Arm® Cortex®-M
32-bit Microcontroller

Nu-Link Keil® Driver User Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

List of Figures

1 Overview

Nuvoton Nu-Link Keil® driver is a plug-in of Keil® µVision IDE used to debug Nuvoton chips via Nu-Link. For example, to start and stop program execution, set breakpoints, access memory and erase/program/verify Flash.
This document introduces how to install and use Nuvoton Cortex®-M Keil® plug-in driver.

1.1 Features

Nu-Link Keil® driver supports the following features. Some functions are triggered by µVision. The usages of these functions can be found in Keil® µVision User Guide.

- Erase/program/verify Nuvoton chips (via Flash algorithm of Nu-Link Keil® driver)
- Easy registers access of Nuvoton chips (via SVD files of Nu-Link Keil® driver)
- Support Hardware/Software/Flash breakpoint
- Support Data breakpoint
- Support various configurations for connection (Reset options, SWD clock, etc.)

1.2 Supported Devices

Download revision history from nuvoton website to see the table of supported devices.

After installing Nuvoton Nu-Link Keil® driver, you can also check supported devices from µVision IDE main window “Help -> Open Books Window -> Supported devices of Nu-Link”, and SVD support status in “Supported SVDs of Nuvoton devices”.

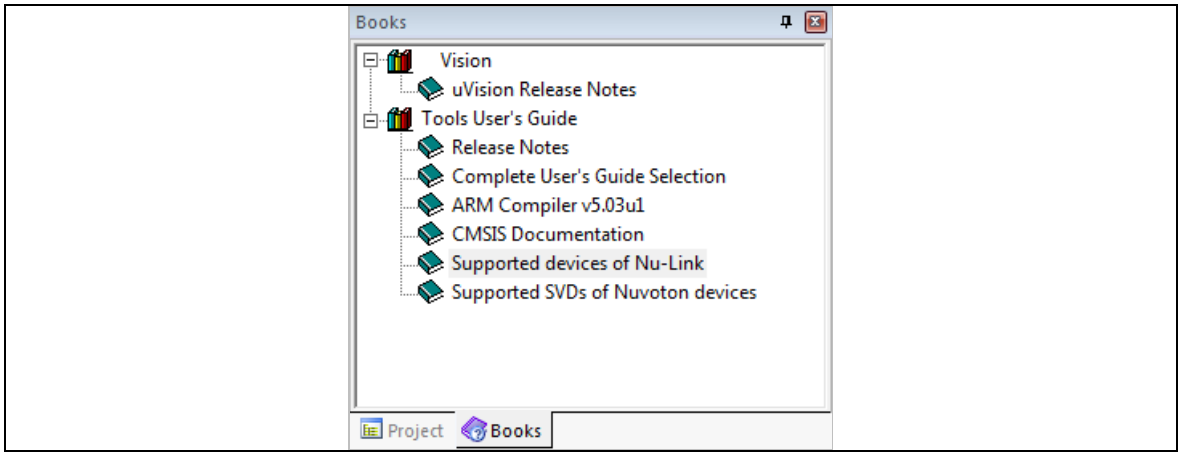

Figure 1-1 Supported Devices List File

2 Installing Nu-Link Keil® Driver

2.1 System Requirements

- **Software:** MDK-Arm® — V3.04 (or later)
- **Hardware:** Nu-Link ICE adapter

2.2 Installation

To use Nu-Link Keil® driver, please follow the steps:

1. Install MDK-Arm® — V3.04(or later) on your PC.
2. Run Nu-Link_Keil_Driver.exe file, and then select the <*Keil® install path*>. The following files can be found after successful installation of package:

- <*Keil® install path*>*Arm**BIN**Nu_Link.dll*: Nu-Link Keil® driver's DLL file.
- <*Keil® install path*>*Arm**NULink*: COPYRIGHT, License, Nu-Link announcement checker and so on.
- <*Keil® install path*>*Arm**Flash*: Flash programming algorithm.
- <*Keil® install path*>*Arm**SFD*: System viewer files.
- <*Keil® install path*> *Arm**NULink**Nuvoton_NuMicro_CortexM_DataBase.cdb* : NuMicro® Cortex®-M MCU database, this file is only for MDK less than v5.0.
- <*Keil® install path*>*Arm**HLP*: Help documents.

After installation, please make sure the Nu-Link Keil® driver version is consistent with the Nu-Link firmware version. If these versions are different, please do Nu-Link firmware upgrade (refer to **chapter 4.2.7**).

3 Nu-Link Keil® Driver Configurations

After selecting “NULink Debugger” as shown in Figure 4-7 and Figure 4-9, you can configure the Nu-Link Keil® driver. This chapter describes the “NULink Debugger” configurations.

3.1 Debug Form

Run µVision, select “**Options for Target – Debug**”, and then click the “**Settings**” button. The debug setting form is shown as follows.

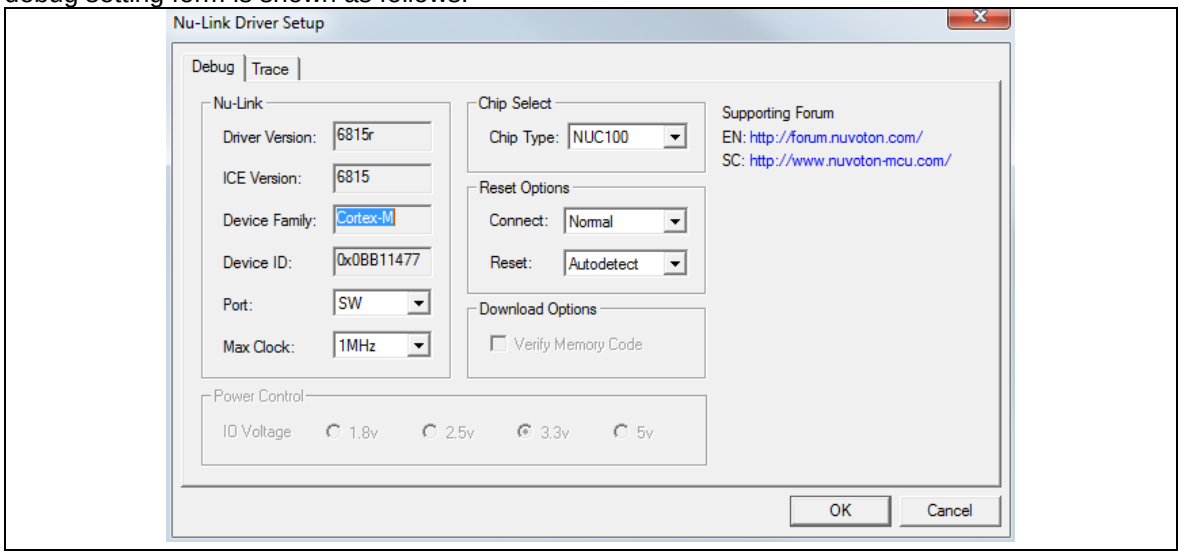

Figure 3-1 Debug Setting Form

- **Driver Version**

Show the latest installed Nu-Link Keil® driver version.

- **ICE Version**

Show the firmware version of the connected Nu-Link ICE.

- **Device Family**

Show device family supported by the driver.

- **Device ID**

Show ID CODE of SWD in a target chip.

- **Port**

Currently, the Nu-Link Keil® driver only supports “SW”, which means SWD port.

- **Max Clock**

Select the maximum SWD clock rate.

- **Chip Type**

Select the target chip type. If user selects the wrong type, the download and debug procedure would be failed.

- **Connect options**

Control the operations that are executed when the debugger connects to the target device.

  - **Normal:** Just stop the CPU at the currently executed instruction after connecting.
  - **under Reset:** Hold the hardware reset (HW RESET) signal active while connecting to the device.

- **Reset options**
  Before getting into Debug mode, Keil® µVision IDE will issue a reset signal to target chip first, there are five reset options as described below:
  - **Auto detect:** Use system reset first; if failed, use hardware reset.
  - **HW RESET:** Use reset pin to reset target chip.
  - **SYSRESETREQ:** Set system reset register of MCU.
  - **VECTRESET:** Set vector reset register. (Cortex®-M0 does not support VECTRESET)
  - **Not Reset:** Not to reset target chip when entering debug mode, which can keep the device status before connected.

- **Verify Memory Code**
  Compare the content of the target memory with the application program loaded in the debugger.

3.2 Flash Programming Form

Run the Keil® µVision IDE, select "**Options for Target – Utilities**", and click the "**Settings**" button. The Flash download form is shown as follows.

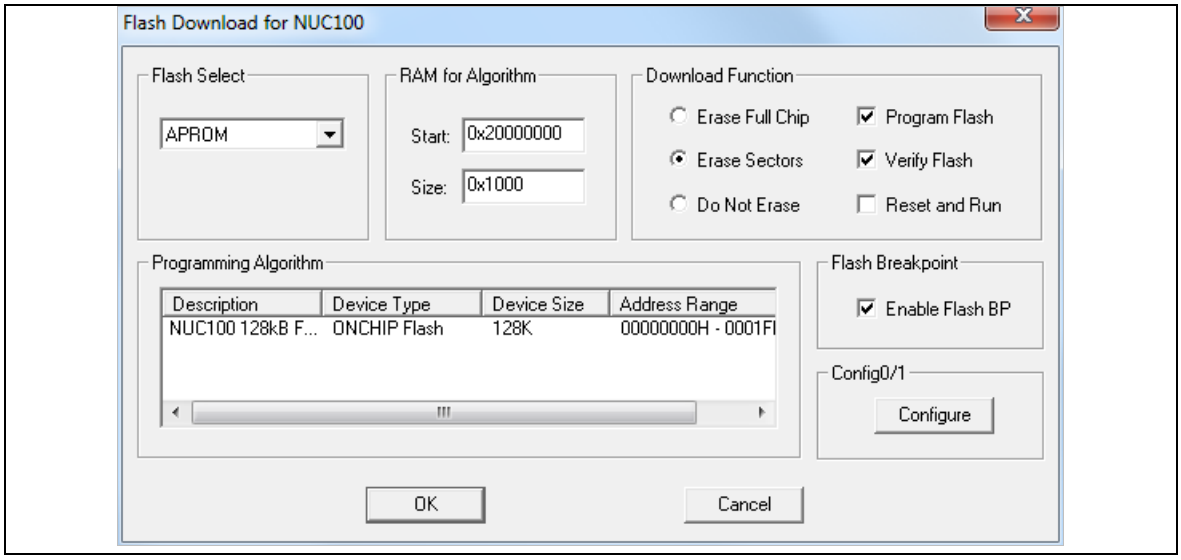

Figure 3-2 Flash Programming Setting Form

- **Flash Select**
  Select the target Flash to download, e.g. APROM or LDROM.

- **RAM for Algorithm**
  Select the Flash algorithm target address and size, programming algorithm will be copied to this address in programming process.

- **Download Function**
  Select active functions, all functions are listed below:
  - **Erase Full Chip:** Erase all Flash data before programming Flash.
  - **Erase Sectors:** Only erase Flash pages those are going to be programmed.
  - **Do Not Erase:** Not to erase Flash before programming it. (Not suggested)
  - **Program Flash:** Program Flash or not.
  - **Verify Flash:** Verify downloaded data or not.
  - **Reset and Run:** When download procedure is completed, IDE will reset target chip; then the target CPU will start to run.

- **Programming Algorithm**
  Display Flash algorithm that will be used in the "download" procedure. If user selects the correct chip type in Debug page, the correct Flash algorithm will be detected automatically.

- **Flash Breakpoint**
  "**Enable Flash BP**" will force Nu-Link Keil® driver to use Flash breakpoints. Otherwise, it will use hardware breakpoints or software breakpoints (When CPU runs on SRAM). Flash breakpoints and software breakpoints are unlimited. Refer to Arm® reference manual for the number of hardware breakpoints (e.g. Cortex®-M0 has 4, and Cortex®-M4 has 6).

- **Update Config0/1**
  Click the "**Configure**" button to open the "**Chip Options**" form. Config Bit can be modified in

the form when a field value is selected and the "**OK**" button is clicked. You can click the "**Cancel**" button to exit the form.

Note: In most NuMicro® chips, it will do "chip erase" before updating config0/1, but in some chips with trust-zone feature (e.g. M2351), only "NS address boundary modify" or "Flash unlock" will cause "chip erase". Thus, users can download image first, and then set their configurations in M2351.

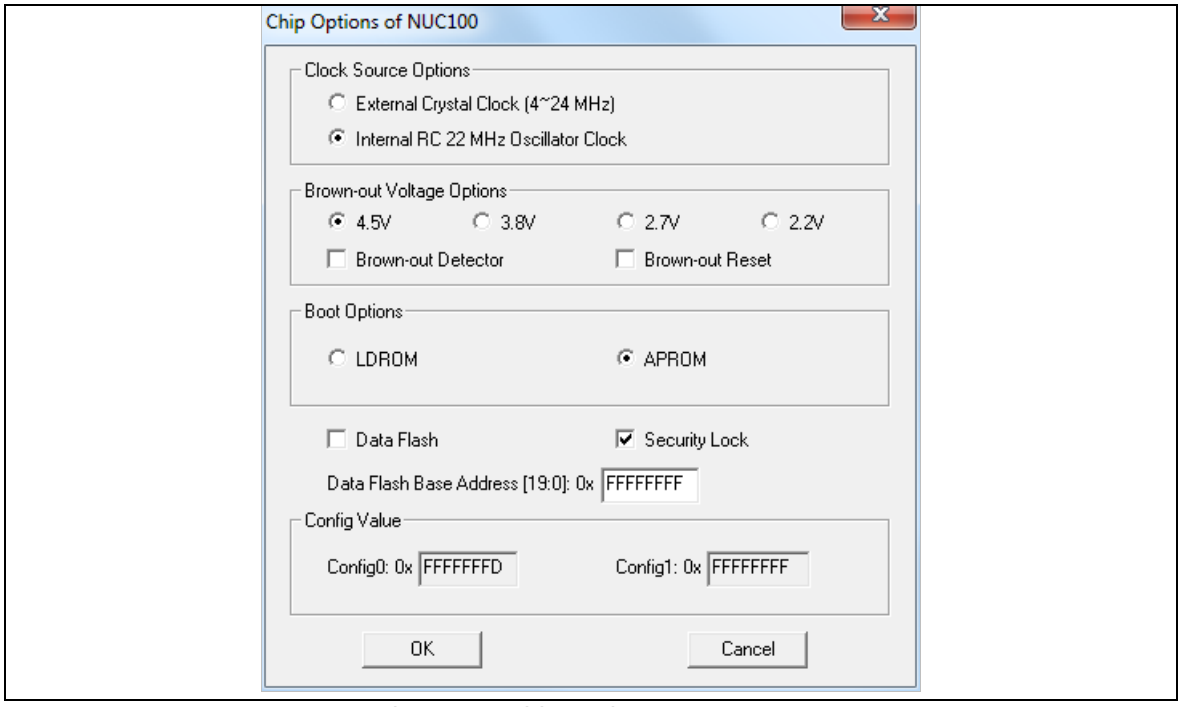

Figure 3-3 Chip Options Form

4 Example – Create & Debug a Project

In general use cases, user can open projects by double clicking project file of Nuvoton BSP.
There are two project file formats – .uvproj for legacy MDK4 and .uvprojx for MDK5.
On the other hand, if you want to create a new project and debug with the Nu-Link Keil® driver, follow the steps below (based on NUC140).

4.1 Create a New Project

1. Run the Keil® µVision IDE. Select "**Project – New Project…**" and the "**Create New Project**" dialog will appear. Input the new project name and click the "**OK**" button. Select "NuMicro Cortex-M Database" which is Nuvoton devices database, as shown below.

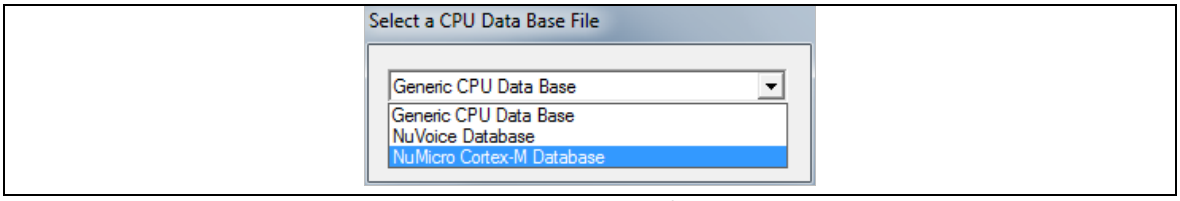

Figure 4-1 Select "NuMicro Cortex®-M Database"

2. Select the microcontroller device that you have, as shown below.

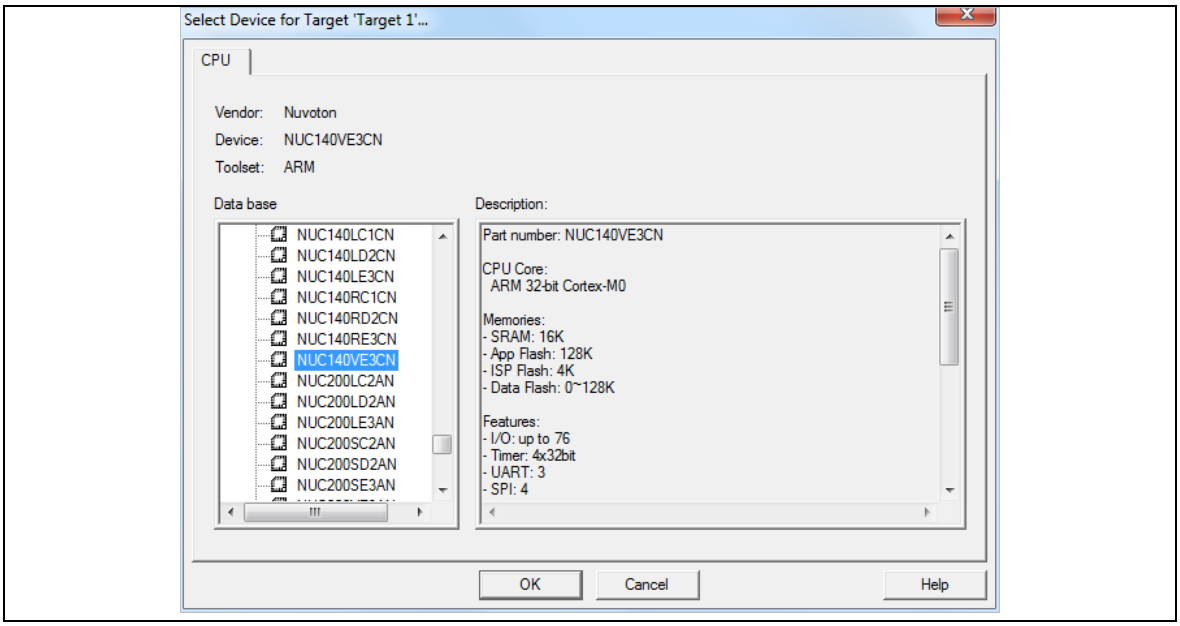

Figure 4-2 Select Device in MDK4

User needs to install "Nuvoton.NuMicro_DFP.pack" before creating the project of .uvprojx format for MDK5. Open *<Keil® install path>\UV4\PackInstaller.exe* as shown in Figure 4-3 and find "Nuvoton.NuMicro_DFP.pack" to install it. After installing the pack, the microcontroller device form is shown as Figure 4-4.

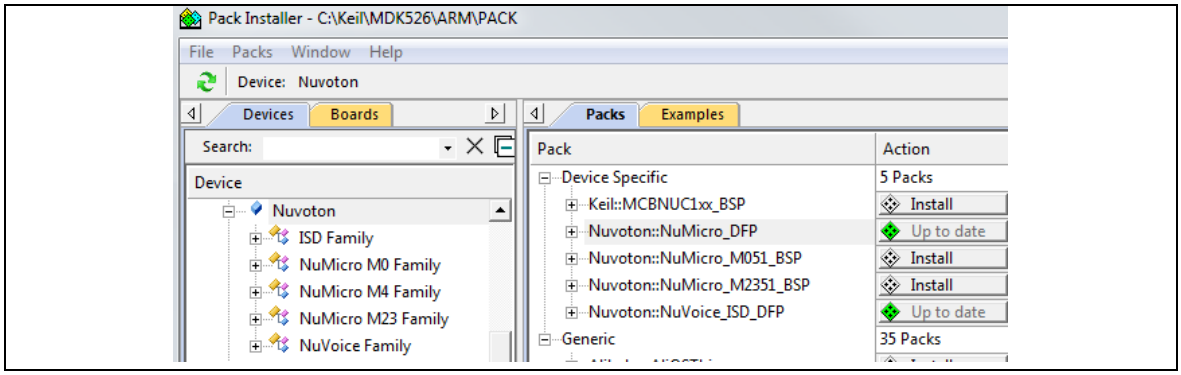

Figure 4-3 Pack Installer in MDK5

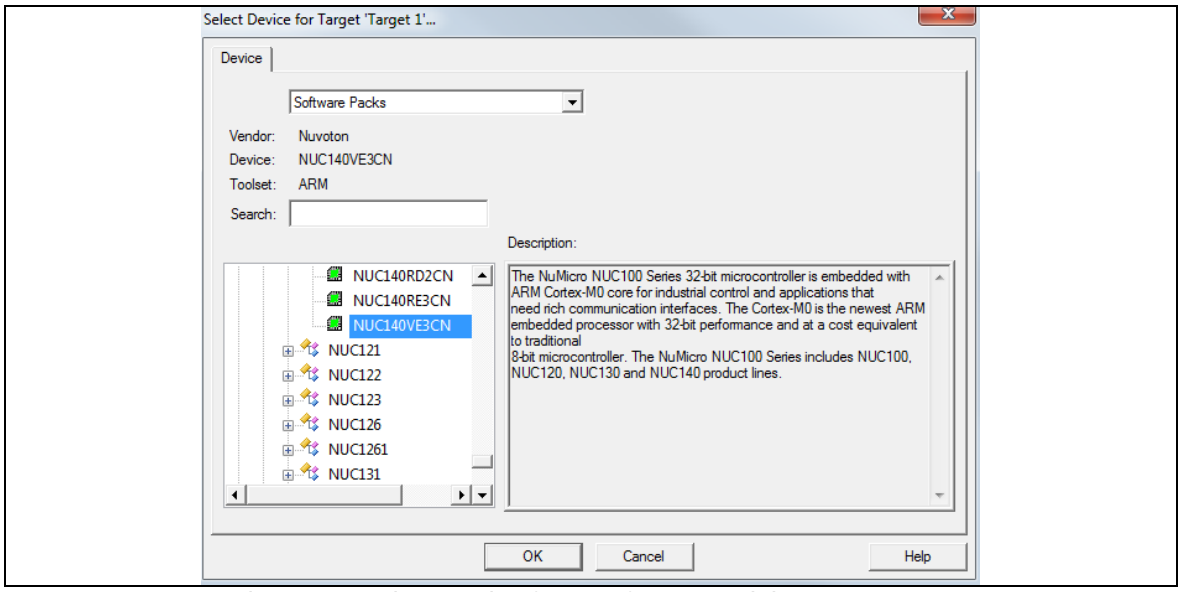

Figure 4-4 Select Device from Software Pack in MDK5

3. If user selects the correct device as shown in Figure 4-2, the RAM/ROM size and *.SFR* file will be filled up automatically. Otherwise, user has to set a correct RAM/ROM size and select correct *.SFR* file from "*<Keil® install path>\Arm\SFD*".

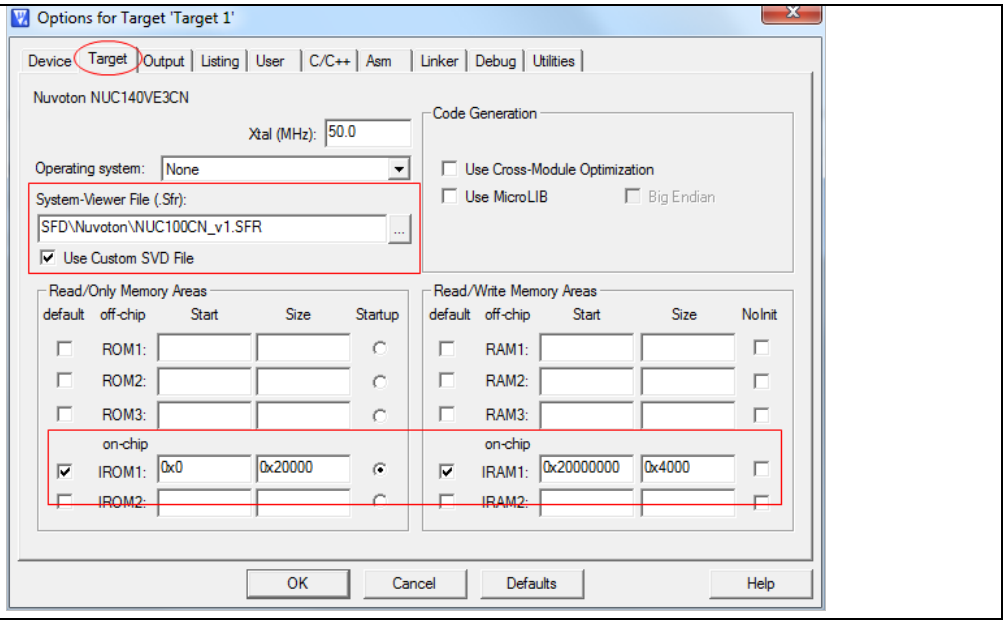

Figure 4-5 Select System Viewer File

4. Enable the "**Thumb Mode**" option.

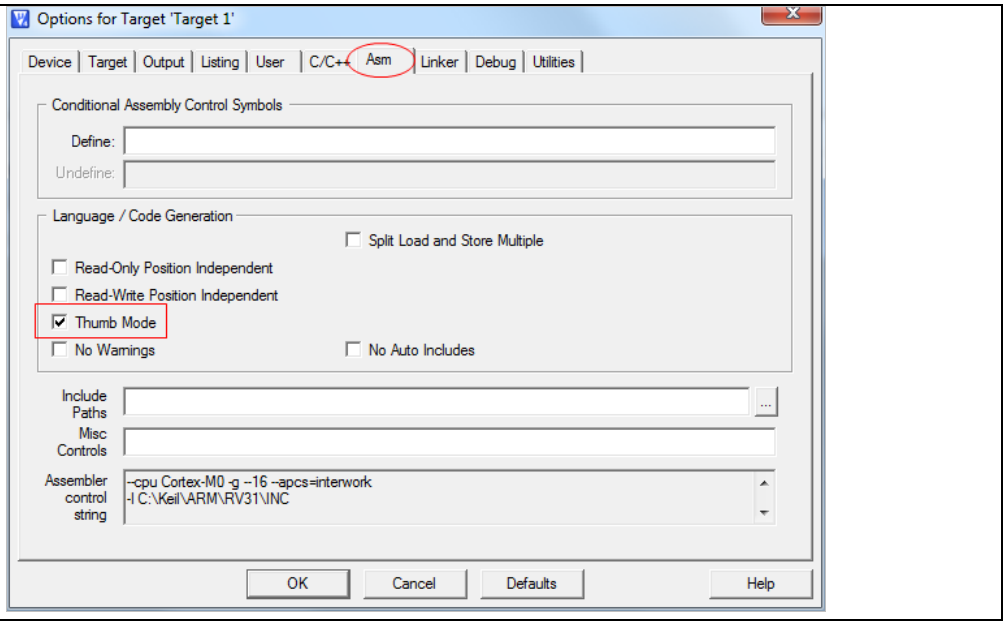

Figure 4-6 Select Thumb Mode

5. Select "**Rebuild all target files**" to build the project.

6. Select "**Options for Target – Debug**". From the combo-box, select "Target ICE MCU" which is the USB Nu-Link Keil® driver. Make sure that the "**Use:**" radio box is checked, as shown below.

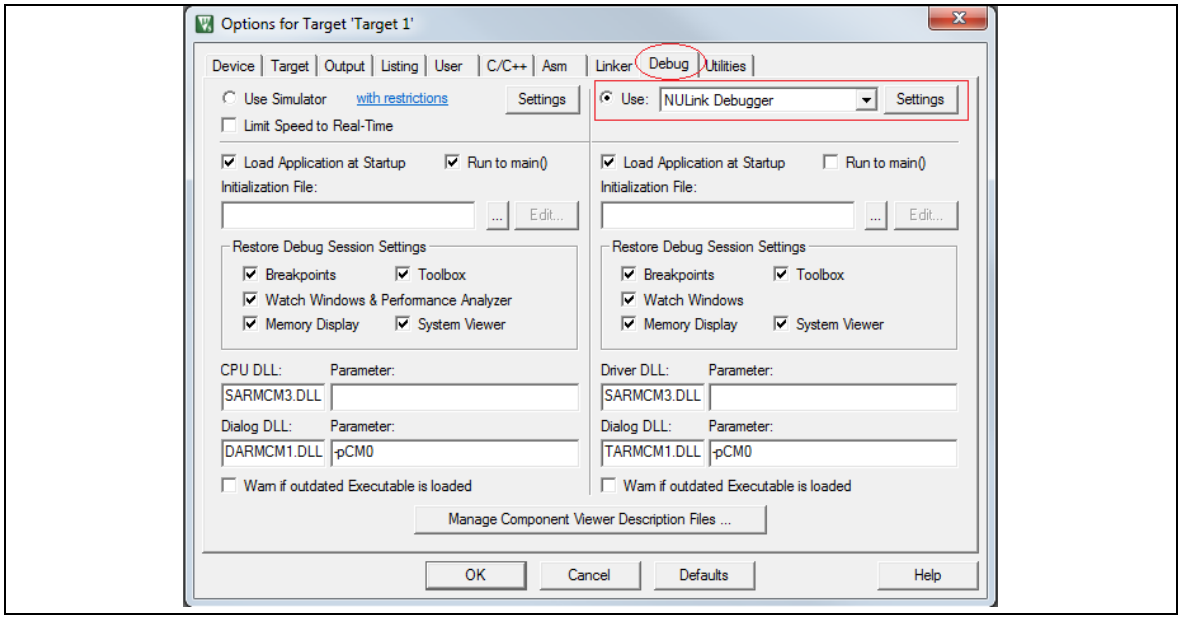

Figure 4-7 Select Debug Driver

7. Click the "**Settings**" button to open the debug setting form, and then select "NUC100" as the Chip Type.

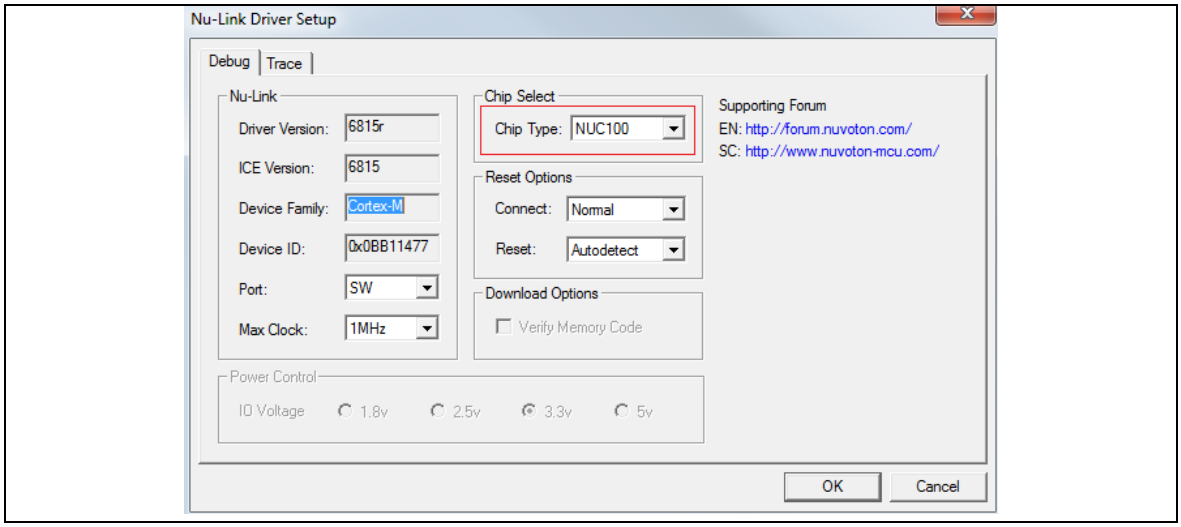

Figure 4-8 Debug Setting Form

8. Select "**Options for Target – Utilities**". From the combo-box, select "Target ICE MCU". Make sure that the "**Use Target Driver for Flash Programming:**" check box is selected, as shown below.

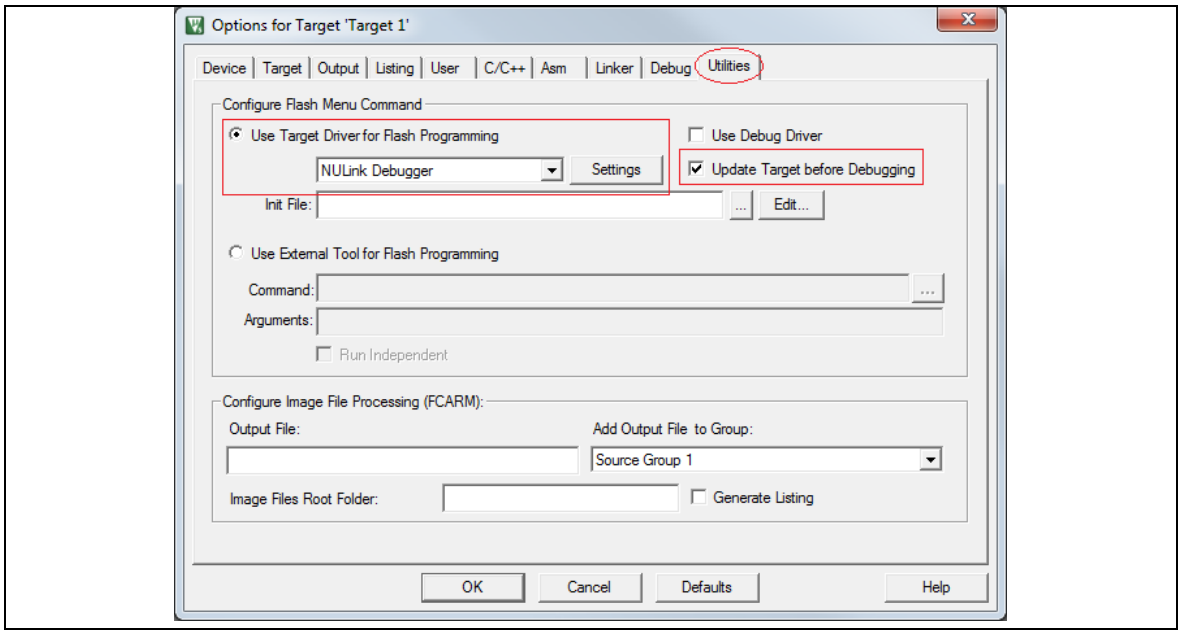

Figure 4-9 Select Flash Programming Driver

9. Click the "**Settings**" button to open the Flash download setting form.

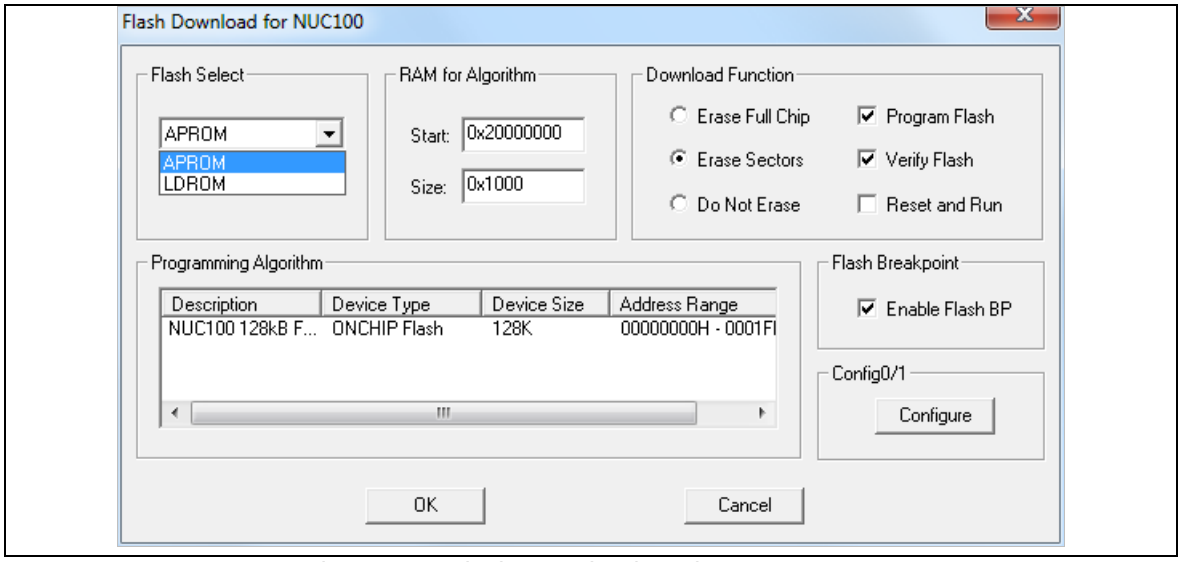

Figure 4-10 Flash Download Setting Form

Note: "**Flash Select**" can download project code to APROM or LDROM or others.

4.2 Debug a Project

This section describes how to debug a project by the Nu-Link Keil® driver (assuming that the installation and configuration steps had been performed).
Please follow the steps below:

- Run the Keil® µVision IDE. Select "**Project – Open Project....**" and the "Open Project" dialog pops up. Choose certain project as follows.

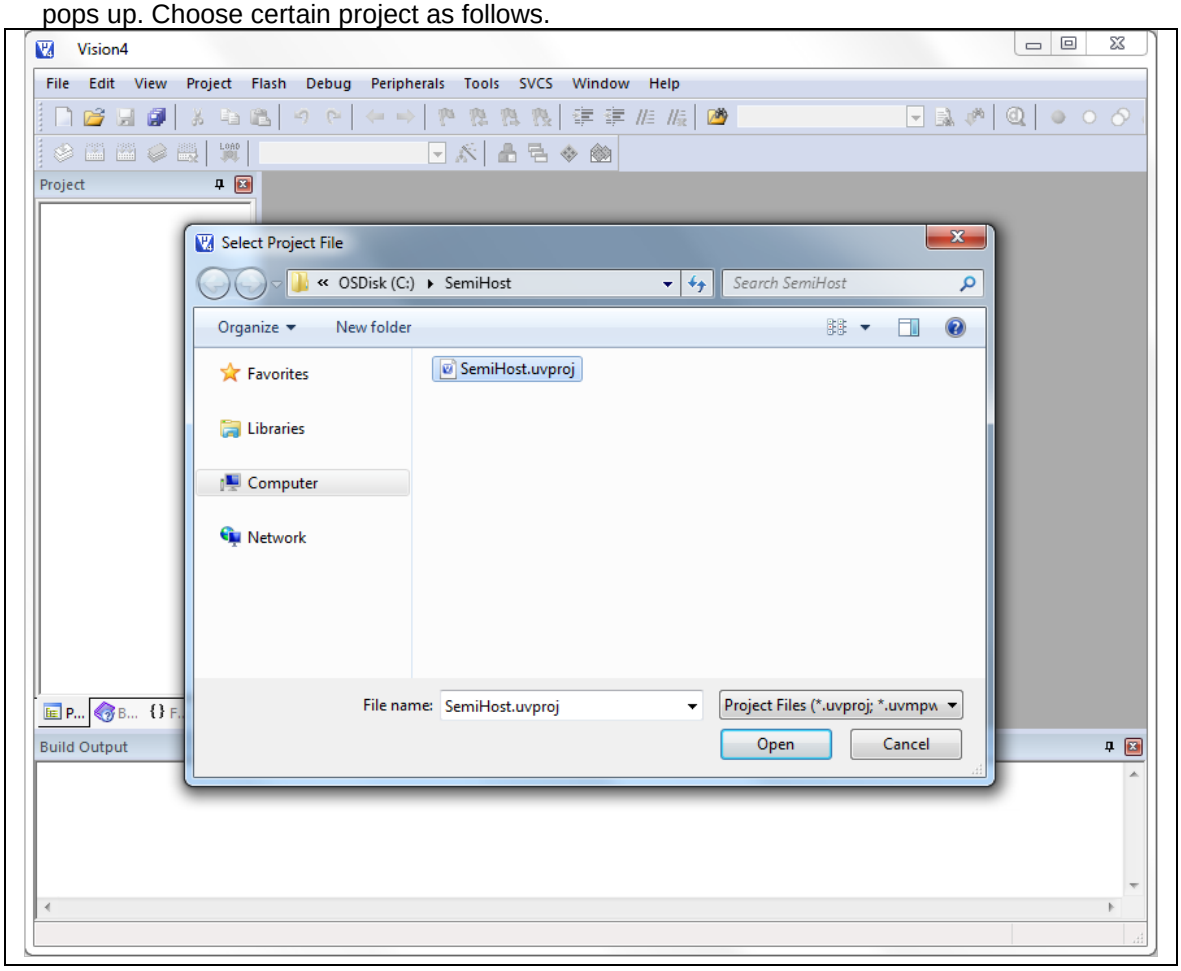

Figure 4-11 Open Project

- Select "**Project - Build target**" to build the project. The build window shown below should report no errors.

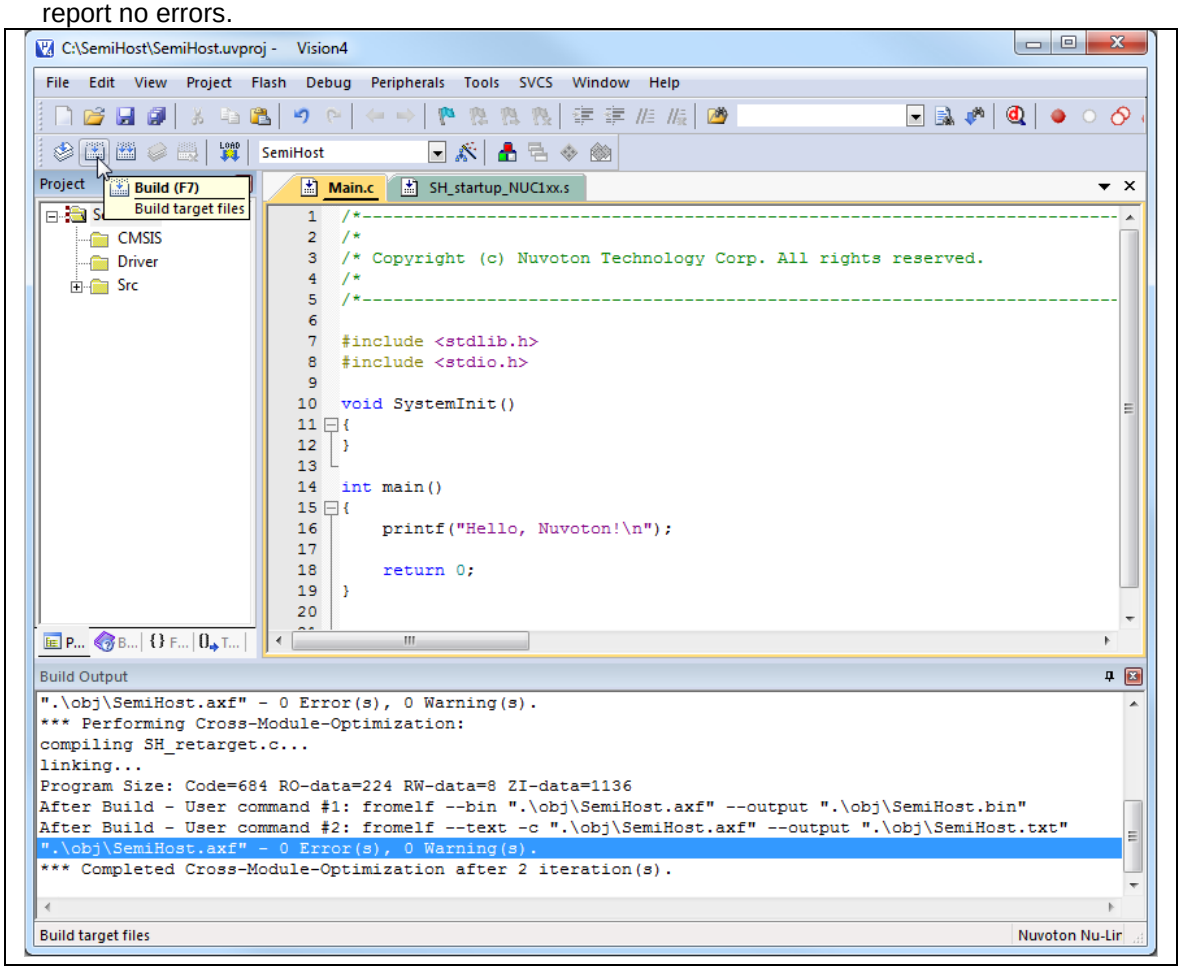

Figure 4-12 Build the Project

- Click the "**Flash - Download**" icon to download the program into Flash.

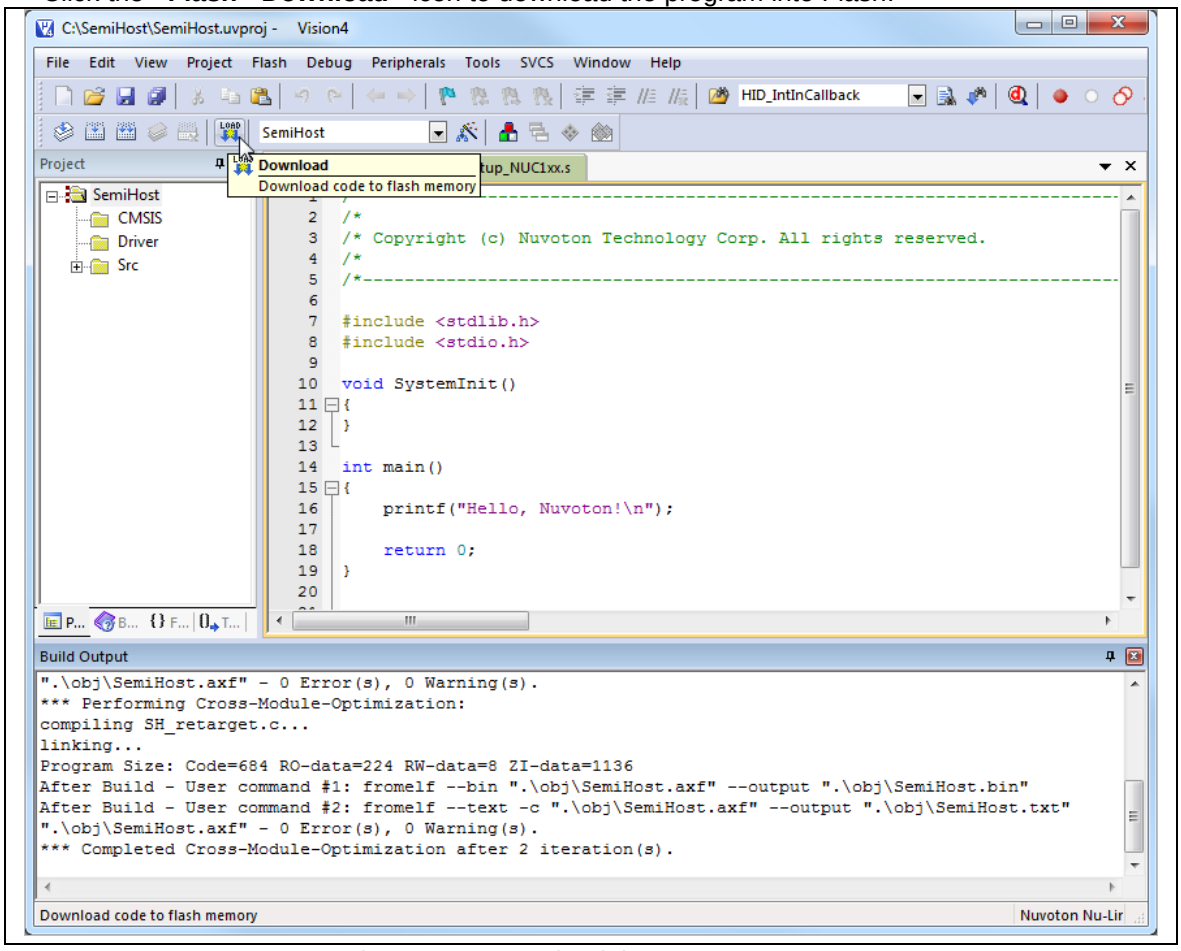

Figure 4-13 Download the Program

- Start to debug by selecting "**Start/Stop Debug session**". When the hardware is properly configured, and the program successfully downloaded, the debugger window should look like Figure 4-14.

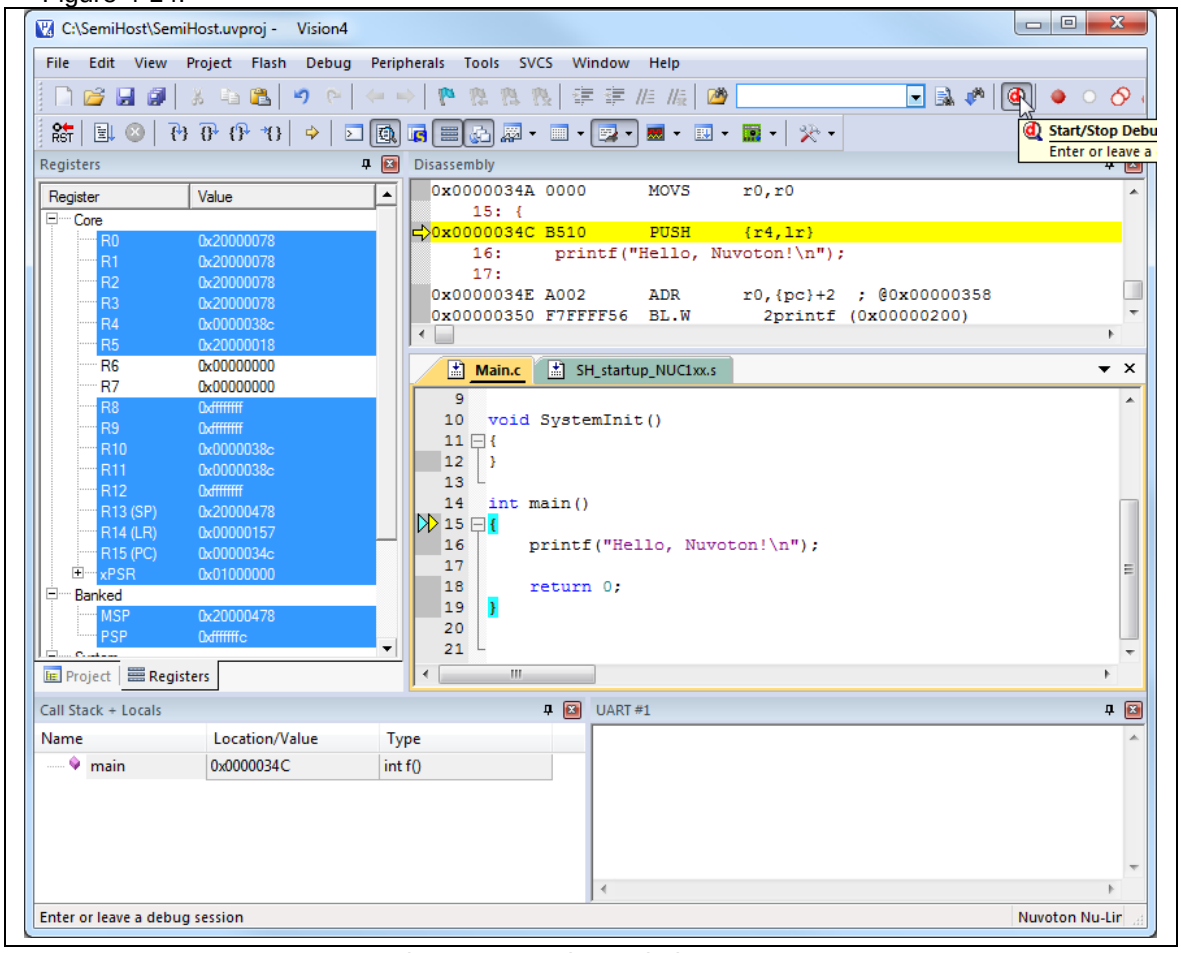

Figure 4-14 Debug Window

- At this point, the program can be run, halted, run step by step, breakpoints can be set/cleared, and variables can be watched, memory areas read/written/modified.

4.2.1 System Viewer

In Debug mode, if user configures system-viewer as shown in Figure 4-5 correctly, the system viewer pane can be used by clicking "**Peripheral**" → "**System viewer**" in the menu bar, which can help developers to configure registers. The "System Viewer" is briefly described as follows.

- Select a registers group such as "GCR", and the registers group will be displayed in the right pane of IDE, as shown below.

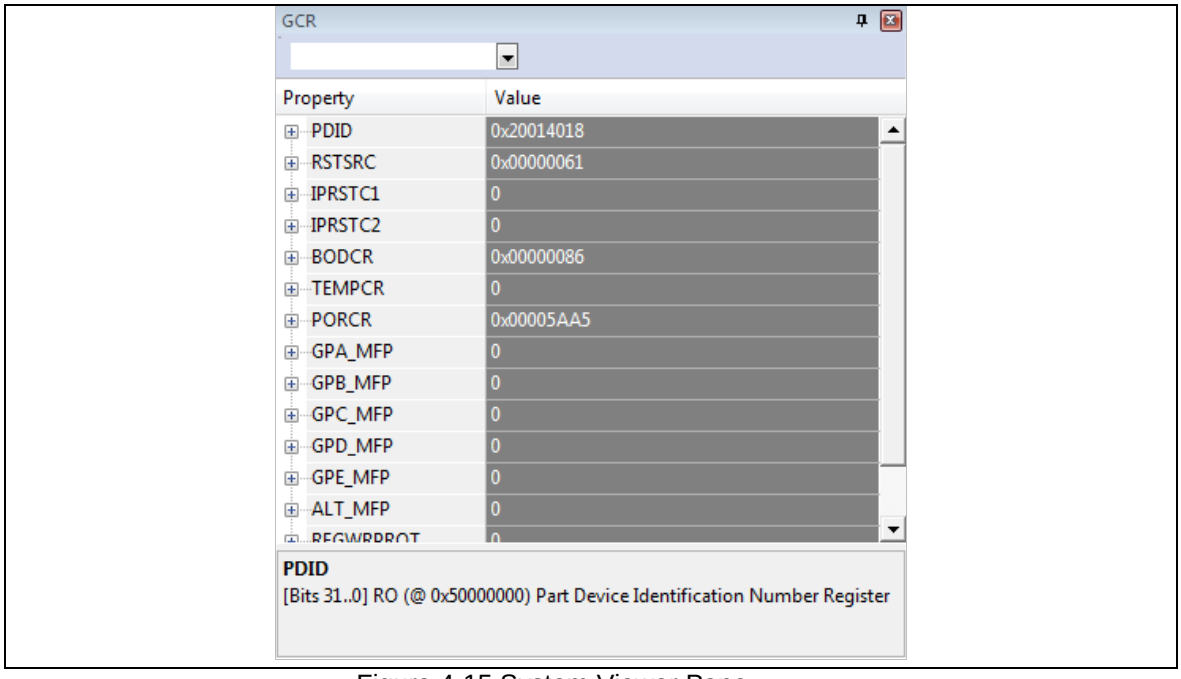

Figure 4-15 System Viewer Pane

- To know detailed information of the registers group, you can expand it by clicking the "**+**" button as shown below.

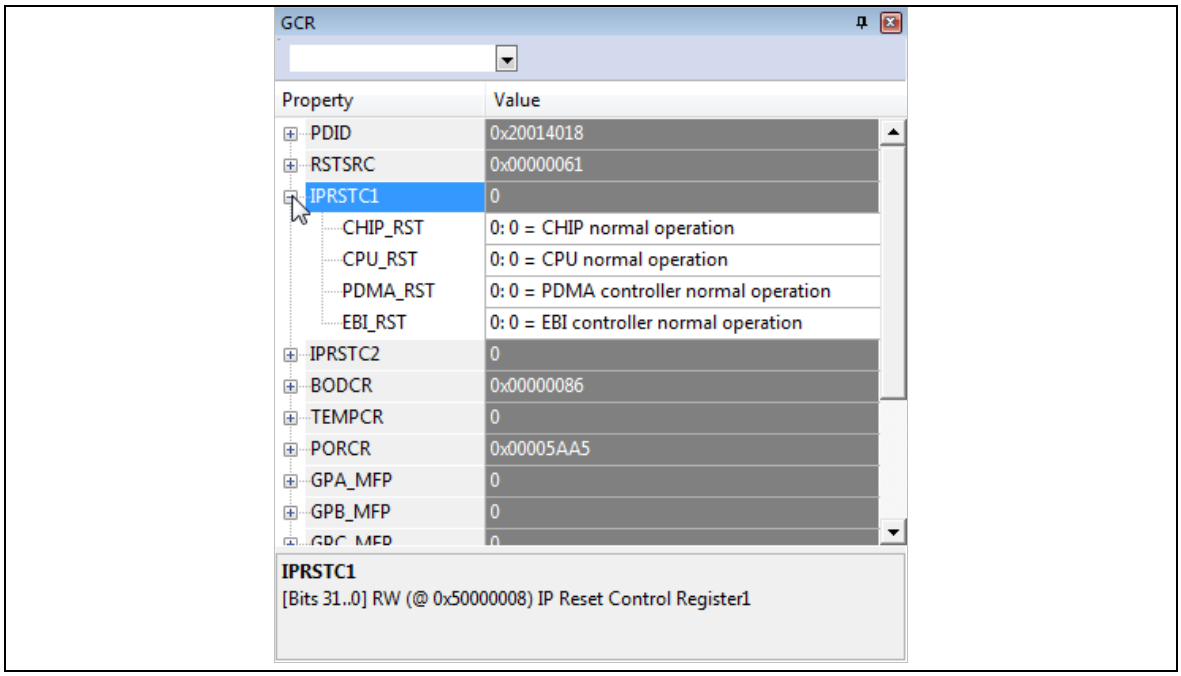

Figure 4-16 Expand Register Group

- To modify the value of register, users can edit the value field, and the detailed register description will be shown at the bottom:

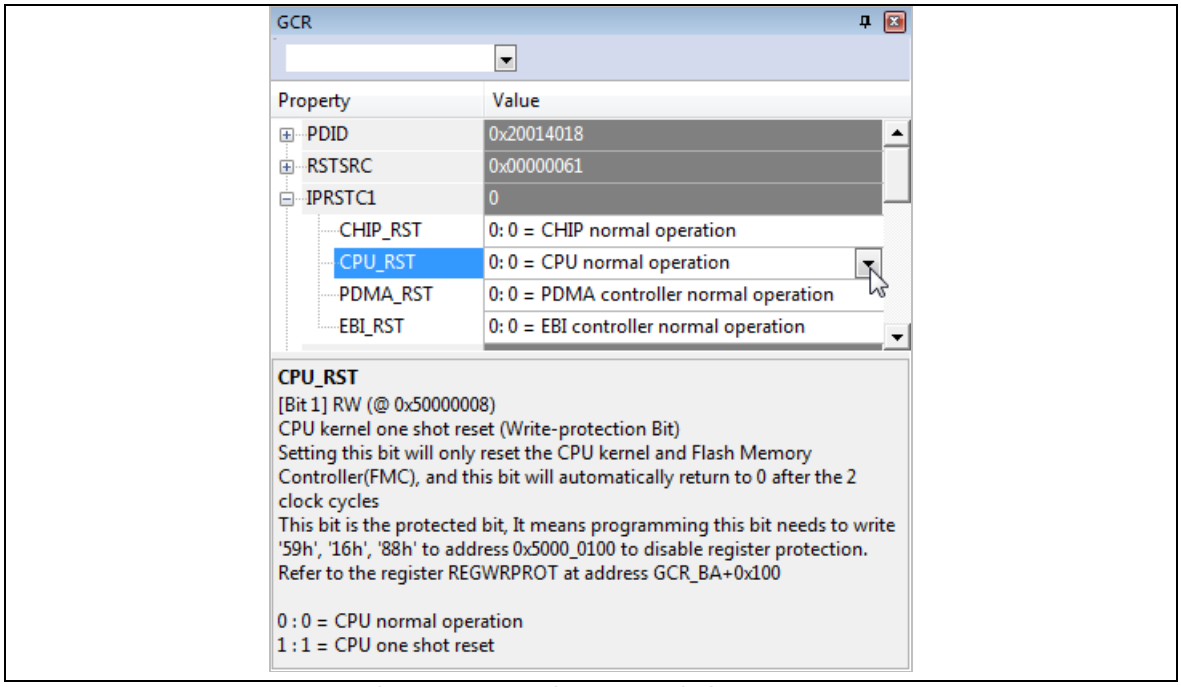

Figure 4-17 Register Description

4.2.2 **Breakpoints**

The Nu-Link Keil® driver supports “Execution Break” and “Access Break” of µVision IDE. You can check the µVision user guide for details: **µVision User Guide → Debugging → Debug Windows** and **Dialogs → Breakpoints Window**).

“Execution Break” can be classified into three kinds of breakpoints: software/hardware/Flash. Each of the breakpoints is described below.

- Software Breakpoints
  When the breakpoint is located in SRAM area, Nu-Link Keil® driver will use software breakpoints, the number of breakpoints is unlimited.

- Hardware Breakpoints
  When the breakpoint is located in FLASH area, Nu-Link Keil® driver will use hardware breakpoints, these hardware breakpoints are provided by Arm® Cortex®-M core, the numbers are limited. (Cortex®-M0 has 4, Cortex®-M4 has 6)

- Flash Breakpoints
  When the breakpoint is located in FLASH area, and hardware breakpoints are all used, Nu-Link Keil® driver will use Flash breakpoints. (Note that “**Enable Flash BP**” in Figure 3-2 must be enabled)
  Flash breakpoints are unlimited, but using Flash breakpoint will make program execution much slower than using hardware breakpoints.

  “Access Break” is also known as data breakpoints.

- Data Breakpoints
  No matter target application runs on SRAM or Flash, users can set data breakpoints which are provided by Arm® Cortex®-M core. The number of data breakpoints are limited (e.g. Cortex®-M0 has 2, and Cortex®-M4 has 4).

4.2.3 PinView Plug-in

In Debug mode, user can select “NuTool – PinView” from “**Debug**” → “**NuTool – PinView**”, and then check the correctness of pin assignment through GUI.

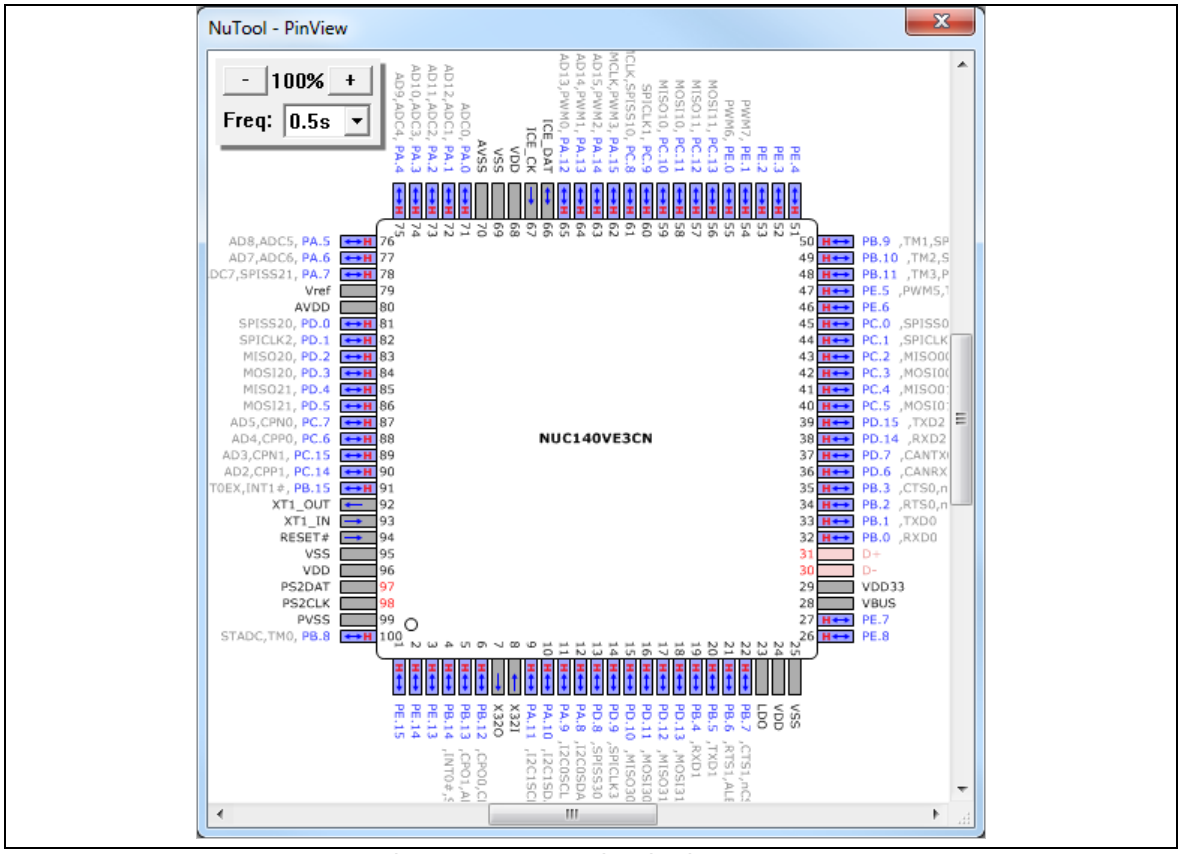

Figure 4-18 NuTool - PinView

4.2.4 Semihosting

The Nu-Link supports some semihosting function for user to easily output and input message in Keil® µVision IDE. To use the semihosting functions, follow the steps below.

1. Open your project, add #define DEBUG_ENABLE_SEMIHOST.

2. Build and run.

3. In Debug mode, select “**View → Serial Windows → UART #1**” and click UART #1 to and open the UART #1 pane. The semihosting input/output will use this pane.

4. Now when you execute the program and run printf(“%s”, string), you can see debug information in the UART #1 pane as shown below.

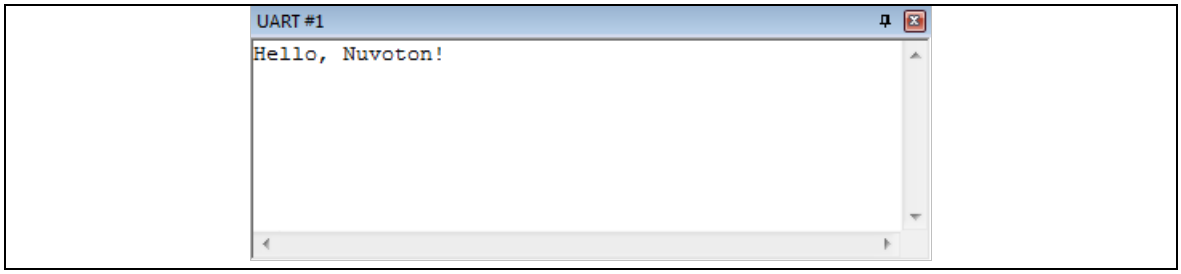

Figure 4-19 Debug Information in UART Pane

4.2.5 NuConsole Plug-in

In addition to semihosting, the Nu-Link provides another I/O mechanism without affecting the target's real time behavior. By using standard debug port SWD, It doesn't need any additional pin or hardware. The only requirement is that the target application should reserve a buffer space (hereinafter referred to as **InfoBlock**) in SRAM in order to store control settings and communicate I/O data between NuConsole and target. To use the NuConsole functions, please follow the steps below:

1. Open "*C:\Keil_install_folder\NULink\NuConsole_Sample*" directory and copy the following files into your project folder:
   - *NuConsole.h/.c*
   - *NuConsole_Config.h*
   - *NuConsole_Retarget.c*

2. Set up the project
   - Replace *retarget.c* with *NuConsole_Retarget.c.*
   - Add *NuConsole.c* to the project and include *NuConsole.h* in the corresponding files.

3. Configuare InfoBlock
   - *In NuConsole_Config.h*, adjust the appropriate size of TX/RX buffers according to the application requirements and hardware limitation. Also, the TX buffer can be configured to be blocking or non-blocking. (optional)
   - Call **NuConnsole_Init()** function to initialize InfoBlock before doing I/O operations (e.g. printf()).

4. Build the project
   - Build *the* code.
   - In the *linker* map "*project_name.map*" file under "*project_path/lst*" directory, find the value of symbol **NuConsole_InfoBlock** variable declared in *NuConsole.c* to get the memory address of InfoBlock. (Optional, Nu-Link Keil® driver will do this automatically.)

5. Download and run

6. In Debug mode, select "**Debug → NuConsole**" to invoke the control dialog. Set up the address of InfoBlock (Optional, Nu-Link Keil® driver will do this automatically.), and then click the start button to process I/O data.

7. Now when you execute the program and run the I/O statements, you can see debug information in the NuConsole dialog as shown below.

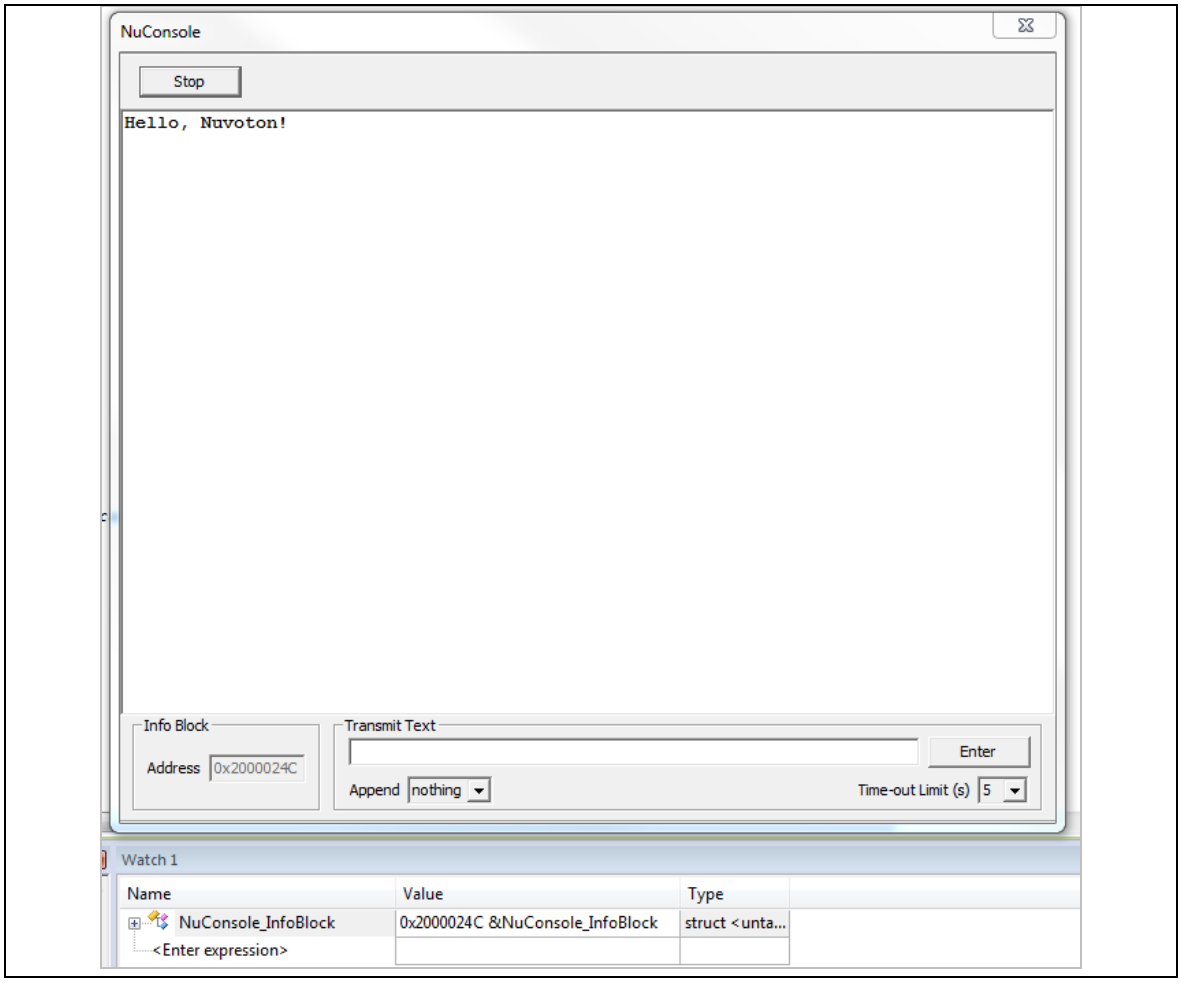

Figure 4-20 Debug Information in NuConsole Dialog Box

4.2.6 ITM/ETM Trace

To start Embedded Trace Macrocell (ETM) tracing on Nuvoton Cortex®-M4/M23 devices, please connect to the device using the Nu-Link2 with 20-pin connector and follow the steps below.

1. Configure the NuTrace.
   - In debug setting dialog, select the "**Trace**" tab.
   - In "**Trace Port**" select **Sync Trace Port with 4 bit data**. It is possible to use other bit sizes but best to use the largest to increase the bandwidth.
   - In "**Capture Mode**", specify whether trace data is collected before or after a trigger.
     - **Trace After:** Capture the trace information after the trigger point and stop capturing when trace buffer is full.
     - **Trace Before:** Capture the most recent trace information before CPU is stopped.
   - **Select Trace Enable** and **ETM Trace Enable.**
   - **Click OK** to save the changes.

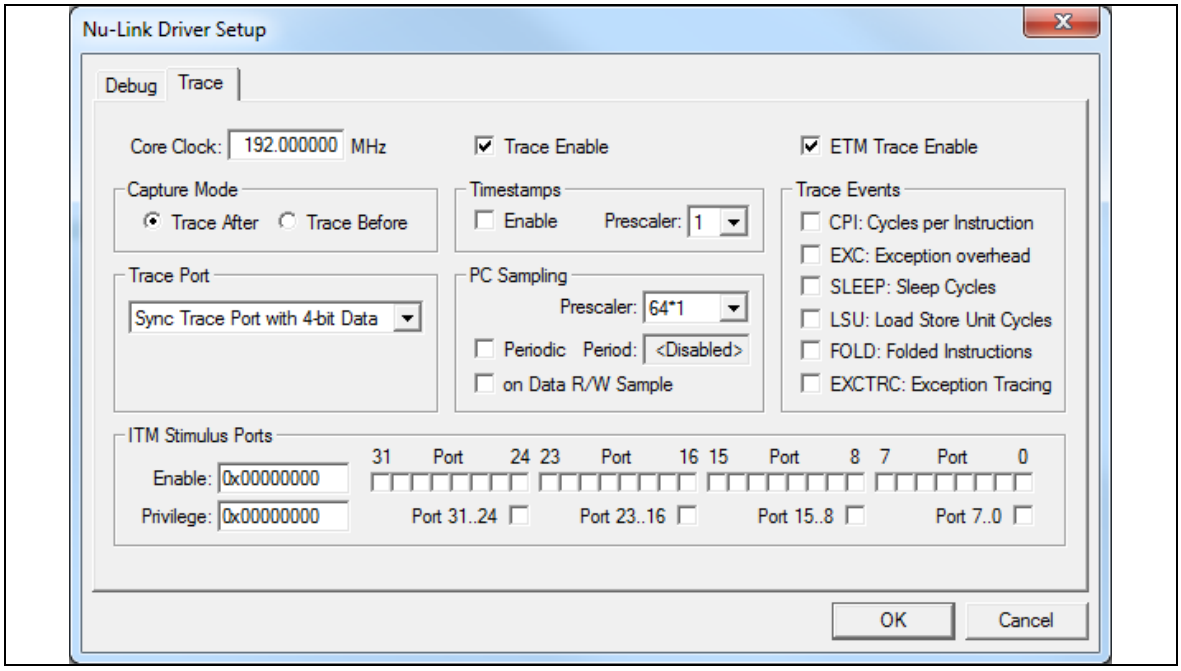

Figure 4-21 Trace Setup with ETM

2. In **Initialization File**, please insert the script file to initialize the device's trace pins when starting the debugger. The following is an example script file.

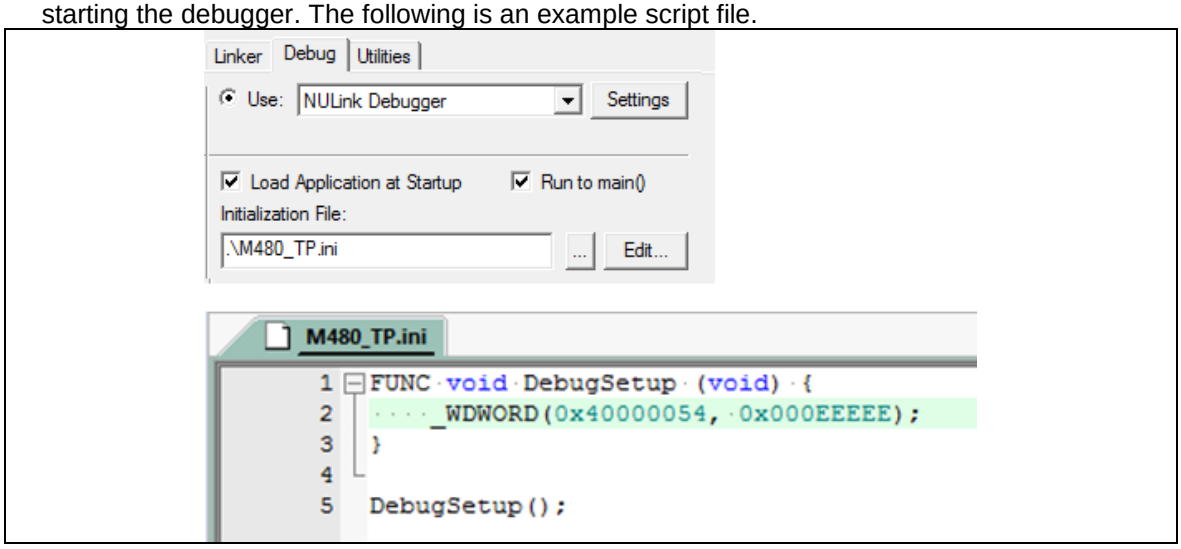

Figure 4-22 Initialize File for Trace Pins

Note: The Nu-Link driver with the version v2.07 or later will automatically setup the trace pins when starting the debugger. Users need not do above configuration for most chips.

3. After doing above settings, user must start the debugger. In Debug mode, please select "Debug" → "NuTrace" to invoke the tracing information dialog, and it will show every single executed instruction in the current application as shown below.

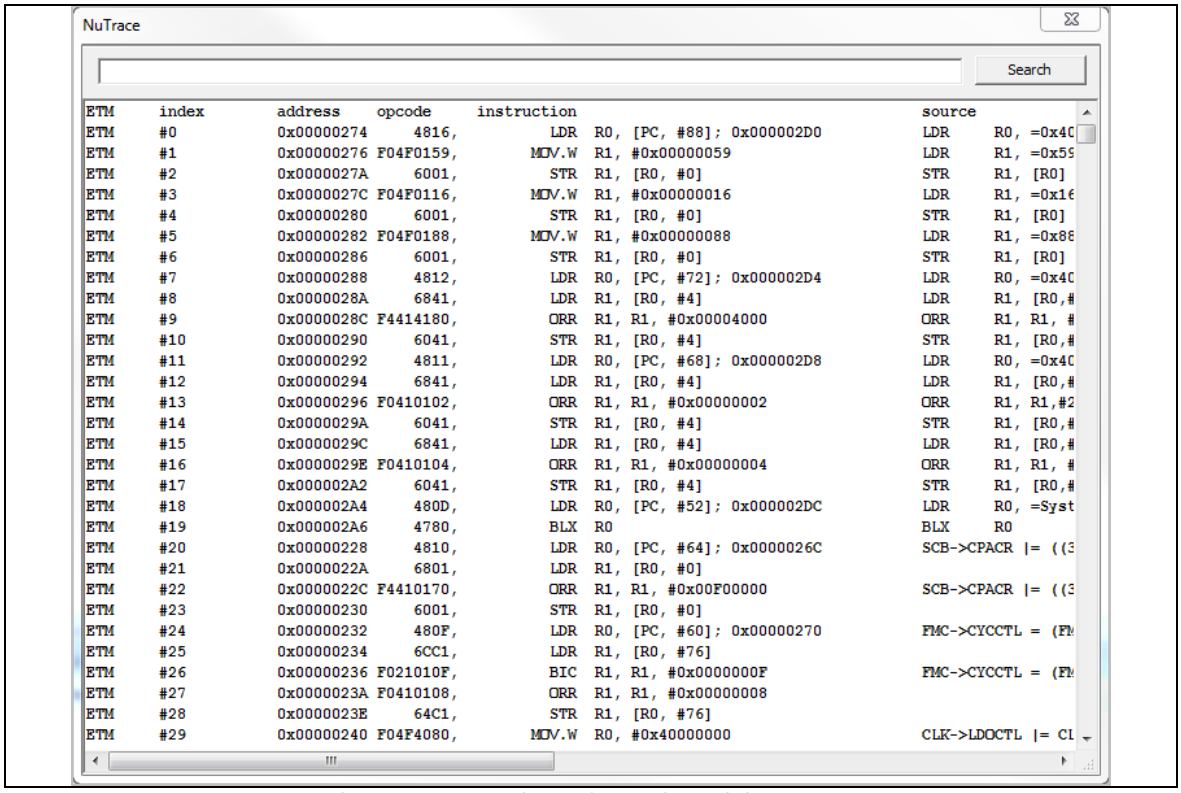

Figure 4-23 Tracing Information Dialog

In the same way, user can also enable Instruction Trace Macrocell (ITM) as the following steps:

1. Select the **ITM Stimulus Port 0** in the trace setup dialog and save the changes.

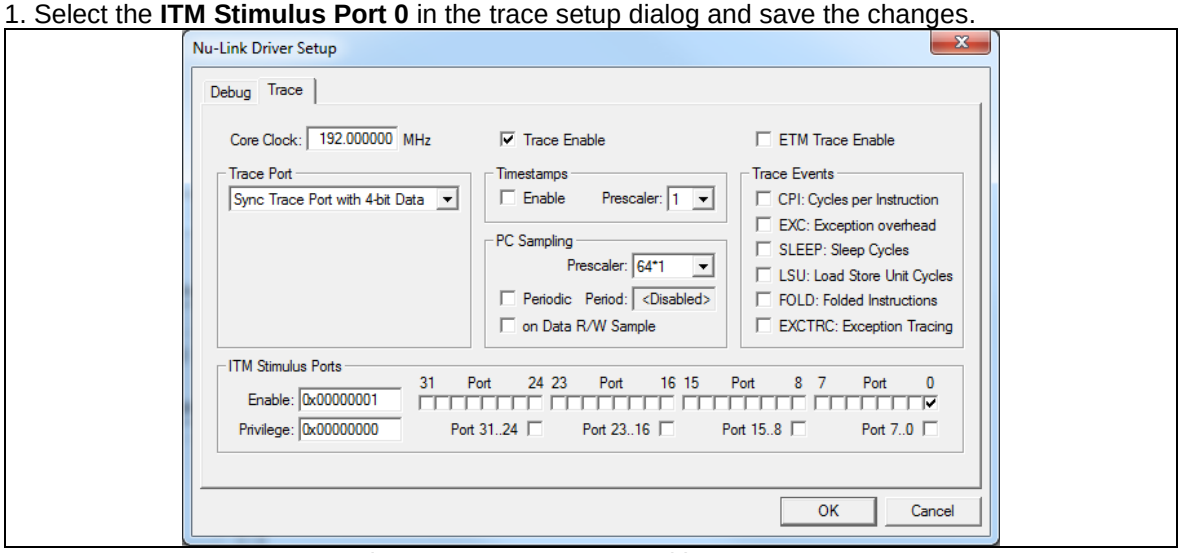

Figure 4-24 Trace Setup with ITM

2. In the development application, user needs to call the "**ITM_SendChar**" function which is defined in the CMSIS header file to trigger ITM events.

After the write to the ITM port 0, e.g. "ITM_SendChar('A')", the debugger will get the data out of the processor and display it in the Debug (printf) Viewer window as shown below.

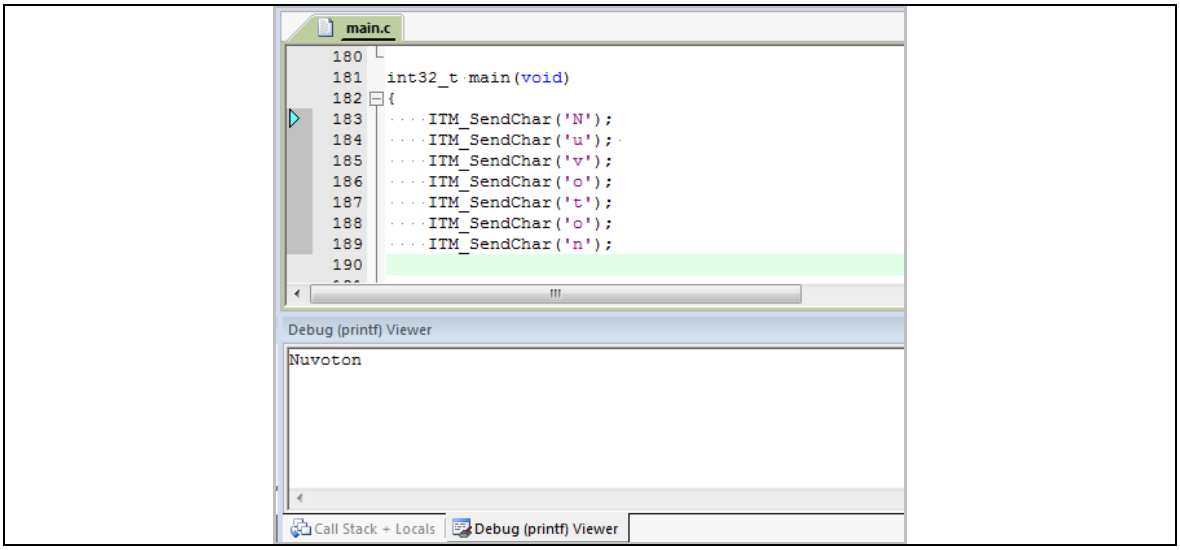

Figure 4-25 Debug Viewer with ITM Data

4.2.7 Memory Access while Running

When driver setup window with memory access running function item means support SFR and RAM value keep update when ICE in free run mode.
Setting as following and enable **"Periodic Window Update"** in debug mode.

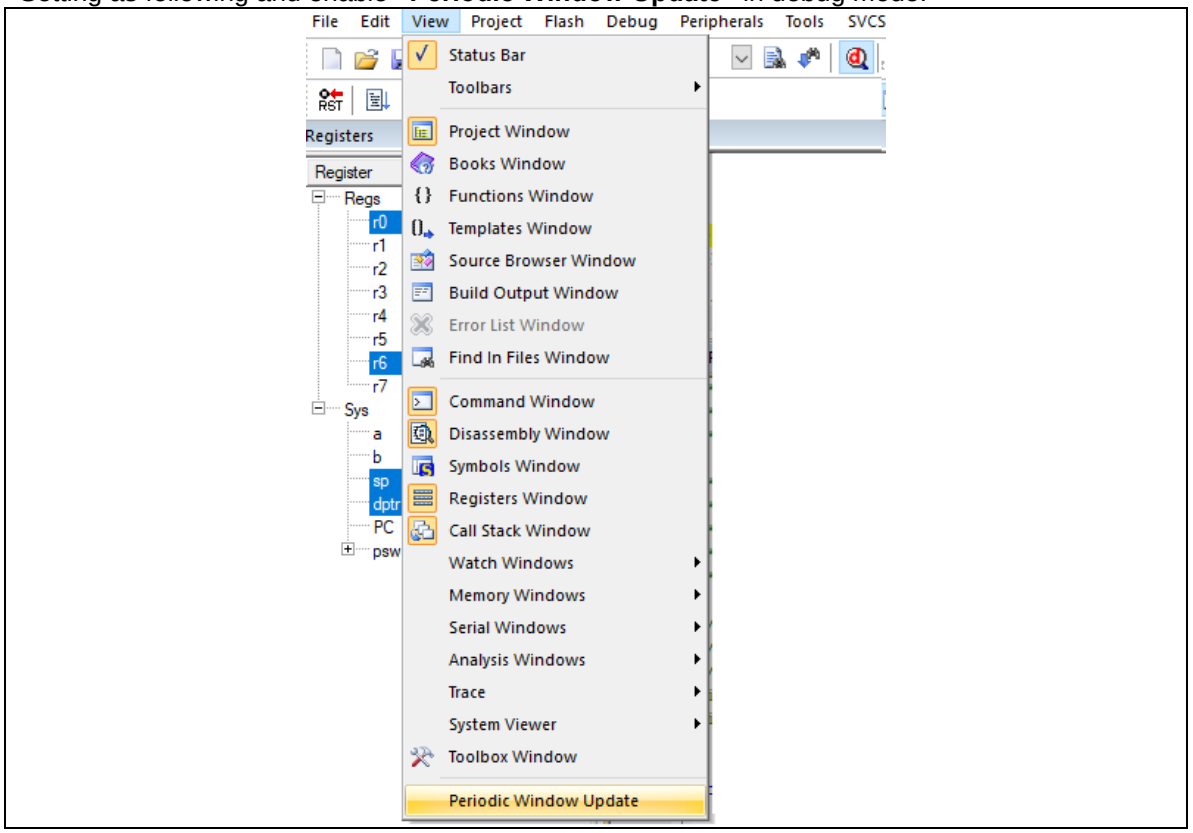

Figure 4-26 Periodic Window Update

5 Firmware Update

The Nu-Link firmware upgrade steps are described below.

1. Run Keil® µVision IDE. "**Options for Target － Debug**" and click "**Settings**" button or "**Options for Target － Utilities**" and click "**Settings**" button. If the current firmware version is not consistent with the installed Nu-Link Keil® driver, a dialog box will pop up informing the firmware update as follows.

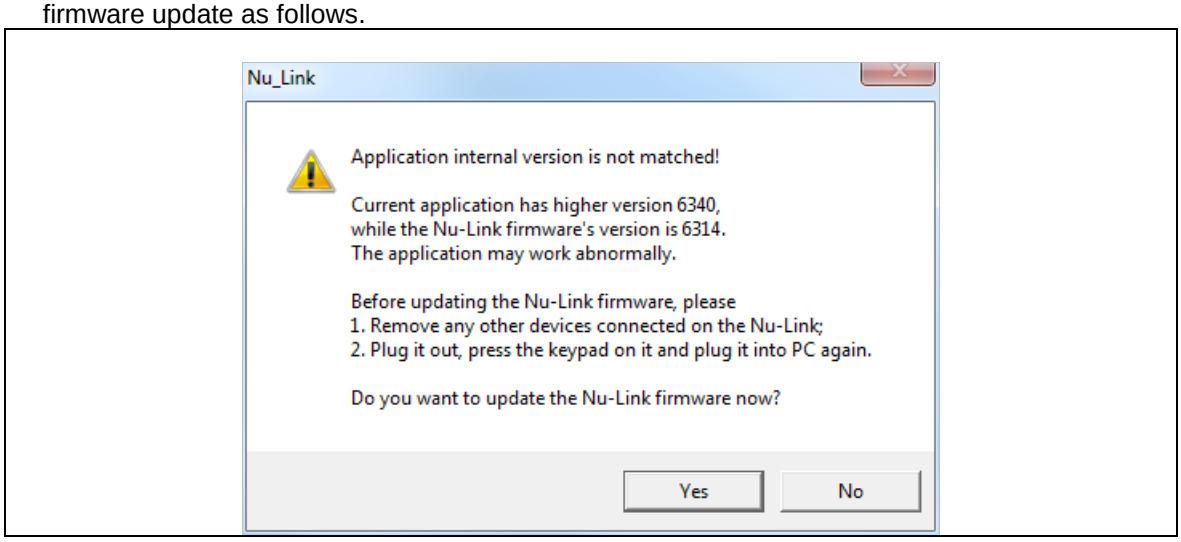

Figure 5-1 Firmware Update Selection Dialog Box

2. Click "**Yes**" to update firmware or click "**No**" to cancel.

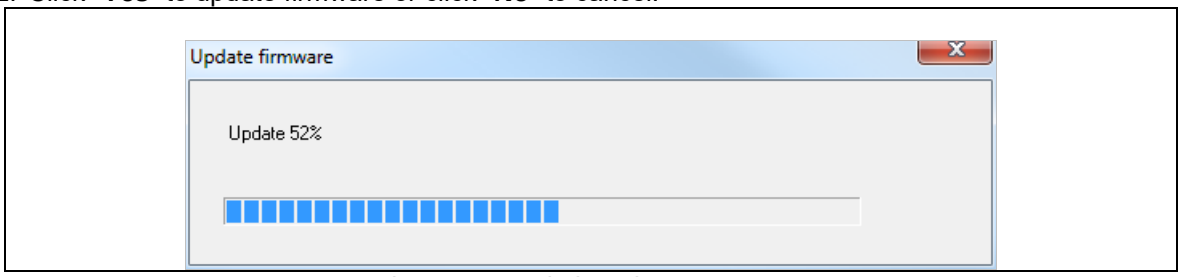

Figure 5-2 Updating Firmware

3. When update is complete, it is necessary to recreate a connection between Nu-Link and PC, and pop-up firmware update OK dialog as follows.

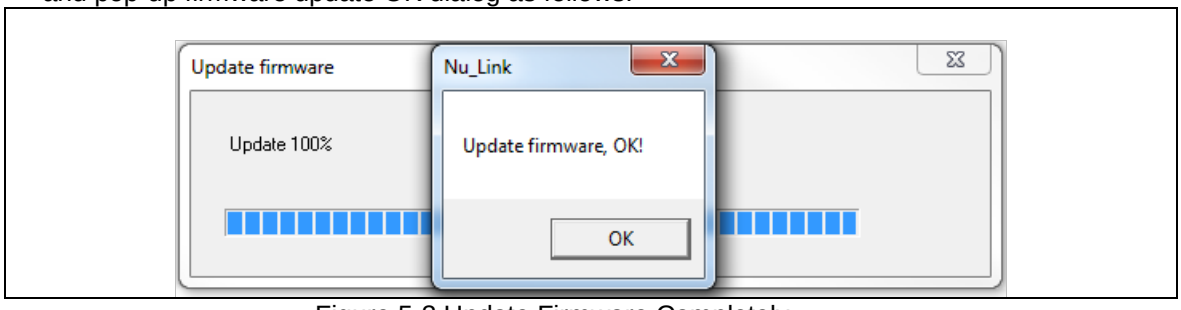

Figure 5-3 Update Firmware Completely

If you are using Nu-Link2 adapter, you can also use drag and drop method to upgrade firmware. Press the button on Nu-Link2 adapter and plug in USB cable, you will see a disk name “Nu-Link2”. Drag and drop bin file into it will upgrade Nu-Link2 firmware. (Note: if you see disk name “NuMicro MCU” it will upgrade the target device firmware instead of Nu-Link2 itself.)

6 Troubleshooting

6.1 ICE Disconnected When MCU Runs in XOM Region

Some NuMicro® chips support XOM feature, when MCU runs in the XOM region, it won't send any response to ICE.
The ICE will be disconnected after timeout in this situation, the timeout interval is different from each ICE vendor.

6.2 Breakpoint Limitation of XOM Region

When user wants to set breakpoints after XOM region, the very first instruction after XOM region can't be set. User must set the address one word away from the last instruction of XOM, or it won't be stopped.

7 User-defined Chip Script

The Nu-Link driver allows users to load a user-defined script to interact with target chips. To get started, the user should add a script file named “Nu_Link_Driver.lua” under the project directory.

7.1 Predefined Actions

The following predefined actions are called during a phase of the debugging.

1. SetupTarget

2. ResetTarget

3. StartRun

4. InRunning

5. StopRun

6. SetupTrace

7.2 User API

The following additional ICE functions can be be used in the script. Note that 3.~7. should go with the bridge interface of Nu-Link2-Pro.

1. ReadMem32 (address)
   return: value

2. WriteMem32 (address, value)
   return: none

3. AdcScan (scan_count)
   return: array { scanned_value1, value2, value3, … }

4. PwmCapture ()
   return: success_or_fail, unit_time_of_counter (ns), high_period_count, low_period_count

5. PwmOutput (gen_start, gen_frequency (Hz), gen_duty_cycle_percentage (0~100%))
   return: none

6. GpioInput (pin_no)
   return: pin_status

7. GpioOutput (pin_no, pin_status)
   return: none

7.3 Script Example

The following is a simple sample of toggling GPIO while the target is in the running state. The output messages will be shown in the Build Output window.

```
function ResetTarget()
    -- PA0->MODE (Output)
    NuLink:WriteMem32(0x40004000, 0x00000001)
end

function InRunning()
    -- PA0->PDIO
    local PA0_PDIO = NuLink:ReadMem32(0x40004800)
    print(string.format("PA0->PDIO: %d", PA0_PDIO))

    -- Toggle IO
    PA0_PDIO = bit32.bxor(PA0_PDIO, 0x00000001)
    NuLink:WriteMem32(0x40004800, PA0_PDIO)
end
```

8 Revision History

Date	Revision	Description
2010.02.05	1.00	First release for beta-site test.
2010.03.08	1.01	Added Config bit, Peripheral UI, semihosting.
2010.06.23	1.02	Modified Debug Setting Dialog and Peripheral.
2010.07.22	1.03	Added M50x series and N572.
2011.08.03	1.17	Added Nano100 series and Mini51 series.
2013.07.01	1.18	Added NUC200 series, NUC123, “Nuvoton Announcement”, and “Firmware Update” chapters.
2014.02.10	1.19	Changed document format, update “Support chips” and “Peripheral” chapters.
2014.10.24	1.20	Changed document and figure format.
2016.03.25	1.31	Updated “Nu-Link Keil® Driver Configurations” chapter.
2017.02.23	2.01	Added “NuConsole” chapter.
2017.10.13	2.03	Added XOM and M2351 notes.
2018.09.05	2.05	Added “ITM/ETM Trace” chapter.
2018.12.21	2.06	Changed document and figure format.
2019.04.02	2.07	Updated “ITM/ETM Trace” chapter
2021.06.24	2.08	Added “User-defined Chip Script” chapter.