortex-M33 cannot be used because it uses HiFi4 DSP calls not available for the LPC55S69. Generating the CIFAR10 model and generating the new Glow CIFAR10 bundle for Cortex-M33 is out of the scope of this document. Please see the Glow documentation for more details on that process.

5.1 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.



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

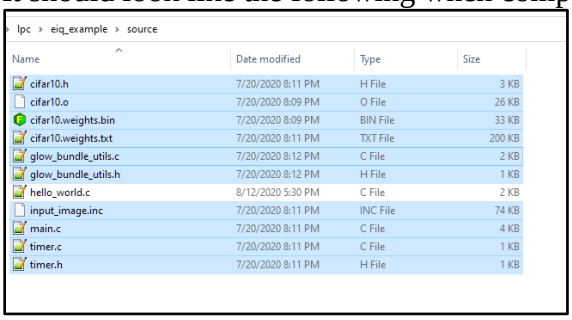


5. A new project will be created in the directory you specified.

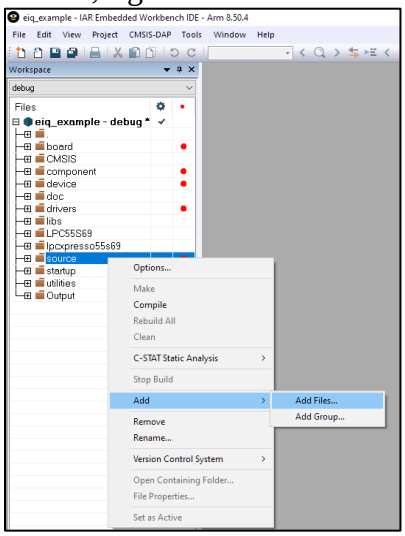
5.2 Source Files

6. In Windows Explorer, go to the project directory created in the last section (ie **C:\Data\eIQ\lpc\eiq_example**), and then go to the **source** folder. We will copy some files into this folder in the next step.

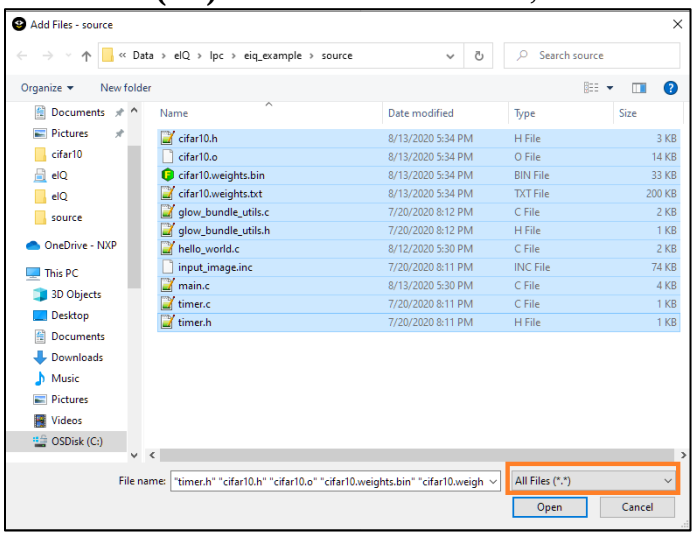
7. Open a new Windows Explorer, and inside the unzipped RT1060 SDK go to **\boards\evkmimxrt1060\eiq_examples\glow_cifar10\source** and copy the **input_image.inc**, **main.c**, **timer.c**, and **timer.h** files in that directory to the **eiq_example \source** directory from the previous step.
8. Because the Glow bundle in the RT1060 example was compiled for a Cortex-M7 core, a new version of the bundle must be created for the Cortex-M33 core on the LPC device because the IAR compiler cannot use a bundle compiled for the M7 core. In this example we'll use the CIFAR10 model, which can be created by following the directions in **\boards\evkmimxrt1060\eiq_examples\glow_cifar10\doc\readme.txt**. Once done, copy the newly generated Glow bundle into the same **eiq_example\source** folder.
9. Then still inside the unzipped RT1060 SDK go to **\middleware\eiq\glow\bundle_utils** and copy both files in that directory to that same **eiq_example\source** directory from the previous step.
10. It should look like the following when complete:



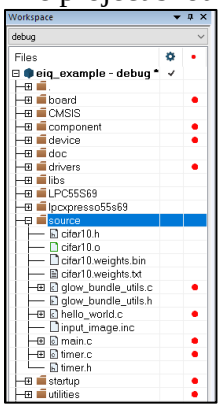
11. Open IAR and open the new project created in the last section by going to File->Open Workspace and navigating to the directory to open the .eww file
12. In IAR, right click on the **source** folder and go to **Add->Add Files…**



13. In the dialog box that comes up, navigate to the **eiq_example\source** folder and change the filter to **All Files (*.*)**. Then select all the files, and then click on **Open** to add them to the IAR project



14. The project should look like the following after all the files have been added:



15. Next the code in **main.c** needs to be updated to work with the LPC55S69.
    a. Open **hello_world.c** by double clicking on the file name
    b. On line 14, comment out the **#include "peripherals.h"** line
    c. Add **#include "fsl_power.h"**
    d. It should look like the following when done:

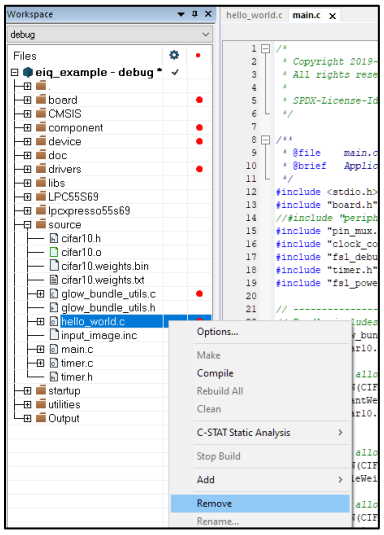
    e. Copy the following lines from **hello_world.c** starting from about line 36 which sets up the clock and board hardware.

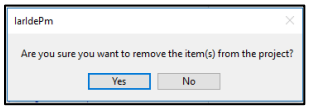
```
/* Init board hardware. */
/* set BOD VBAT level to 1.65V */
POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
/* attach main clock divide to FLEXCOMM0 (debug console) */
CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

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

f. Overwrite the similar board setup code in **main.c** near the beginning of **main(void)**.
g. It should look like the following when added:

```
/*
 * @brief   Application entry point.
 */
int main(void) {

    /* 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();

  init_timer();

  // Timer variables.
  uint32_t start_time, stop_time;
  uint32_t duration_ms;
```

16. 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.

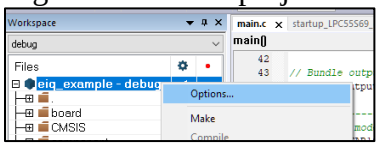




5.3 IAR Project Settings

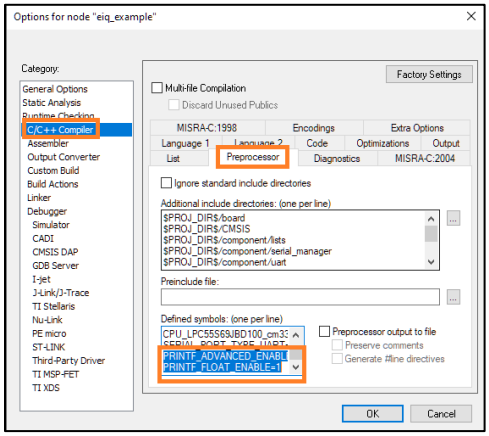
The final changes involve adjusting the IAR project settings.

17. Right click on the project name and select **Options…**

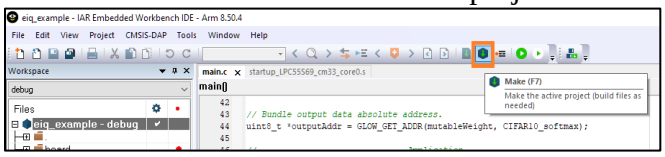


18. In the **C/C++ Compiler** category, go to the **Preprocessor** tab, and add the following to the Defined symbols box. Then click on **OK** to save the changes.

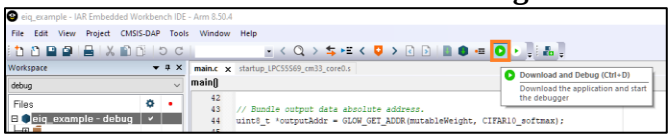
PRINTF_ADVANCED_ENABLE=1
PRINTF_FLOAT_ENABLE=1



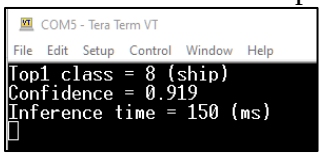
19. Click on the **Make** icon to build the project



20. Open a terminal program and connect to the COM port the board enumerated as
21. Then click on the **Download and Debug** icon to load and debug the project



22. You should see the output in the terminal:

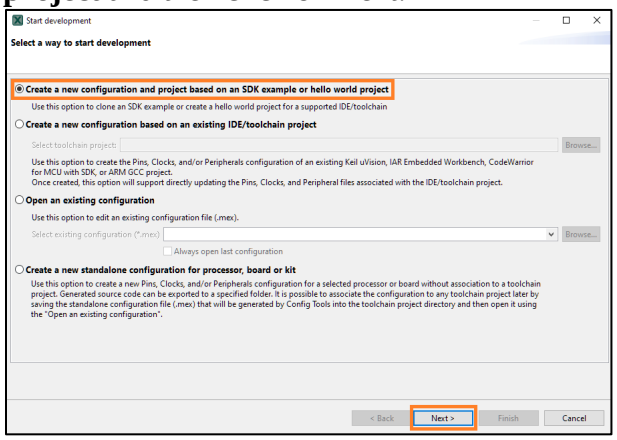


6 Keil MDK Porting Glow using CIFAR-10 Demo

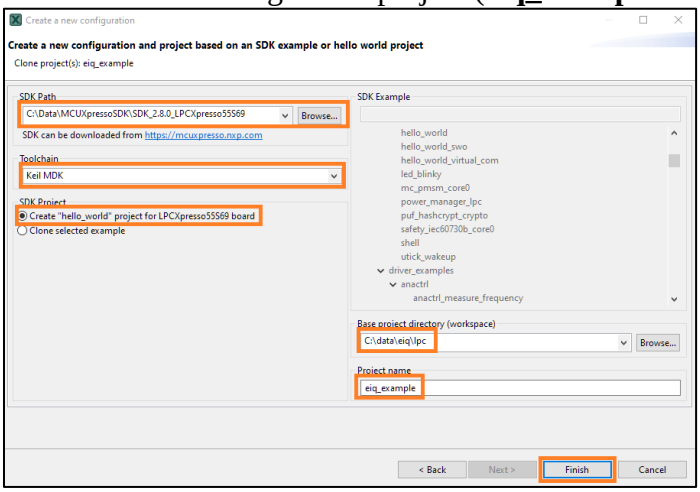
This section will cover how to port Glow in Keil MDK. A CIFAR-10 image classification model will be used as an example, and it must be compiled by Glow specifically for the Cortex-M33 core found on the LPC55S69. The RT685 SDK Glow bundle for Cortex-M33 cannot be used because it uses HiFi4 DSP calls not available for the LPC55S69. Generating the CIFAR10 model and generating the new Glow bundle is out of the scope of this document. Please see the Glow documentation for more details on that process.

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.



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

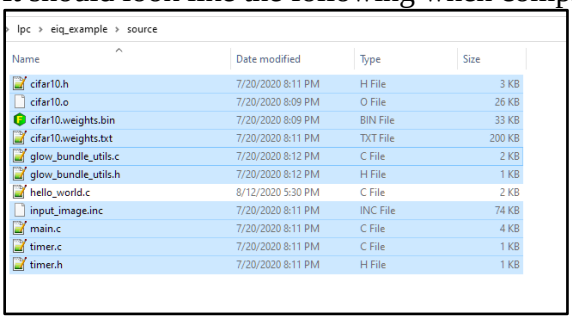


5. A new project will be created in the directory you specified.

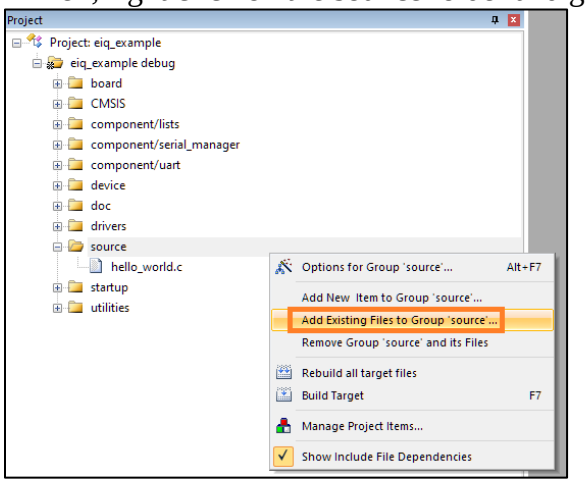
6.2 Source Files

6. In Windows Explorer, go to the project directory created in the last section (ie **C:\Data\eIQ\lpc\eiq_example**), and then go to the **source** folder. We will copy some files into this folder in the next step.

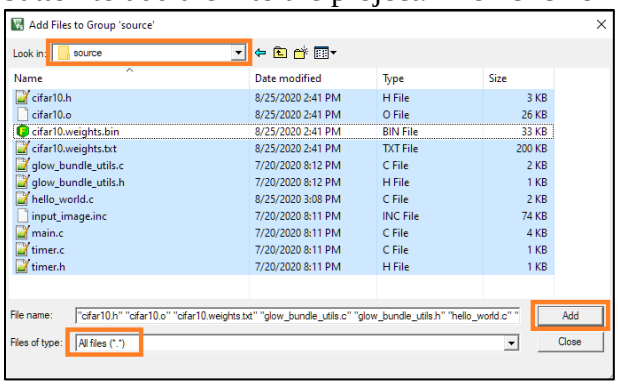
7. Open a new Windows Explorer, and inside the unzipped RT1060 SDK go to **\boards\evkmimxrt1060\eiq_examples\glow_cifar10\source** and copy the **input_image.inc**, **main.c**, **timer.c**, and **timer.h** files in that directory to the **eiq_example \source** directory from the previous step.
8. Because the Glow bundle in the RT1060 example was compiled for a Cortex-M7 core, a new version of the bundle must be created for the Cortex-M33 core on the LPC device because the Keil compiler cannot use a bundle compiled for the M7 core. In this example we'll use the CIFAR10 model, which can be created by following the directions in **\boards\evkmimxrt1060\eiq_examples\glow_cifar10\doc\readme.txt**. Copy the newly generated bundle into the same **eiq_example\source** folder.
9. Then still inside the unzipped RT1060 SDK go to **\middleware\eiq\glow\bundle_utils** and copy both files in that directory to that same **eiq_example\source** directory from the previous step.
10. It should look like the following when complete:



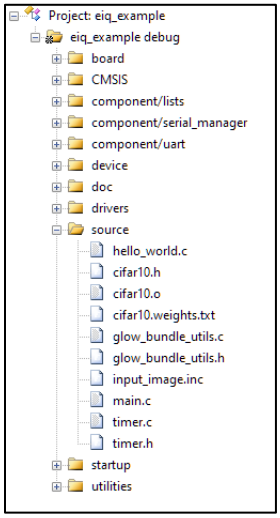
11. Open Keil and open the new project created in the last section by going to File->Open Workspace and navigating to the directory to open the .uvprojx file. If needed, install any required device packs.
12. In Keil, right click on the **source** folder and go to **Add Existing Files to Group 'source'…**



13. In the dialog box that comes up, navigate to the **eiq_example\source** folder. Set the filer to **All files**(*.*), and then select all the files except for the **cifar10.weights.bin** file. Click on the **Add** button to add them to the project. Then click on **Close** to close the dialog box.



14. It should look like the following when completed:



23. Next the code in **main.c** needs to be updated to work with the LPC55S69.
    a. Open **hello_world.c** by double clicking on the file name
    b. On line 14, comment out the **#include "peripherals.h"** line
    c. Add **#include "fsl_power.h"**
    d. It should look like the following when done:

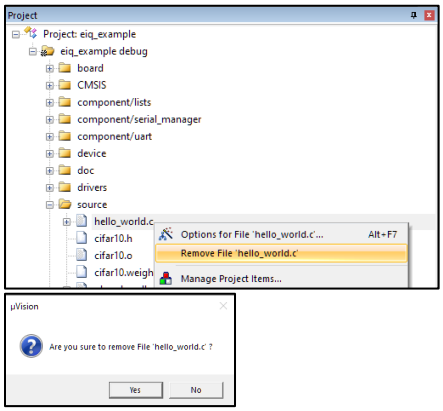
    e. Copy the following lines from **hello_world.c** starting from about line 36 which sets up the clock and board hardware.

```
/* 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();
```

f. Overwrite the similar board setup code in **main.c** near the beginning of **main(void)**.
g. It should look like the following when added:

```
/*
 * @brief   Application entry point.
 */
int main(void) {

    /* 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();

  init_timer();

  // Timer variables.
  uint32_t start_time, stop_time;
  uint32_t duration_ms;
```

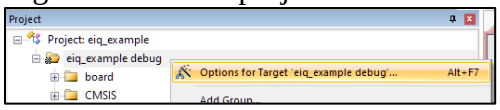
24. Finally, remove the **hello_world.c** from the Keil project by right clicking on that file and selecting **Remove File 'hello_world.c'**. Keil will pop up a dialog asking if you are sure, so click on **Yes** to remove the file.



6.3 Keil Project Settings

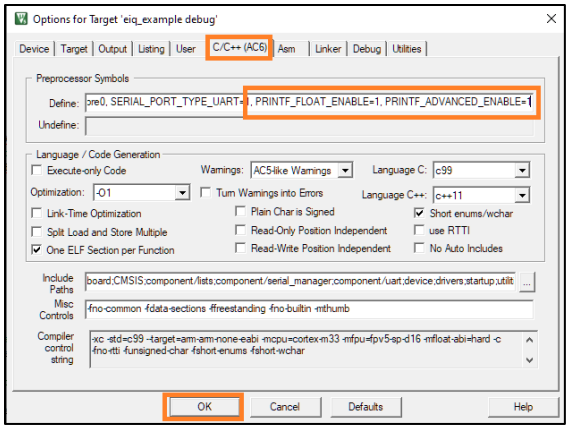
The final changes involve adjusting the Keil project settings.

25. Right click on the project name and select **Options for Target 'eiq_example debug'…**

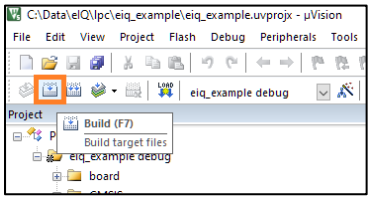


26. Go to the **C/C++ (AC6)** tab and add the following to the **Define** box. Make sure to add the commas to delineate the new defines. Then click on **OK** to save the changes.

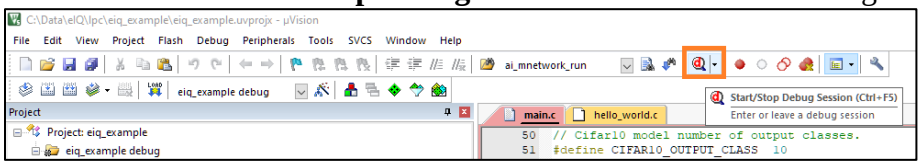
, PRINTF_ADVANCED_ENABLE=1, PRINTF_FLOAT_ENABLE=1



15. Click on the Build icon to compile the project



27. Open a terminal program and connect to the COM port the board enumerated as
28. Then click on the **Start/Stop Debug Session** icon to load and debug the project



29. You should see the output in the terminal: