Using Keil™ ULINKpro with the AT&T® Cellular IoT Starter Kit

Make problems easier to solve with the right tools…

Table of Contents

Overview

The purpose of this paper is to discuss and demonstrate how to use the ARM Keil MDK toolkit with µVision® IDE with the Avnet-AT&T Cellular IoT Kit at http://cloudconnectkits.org/product/att-cellular-iot-starter-kit.

The paper uses the Avnet-AT&T Cellular IoT Kit software which is located at https://developer.mbed.org/users/JMF/code/Avnet_ATT_Cellular_IOT/ and focuses on how to connect the ULINK*pro* JTAG debugger to the hardware, what to modify in the software, and how to configure the µVision IDE.

Additional information on the ULINK tools & Keil Environment can be found in the following App Notes from Keil, NXP K64 Freedom: www.keil.com/appnotes/docs/apnt_287.asp and NXP K64 Tower: www.keil.com/appnotes/docs/apnt_288.asp.

Why Use Keil MDK?

Development on the NXP/FRDM-K64F is simplified for users by having OpenSDA tools pre-installed when it is delivered. While this environment is suitable for many development activities, it lacks the hardware and performance specific capabilities that a USB-JTAG/SW Debug and Trace Unit provide. Using the Keil µVision IDE and ULINK*pro* environment, users enjoy the following:

- A µVision IDE with Integrated Debugger and Flash programmer
- The ARM® Compiler tool chain
- USB-JTAG Debugging (and/or Serial Wire Debug) with CoreSight™ Serial Wire Viewer and ETM trace capability

Other Keil tools such as ULINK2 and ULINK-ME and the Segger J-Link can be used in place of the ULINK*pro* but these units do not support the ETM, Performance Analysis, and Code Coverage features.

Finally, being able to switch between the online development at mbed.org and the ARM/Keil environment is the goal of this paper. While integrating the Avnet WNC-Shield board, more extensive hardware debug capabilities were needed which drove the need for this capability.

ULINK*pro* features

The specific features of the ULINK*pro* that were needed are as follows:

1. Serial Wire Viewer (SWV) data trace including exceptions and ETM instruction trace
2. Real-time read and write to memory locations for watch, and memory windows
3. Hardware breakpoints and watchpoints (also called access breaks)

Introduction to the MDK

This paper uses MDK 5.20 and Software Pack 1.4.0. Install MDK 5 Core first and then download the required Software Packs. Software Packs are an ARM CMSIS standard[1].

[1] See www.keil.com/CMSIS or www.keil.com/dd2/pack for the complete list of software packs.

Installing the Software

Download and Install

1. Download MDK Core from the Keil website at www.keil.com/mdk5/install.
2. Install MDK into the default folder. You can install into any folder, but for this example the default is C:\Keil_v5.

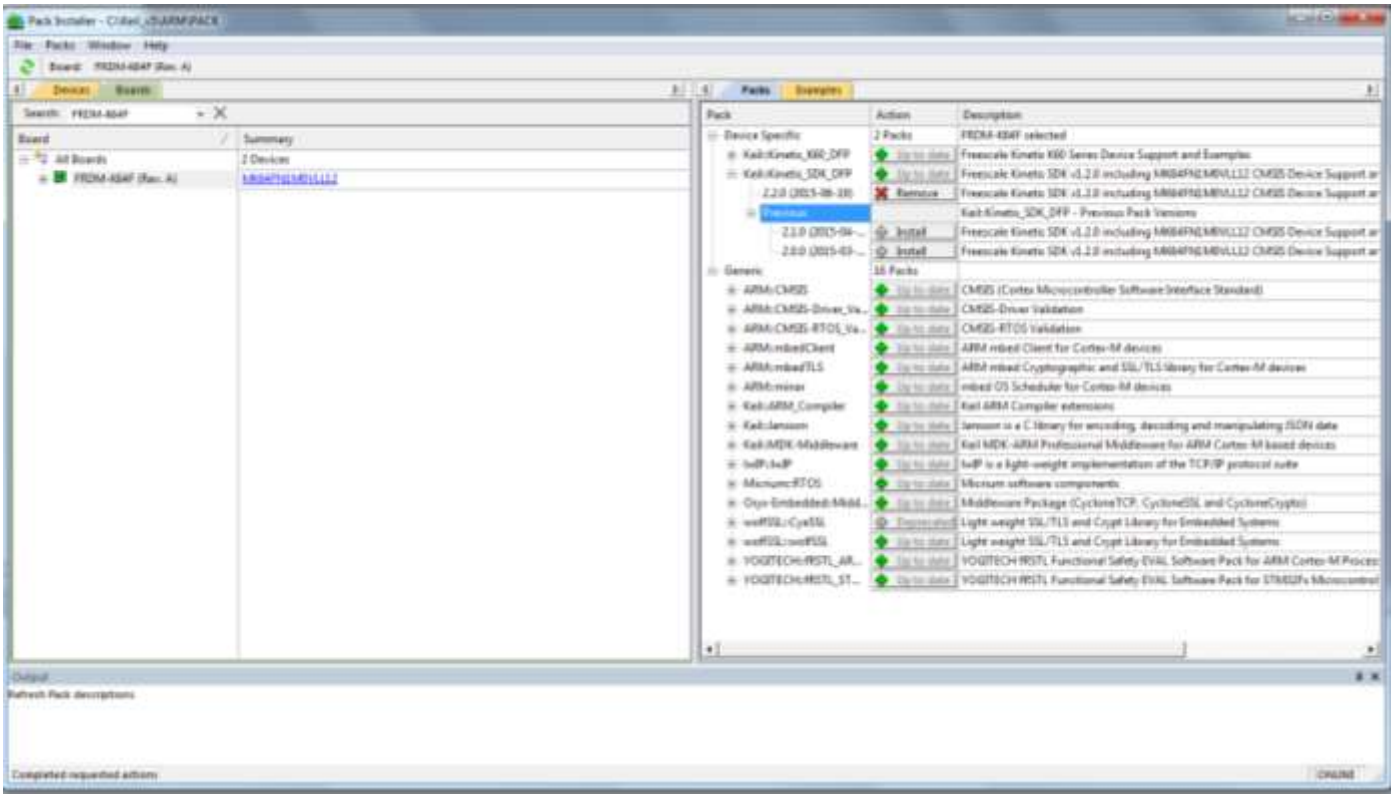
Selecting Software Packs

This µVision utility provides the ability to choose various Software Pack versions to be installed.

1. Start µVision and open Pack Installer.

After the first MDK install is complete, and if you are connected to the Internet, µVision and software packs will automatically start. Otherwise, connect your computer to the Internet to download the software packs and start µVision by clicking the desktop icon. (Initially, the pack list must be downloaded from the Web).

2. Click the Pack Installer icon. A Pack Installer Welcome screen opens.



3. Select the Boards tab.
4. Type FRDM-K64F in the Search box to filter the listings.
5. Select FRDM-K64F. You can also select individual processors under the Devices tab.

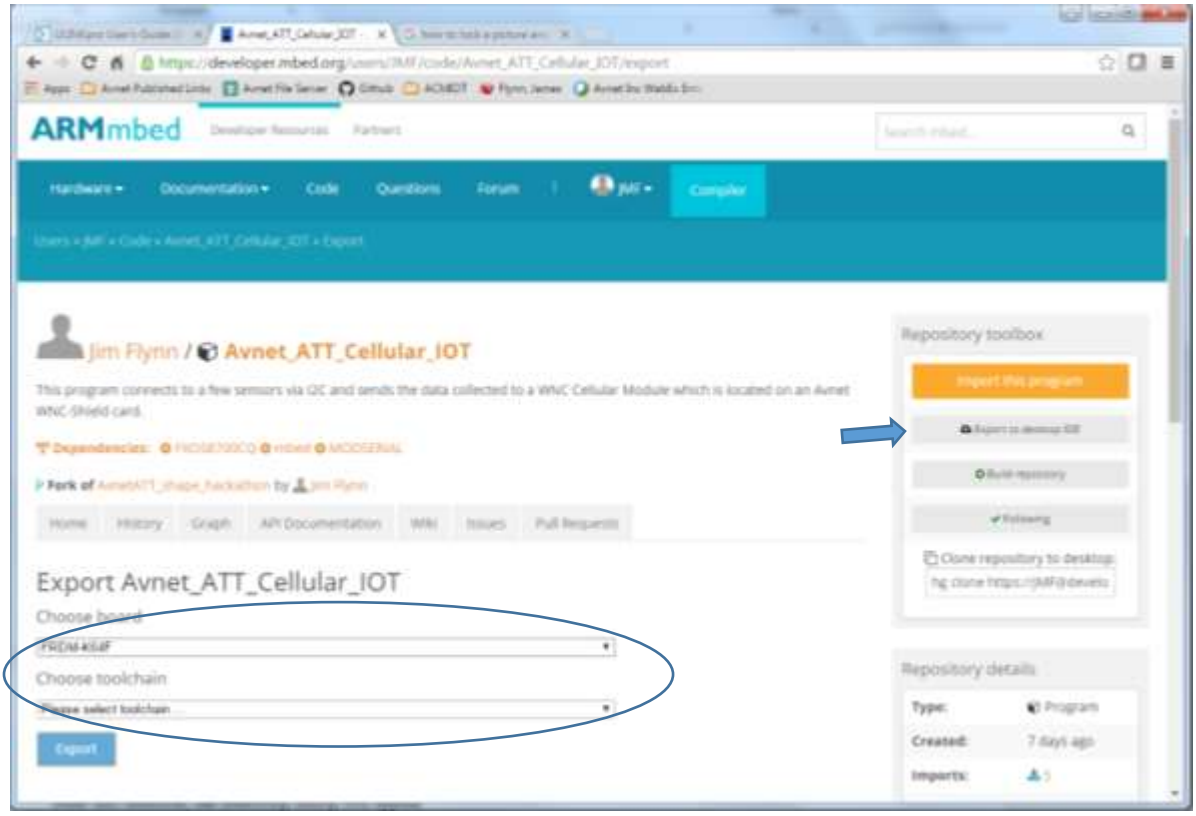
NOTE: If there are no entries shown, you were not connected to the Internet when Pack Installer opened; select Packs/Check for Updates or refresh once you have connected to the Internet. This is not done automatically.

6. Install the Kinetis K60 Device Family Pack (K60_DFP) by performing the following steps:
    a. Click the Packs tab. Initially, the Software Pack ARM::CMSIS is installed by default.
    b. Select Keil::Kinetis_K60_DFP and click Install. The latest Pack downloads and installs to C:\Keil_v5\ARM\Pack\Keil\Kinetis_K60_DFP\ by default. This download can take two to four minutes. The status updates and indicates “Up-to-date”.
7. Repeat the above process for the Kinetis K60 Software Development Kit (SDK_DFP).

Locate & Install the IoT Operational Software

The following assumes you are working with the Avnet AT&T Cellular IoT Kit software.

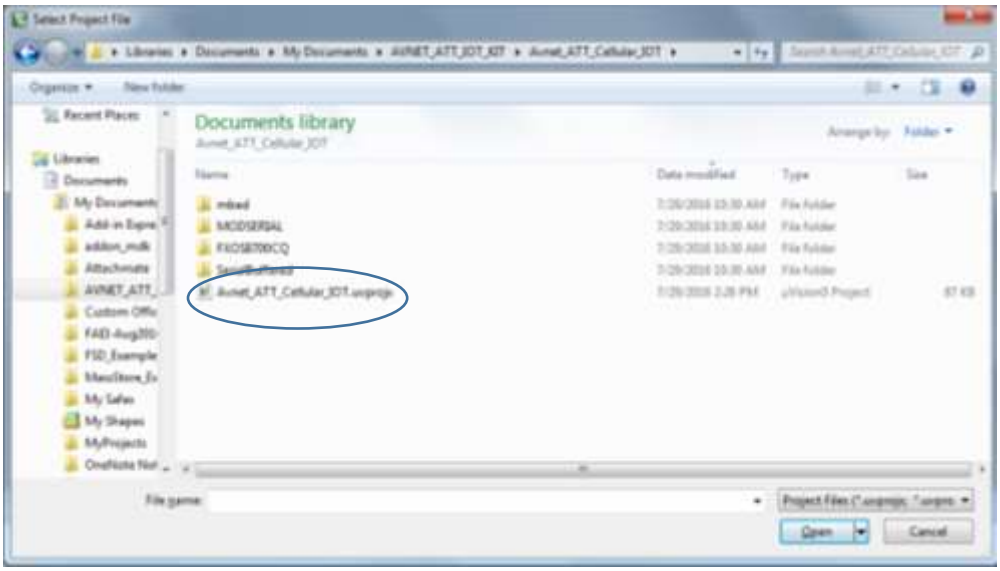
1. Export from developer.mbed.org.
2. Open your browser and navigate to https://developer.mbed.org/users/JMF/code/Avnet_ATT_Cellular_IOT/. The page displayed will look like the following:



3. On this page, select the Export to desktop IDE button. The area shown in the circle appears.
4. Select FRDM-K64F and µVision 5.
5. Click the Export button to generate a ZIP file, with the project in it, and automatically download it to your download folder.
6. Using Windows Explorer, open the ZIP file and extract it to your desired location (e.g., Documents folder). Once it is extracted, you are ready to import it into the Keil IDE.

Import into Keil IDE

1. Open the Keil IDE and choose the **Project** menu item in the menu bar.
2. Select **Open Project.** A Dialog box opens similar to the one below:



3. Navigate to the location where you saved the Avnet_ATT_Cellular_IOT project.
4. Select the Avnet_ATT_Cellular_IOT.uvprojx. You have now imported the project from the mbed.org IDE.

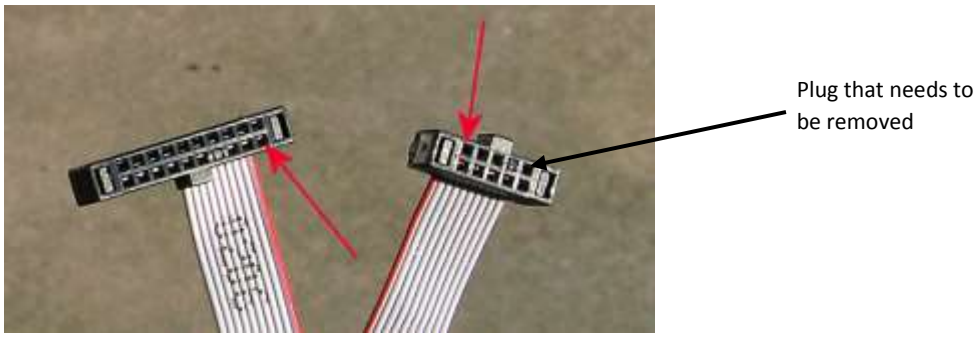
Connecting and Configuring Debug Adapter

The ULINK*pro* supports all SWV features and has ETM Trace support. It also provides Flash programming capabilities.

NOTE: When you use the flash programming capabilities, the FRDM-K64F Flash is erased and the OpenSDA boot loader that was previously in the Flash is gone. If you want to return to using OpenSDA, you must reprogram the OpenSDA bootloader.

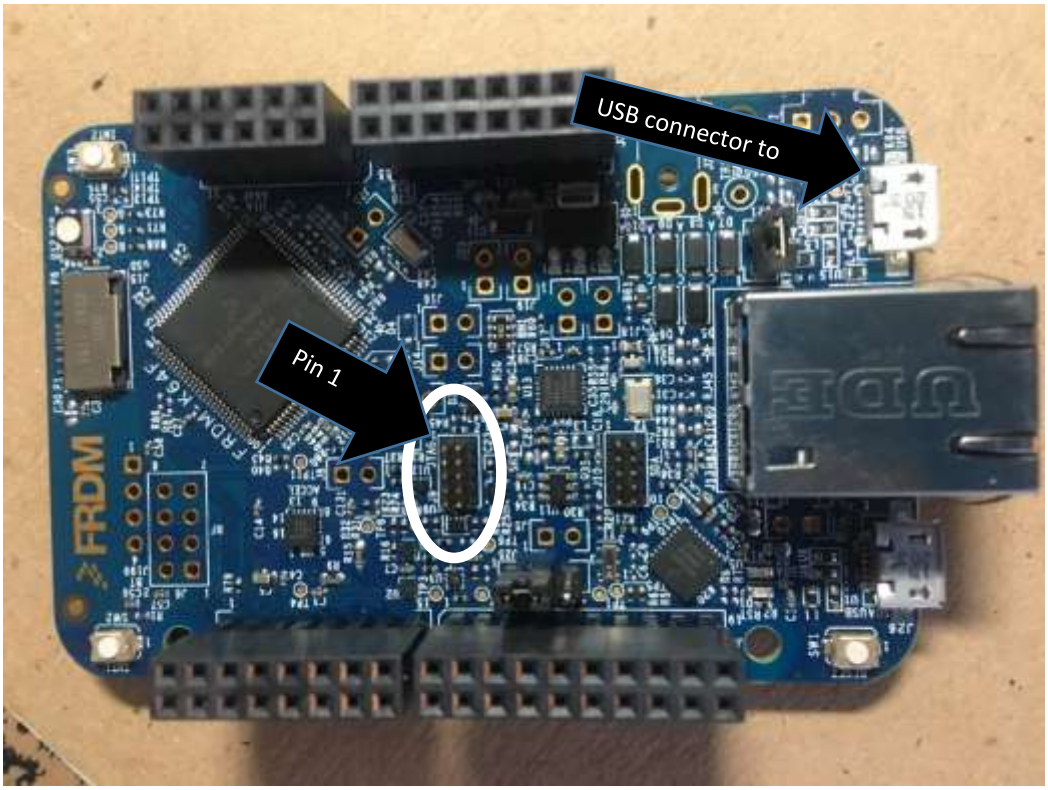
Because you want to use Serial Wire Viewer, it is necessary to install the 20 to 10 pin CoreSight cable to the JTAG unit. This new cable is provided with the ULINK*pro* unit.

The new cable provided with the ULINK*pro* unit may have pin 7 filled with a plastic plug. If so, remove the plastic plug before connecting to the Kinetis target. This is easily done with a sharp needle. Merely pry the plastic pin out.



NOTE: Do not remove or disturb the battery. The ULINK*pro* will not operate if battery power is lost.

With the 20 to 10 pin adapter installed on the ULINK*pro* unit and any pin plugs removed, connect the cable to the **J9-JTAG** connector (shown circled below). Note the location of Pin 1 will correspond to Pin 1 on the connector (the plug is not keyed).

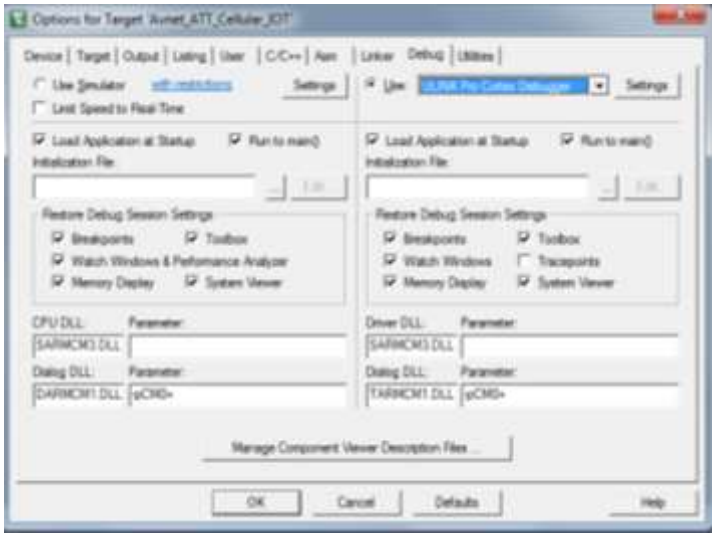


When using OpenSDA, the computer is connected to the SDA-USB port. Ensure this is NOT connected when the ULINK*pro* is used. Instead, insert the USB cable in the USB port (as shown above). This provides power to the K64F board. Once the JTAG unit is connected to the FRDM-K64F board, install the WNC-Shield board and make the necessary connections (e.g., antenna, power).

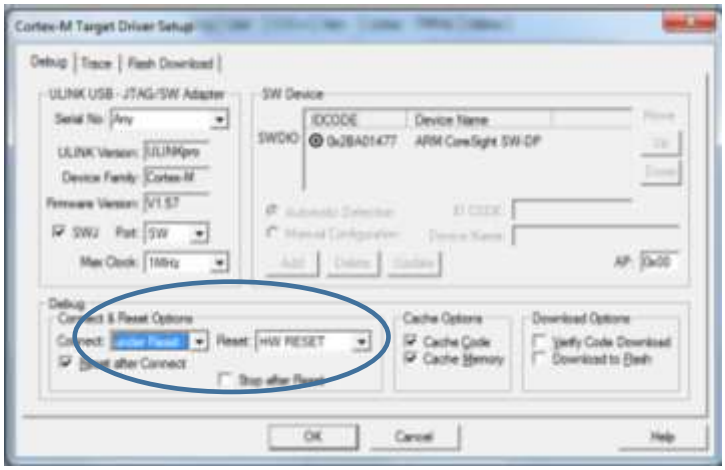
Configuring µVision v5

With the board assembled and connected to the ULINK*pro* JTAG unit, perform the tool configuration tasks. These tasks are all performed from the **Options** dialog. This dialog box is found under the **Project** menu or button. 

1. Enter the **Debug** tab. From the Options tab, modify the “Debug”, “Linker”, and “C/C++” tabs.

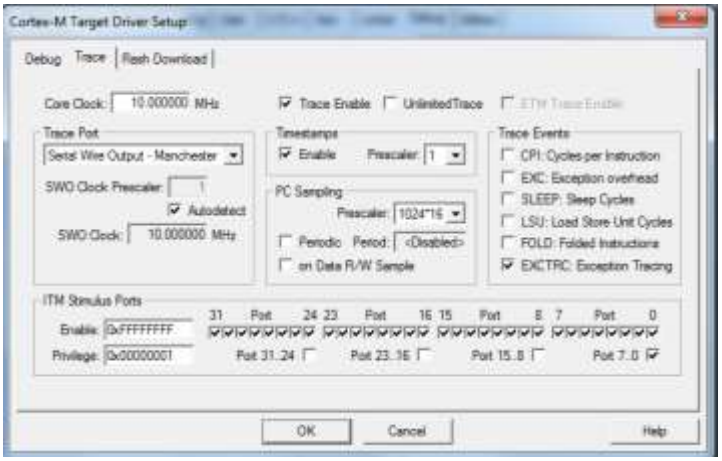


2. From the **Use** selector, choose “ULINK*pro* Cortex Debugger” (highlighted). The software connects to the ULINK*pro* debugger.
3. Select the **Settings** button to select specific features. The dialog opens as follows:

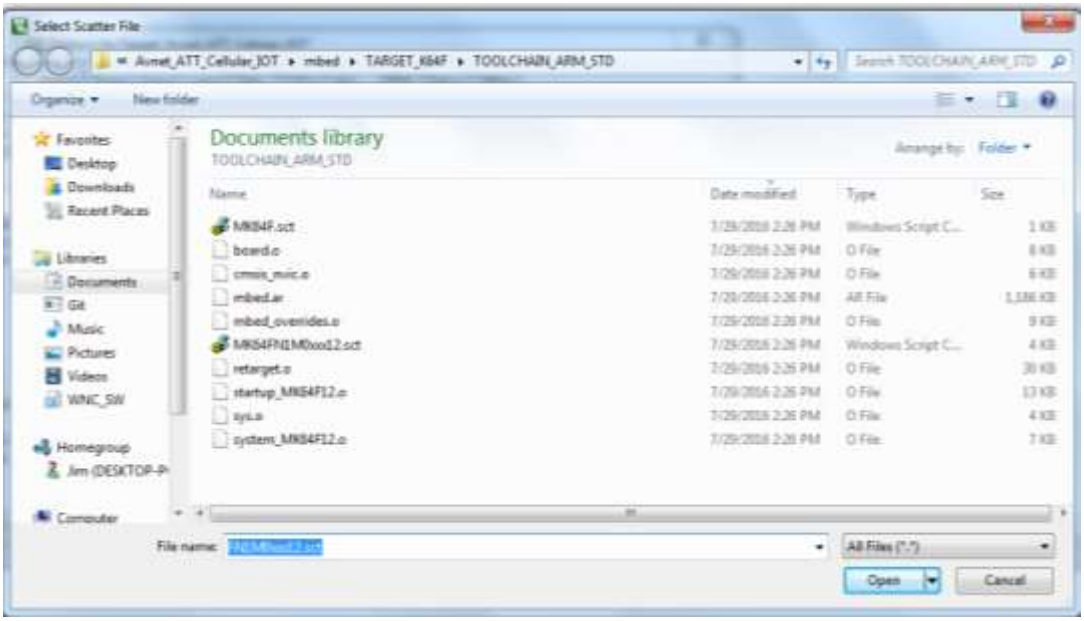


4. Ensure the **Connect & Reset Options** are set as indicated above. That is, SWJ is selected and **Port:** is set to SW.

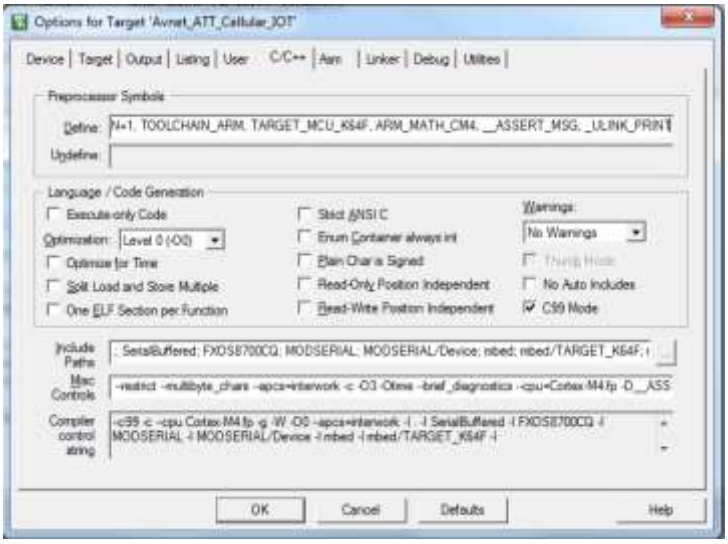
5. Move to the **Trace** tab to set up the trace configuration.



6. Ensure **Port 7..0 and Pin 0** is selected so information can be output to the **Printf (debug)** window. Also verify that **Serial Wire Output – Manchester** is selected. uVision uses Port 0 and 31. All other ports are not used at this time.
   NOTE: The Core Clock should be set to 120.0 MHz to match the operating environment to ensure the timing values in various trace windows are correct.
7. Return to the Options dialog and select the **Linker** tab. You need to select a different scatter file for this project.
8. In the **Scatter File** box, use file picker to open a file navigator dialog.
9. Go to the .\mbed\TARGET_K64F\TOOLCHAIN_ARM_STD\ directory and select the MK64FN1M0xxx12.sct file as shown:

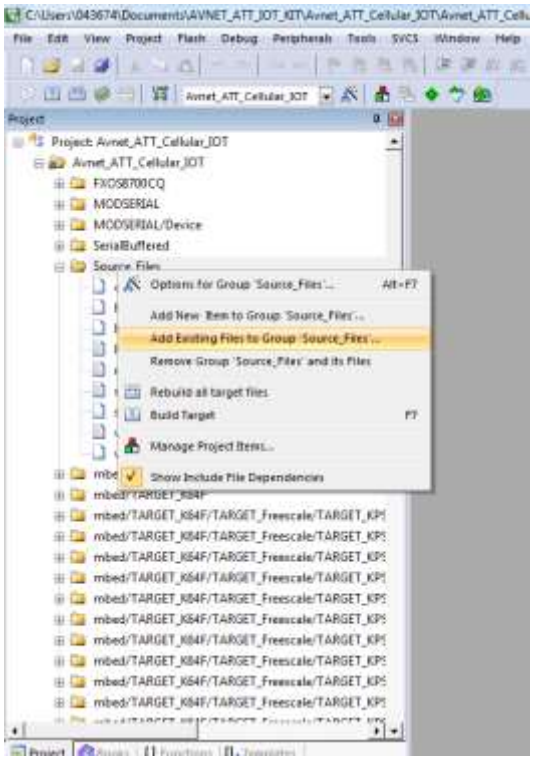


10. From the Options dialog, select the C/C++ tab. At this point, you will add a new define to the project call **_ULINK_PRINT**. This is added to the end of the **Preprocessor Symbols** Define filed as shown:



Updating the Project Files

Because debug/information output is sent to the **Debug (printf) viewer** when using the ULINK, you need to include new files into the project that will accomplish this.



2

These files redefine two macros, PRINTF and PUTS, which are implemented differently for either OpenSDA or ULINK.

To add these two files go to the **Project** window, right click the **Source Files** selection, and select **Add Existing File to Group.** From within the dialog box that opens, select **itm_output.cpp**

Modifying IoT Software for ULINK*pro*

Source Code Changes

The code changes required to use the ULINK*pro* are minimal and involve redirecting all *printf* and *puts* output to the ITM SWV output. This is accomplished in the project by modifying the following files:

config_me.h: By adding the **_ULINK_PRINT** print define in the "C/C++" tab, you enable the automatic inclusion of *itm_output.h* file. This file contains the ITM addresses and functions needed to output to the Debug (printf) viewer. If **_ULINK_PRINT** is not selected, it defaults to the standard *printf* and *puts* functions in the C runtime library.

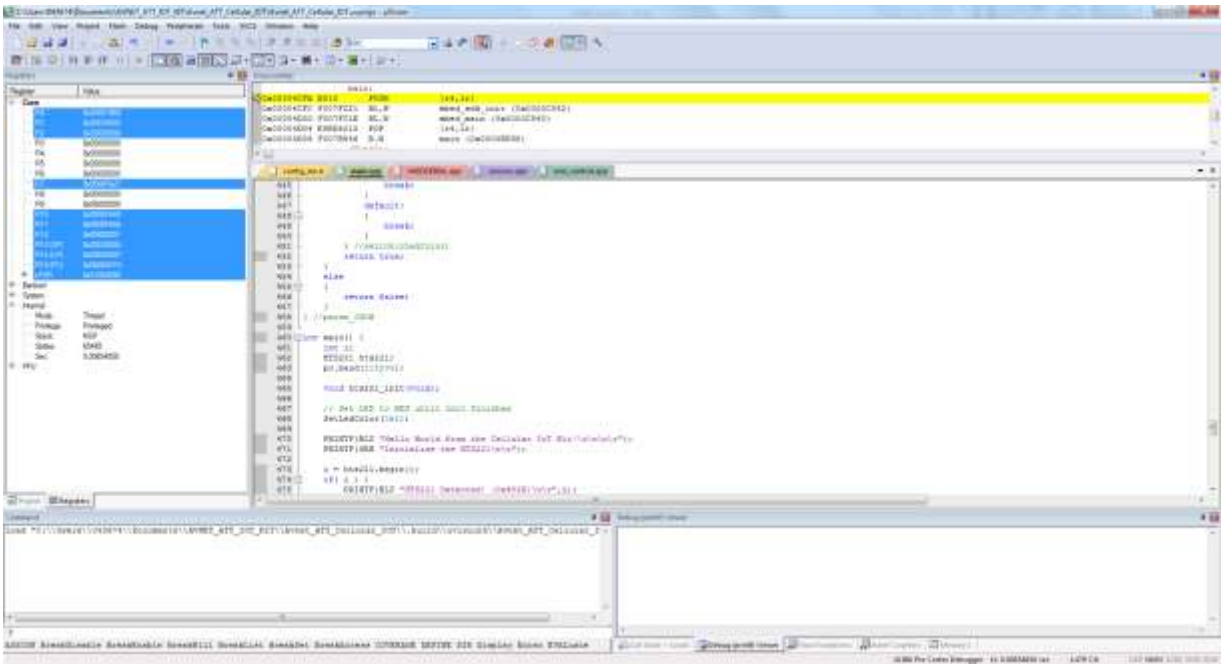
```
#ifdef _ULINK_PRINT
#include "itm_output.h"
#else
#define PRINTF
#define PUTS
#endif
```

main.cpp, Wnc_control.cpp, sensor.cpp: All the changes in these files involve changing the *printf*, *pc.printf,* and *puts* statements to their corresponding pound defines. Lastly, add **config_me.h** to **sensors.cpp**.

Running/Debugging the Program

After all changes are made, build the project so it operates correctly with µVision and the ULINKp*ro.*

1. Press the debug button to have it flash programmed to the FRDM-K64F board and have the debugger start[2]. After the program is flashed into the device and the debugger starts, a screen similar to the following appears:



2. Click the Run (F5) button to run the program. In the **Debug (printf) Viewer**, the debug information appears.

```
Initialize the HTS221

HTS221 Detected! (0xBC)

Temp is: 90.26 F

Humid is: 76 %

Si7020 sensor not found

Si1145 sensor not found

FXOS8700CQ WhoAmI = C7

Modem initializing... will take up to 60 seconds
```

Note: If the SiSensor board is not installed, it will report the sensor not found (as shown).

[2] When entering Debug mode, the K64F FLASH memory will be programmed and the OpenSDA boot loader/debugger erased.

Summary of Steps

When using the mbed.org on-line IDE, use the SDA-USB port for outputting debug/information messages. This is intrusive in that it takes time and cycles to send the data out the USB port. When you move to the Keil/ULINK*pro* environment, you must re-target the debug/information messages.

To retarget the debug/information messages, use ITM port-0, a few ITM specific output functions defined in itm_output.cpp, and redefine the macros PRINTF and PUTS so the output is sent to the Debug Viewer in µVision. Because outputting these messages has less performance penalty when using ULINK*pro*, program timing and execution may be different. These differences can uncover race conditions and other timing issues that would rarely be encountered and thus extremely difficult to debug.

Once recompiled with the output redirected, the remainder of steps involves configuring the µVision/*ULINKpro* tools to operate with the FRMD-K64F board as previously described.

For additional information on using the Keil tools and ULINK*pro* JTAG, refer to the References section.

Useful Definitions/Terms

Instrumentation Trace Macrocell (ITM)
As used by µVision, ITM is thirty-two 32 bit memory addresses (Port 0 through 31) that, when written to, will be output on either the SWO or Trace Port. This is useful for printf type operations. µVision uses Port 0.

Embedded Trace Macrocell (ETM)
Displays all the executed instructions. The ULINK*pro* provides ETM and requires a special 20 pin CoreSight connector. ETM also provides Code Coverage and Performance Analysis.

Embedded Trace Buffer (ETB)
A small amount of internal RAM used as an ETM trace buffer.

Hardware Breakpoints
The Cortex-M4 has six hardware breakpoints. These can be set/unset on-the-fly without stopping the processor. They are no skid; they do not execute the instruction they are set on when a match occurs. The CPU is halted before the instruction is executed.

Watchpoints
The Cortex-M4 has two watchpoints. These are conditional breakpoints. They stop the program when a specified value is read and/or written to a specified address or variable.

References

General Information: www.keil.com/NXP

Books

Getting Started with MDK 5: Obtain this free book at www.keil.com/gsg/

Various books from ARM: www.arm.com/support/resources/arm-books/index.php

Resources at: www.arm.com/products/processors/cortex-m/index.php
Click the Resources tab or select the Cortex-M processor you want in the Processor panel on the left.

Embedded Systems: Introduction to Arm Cortex-M Microcontrollers (3 volumes) by Jonathan Valvano

Application Notes

App Note 287 NXP K64 Freedom: www.keil.com/appnotes/docs/apnt_287.asp

App Note 288 NXP K64 Tower: www.keil.com/appnotes/docs/apnt_288.asp

ARM Compiler Qualification Kit: Compiler Safety Certification: www.keil.com/safety

Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/files/apnt209.pdf

Porting mbed Project to Keil MDK™ 4 www.keil.com/appnotes/docs/apnt_207.asp

MDK-ARM™ Compiler Optimizations www.keil.com/appnotes/docs/apnt_202.asp

GNU tools (GCC) for use with µVision https://launchpad.net/gcc-arm-embedded

RTX CMSIS-RTOS Download https://github.com/ARM-software/CMSIS_5

Barrier Instructions http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html

Cortex Debug Connectors: www.keil.com/coresight/coresight-connectors

Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp

FlexMemory configuration using MDK www.keil.com/appnotes/files/apnt220.pdf

ARMv8-M Architecture Technical Overview https://community.arm.com/docs/DOC-10896

NXP Kinetis: Twr-K60D100M Cortex™-M4 Lab http://www.keil.com/appnotes/files/apnt_284.pdf

Useful ARM Websites

CMSIS Standards: https://github.com/ARM-software/CMSIS_5 and www.keil.com/cmsis

ARM and Keil Community Forums: www.keil.com/forum and http://community.arm.com/groups/tools/content

ARM University Program**:** www.arm.com/university

mbed™: http://mbed.org