

Arm<sup>®</sup> Cortex<sup>®</sup>-M 32-bit Microcontroller

# Nu-Link Keil<sup>®</sup> Driver User Manual

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

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

www.nuvoton.com

# **Table of Contents**

1	Overview					
	1.1	Fea	tures	4		
	1.2	Sup	ported Devices	4		
2	Installing Nu-Link Keil <sup>®</sup> Driver					
	2.1	Sys	tem Requirements	5		
	2.2	Inst	allation	5		
3	Nu-L	ink Keil	<sup>®</sup> Driver Configurations	6		
	3.1	Deb	bug Form	6		
	3.2	Flas	sh Programming Form	8		
4	4 Example – Create & Debug a Project					
	4.1	Cre	ate a New Project	10		
	4.2	Deb	bug a Project	15		
		4.2.1	System Viewer	18		
		4.2.2	Breakpoints	20		
		4.2.3	PinView Plug-in	21		
		4.2.4	Semihosting	21		
		4.2.5	NuConsole Plug-in	22		
		4.2.6	ITM/ETM Trace	23		
		4.2.7	Memory Access while Running	26		
		4.2.8	LCDView Plug-in	27		
5	5 Firmware Update2					
6 Troubleshooting						
	6.1	ICE	Disconnected When MCU Runs in XOM Region	30		
	6.2	Bre	akpoint Limitation of XOM Region	30		
7	Usei	r-define	d Chip Script	31		
7.1 Predefined Actions				31		
	7.2	Use	er API	31		
	7.3 Script Example					
8	Revision History					

# List of Figures

## 1 Overview

Nuvoton Nu-Link Keil<sup>®</sup> driver is a plug-in of Keil<sup>®</sup> 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<sup>®</sup> driver supports the following features. Some functions are triggered by  $\mu$ Vision. The usages of these functions can be found in Keil<sup>®</sup>  $\mu$ Vision User Guide.

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

### **1.2 Supported Devices**

Open the hyperlink and you can see the table of supported devices: Link of Supported devices.

After installing Nuvoton Nu-Link Keil<sup>®</sup> 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".

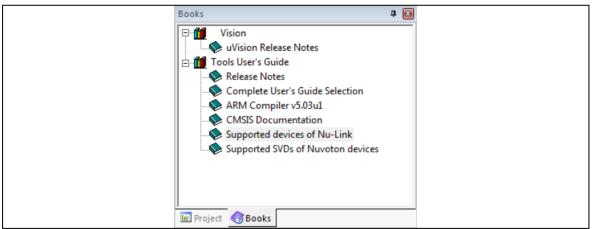


Figure 1-1 Supported Devices List File

# nuvoTon

# 2 Installing Nu-Link Keil® Driver

# 2.1 System Requirements

- Software: MDK-Arm<sup>®</sup> 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<sup>®</sup> V3.04(or later) on your PC.
- 2. Run Nu-Link\_Keil\_Driver.exe file, and then select the <*Keil*<sup>®</sup> install path>. The following files can be found after successful installation of package:
- *<Keil<sup>®</sup> install path>\Arm\BIN\Wu\_Link.dll*: Nu-Link Keil<sup>®</sup> driver's DLL file.
- <Keil<sup>®</sup> install path>\Arm\NULink: COPYRIGHT, License, Nu-Link announcement checker and so on.
- *<Keil<sup>®</sup> install path>\Arm\Flash*: Flash programming algorithm.
- <*Keil<sup>®</sup> install path*>\*Arm*\*SFD*: System viewer files.
- <Keil<sup>®</sup> install path> \Arm\NULink\Nuvoton\_NuMicro\_CortexM\_DataBase.cdb : NuMicro<sup>®</sup> Cortex<sup>®</sup>-M MCU database, this file is only for MDK less than v5.0.
- <*Keil*<sup>®</sup> install path>\Arm\HLP: Help documents.

After installation, please make sure the Nu-Link Keil<sup>®</sup> 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<sup>®</sup> Driver Configurations

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

### 3.1 Debug Form

Run  $\mu$ Vision, select "**Options for Target – Debug**", and then click the "**Settings**" button. The debug setting form is shown as follows.

Nu-Link Driver Setup			
Nu-Link		Chip Select	Supporting Forum
Driver Version:	6815r	Chip Type: NUC100 💌	EN: http://forum.nuvoton.com/
ICE Version:	6815	- Reset Options	SC: http://www.nuvoton-mcu.com/
Device Family:	Cortex-M	Connect: Normal 💌	
Device ID:	0x0BB11477	Reset: Autodetect 💌	
Port:	SW 💌	Download Options	
Max Clock:	1MHz 💌	Verify Memory Code	
Power Control-			
10 Voltage	O 1.8v O 2.	5v 🕫 3.3v 🔿 5v	
			OK Cancel

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<sup>®</sup> 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<sup>®</sup>-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<sup>®</sup>  $\mu$ Vision IDE, select "**Options for Target – Utilities**", and click the "**Settings**" button. The Flash download form is shown as follows.

Flash Download for NUC10	0		×
Flash Select	RAM for Algorithm Start: 0x20000000 Size: 0x1000	Download Function C Erase Full Chip C Erase Sectors C Do Not Erase	<ul> <li>✓ Program Flash</li> <li>✓ Verify Flash</li> <li>✓ Reset and Run</li> </ul>
	vice Type Device Size ICHIP Flash 128K	Address Range 00000000H - 0001FI	Flash Breakpoint Flash BP Config0/1 Configure
	ОК	Cancel	

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<sup>®</sup> 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<sup>®</sup> reference manual for the number of hardware breakpoints (e.g. Cortex<sup>®</sup>-M0 has 4, and Cortex<sup>®</sup>-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<sup>®</sup> 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.

Chip Options of NUC100	X
Clock Source Options	
C External Crystal Clock (4~24	MHz)
Internal RC 22 MHz Oscillato	or Clock
Brown-out Voltage Options	
● 4.5V ○ 3.8V	C 2.7V C 2.2V
Brown-out Detector	Brown-out Reset
Boot Options	
C LDROM	APROM
🗖 Data Flash	Security Lock
Data Flash Base Address [19:0]:	: 0x FFFFFFFF
Config Value	
Config0: 0x FFFFFFFD	Config1: 0x FFFFFFFF
ОК	Cancel

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<sup>®</sup> driver, follow the steps below (based on NUC140).

# 4.1 Create a New Project

Run the Keil<sup>®</sup>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.

Select a CPU Data Base File
Generic CPU Data Base Generic CPU Data Base NuVoice Database NuMicro Cortex-M Database

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

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

Select Device for Target 'Target 1'  CPU Vendor: Nuvoton Device: NUC140VE3CN	
Toolset: ARM	
Data base Description:	
Image: August of the second	
OK Cancel Help	

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.

# nuvoTon

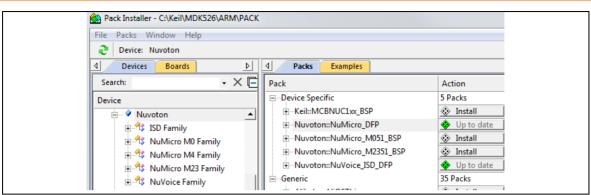


Figure 4-3 Pack Installer in MDK5

Select	ct Device for Target 'Target 1'
Dev	evice
	Software Packs
De To	/endor: Nuvoton Device: NUC140VE3CN Foolset: ARM
	OK Cancel Help

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".

# nuvoton

V Options for Target 'Target 1'	X
Device Target Output   Listing   User   C/C++   Asm   Linker   Debug   Utilities	8
Nuvoton NUC140VE3CN Xtal (MHz): 50.0	
Operating system: None 🔽 🗍 Use Cross-Module	e Optimization
System-Viewer File (.Sfr): Use MicroLIB SFD\Nuvoton\NUC100CN_v1.SFR	🗖 Big Endian
Read/Only Memory Areas	reas
default off-chip Start Size Startup default off-chip	Start Size Nolnit
□ ROM1: □ RAM1: □	
□ ROM2: □ RAM2: □	
C RAM3:	
on-chip on-chip IROM1: 0x0 0x20000 ○ IF IRAM1: 0x20 IROM2:	
OK Cancel Defaults	Help

Figure 4-5 Select System Viewer File

4. Enable the "Thumb Mode" option.

V Options for Target 'Target 1'	×
Device   Target   Output   Listing   User   C/C+(Asm) Linker   Debug   Utilities	
Conditional Assembly Control Symbols Define: Undefine:	
Language / Code Generation	
Read-Write Position Independent      Thumb Mode      No Warnings     No Auto Includes	
Include Paths Misc Controls	
Assembler control string	
OK Cancel Defaults Hel	łp

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<sup>®</sup> driver. Make sure that the "**Use**:" radio box is checked, as shown below.

Options for Target 'Target 1'	×			
Device   Target   Output   Listing   User   C/C++   Asm   Linker Debug   Utilities				
C Use Simulator with restrictions Settings Use: NULink Debugger	✓ Settings			
Load Application at Startup     Initialization File:     Initialization File:     Initialization File:	Run to main()			
I Watch Windows & Performance Analyzer I Watch Windows	gs Toolbox System Viewer			
DARMCM1.DLL         pCM0         TARMCM1.DLL         pCM0           Warn if outdated Executable is loaded         Warn if outdated Executable         Manage Component Viewer Description Files	is loaded			
OK Cancel Defaults	Help			

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.

Options for Target 'Target 1'
Device   Target   Output   Listing   User   C/C++   Asm   Linker   Debug Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programming     Use Debug Driver
NULink Debugger Settings Vupdate Target before Debugging
Init File: Edit
C Use External Tool for Flash Programming
Command:
Arguments:
Run Independent
Configure Image File Processing (FCARM):
Output File: Add Output File to Group:
Source Group 1
Image Files Root Folder:
OK Cancel Defaults Help

Figure 4-9 Select Flash Programming Driver

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

Flash Download for NUC100			×
Flash Select	RAM for Algorithm Start: 0x20000000 Size: 0x1000	Download Function C Erase Full Chip C Erase Sectors C Do Not Erase	_
Programming Algorithm Description Devic NUC100 128kB F ONCH	e Type Device Size IIP Flash 128K	Address Range 00000000H - 0001FI	Flash Breakpoint Enable Flash BP Config0/1 Configure
	ОК	Cancel	

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<sup>®</sup> driver (assuming that the installation and configuration steps had been performed). Please follow the steps below:

• Run the Keil<sup>®</sup> µVision IDE. Select "**Project – Open Project....**" and the "Open Project" dialog pops up. Choose certain project as follows.

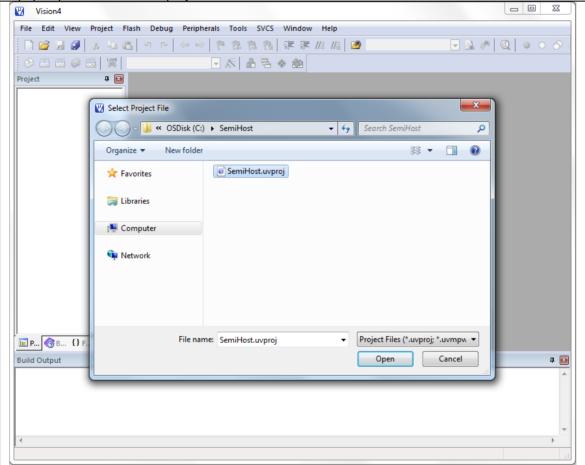


Figure 4-11 Open Project

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

C:\SemiHost\SemiHost.uvproj - Vision4	×
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
□ 📽 🖬 🕼   3 👊 🛍   9 🗠   ← →   🥐 豫 豫 豫   準 準 准 版   🖄   🖉   🔍 🔍 🔍   • <	0
😥 🎬 🕮 🥪 🤐 🙀 SemiHost 💽 🔊 📥 🗟 🚸 🚳	
Project 🚰 Build (F7) 🔛 Main.c 🖆 SH_startup_NUC1xx.s	▼ ×
S Build target files	
← CMSIS 2 /*	
Driver 3 /* Copyright (c) Nuvoton Technology Corp. All rights reserved.	
Brc 4 /* 5 /*	
5 /*	
7 #include <stdlib.h></stdlib.h>	
8 #include <stdio.h></stdio.h>	
9	
10 void SystemInit()	=
13 - 13 - 14 int main()	
16 printf("Hello, Nuvoton!\n");	
17	
18 return 0;	
19 }	
	-
□ P (⑦ F   0 + T   -   < □	•
Build Output	д 🖂
".\obj\SemiHost.axf" - 0 Error(s), 0 Warning(s).	*
*** Performing Cross-Module-Optimization:	
compiling SH_retarget.c linking	
Program Size: Code=684 RO-data=224 RW-data=8 ZI-data=1136	
After Build - User command #1: fromelfbin ".\obj\SemiHost.axf"output ".\obj\SemiHost.bin"	
After Build - User command #2: fromelftext -c ".\obj\SemiHost.axf"output ".\obj\SemiHost.txt"	E
".\obj\SemiHost.axf" - 0 Error(s), 0 Warning(s).	
*** Completed Cross-Module-Optimization after 2 iteration(s).	
4	
	P
Build target files Nuvoton N	lu-Lir

Figure 4-12 Build the Project

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

C:\SemiHost\SemiHost.vproj - Vision4	×
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
🗋 😂 🛃 🥔 🕺 🛍 🕲 🗢 📯 🦛 🕐 🥐 🎊 🎊 🎼 🊎 澤 //E //長 🖄 HID_IntinCallback 💿 見 💸 👰 🧕 🧕	ି ଚ
🥸 🕮 🧼 🗮 🕅 SemiHost 💽 🔊 🛔 🗟 🧇 🍏	
Project 🛛 🕫 Download tup_NUC1xx.s	▼ ×
Exercition SemiHost	
Driver 3 /* Copyright (c) Nuvoton Technology Corp. All rights reserved.	
E _ Src 4 /*	
5 /*	
6	
7 #include <stdlib.h></stdlib.h>	
8 #include <stdio.h></stdio.h>	
void SystemInit()	
	=
12 }	
13 L	
14 int main()	
15	
16 printf("Hello, Nuvoton!\n");	
17 18 return 0:	
18 return 0; 19 }	
	•
uild Output	<b>д</b> [
'.\obj\SemiHost.axf" - 0 Error(s), 0 Warning(s).	
** Performing Cross-Module-Optimization: compiling SH retarget.c	
inking	
rogram Size: Code=684 RO-data=224 RW-data=8 ZI-data=1136	
ffer Build - User command #1: fromelfbin ".\obj\SemiHost.axf"output ".\obj\SemiHost.bin"	
After Build - User command #2: fromelftext -c ".\obj\SemiHost.axf"output ".\obj\SemiHost.txt"	
'.\obj\SemiHost.axf" - 0 Error(s), 0 Warning(s).	
<pre>(** Completed Cross-Module-Optimization after 2 iteration(s).</pre>	_
4	Þ
Download code to flash memory Nuvoton N	La Lie
Download code to flash memory Nuvoton N	iu-tir

Figure 4-13 Download the Program

- ηυνοτοη
  - 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.

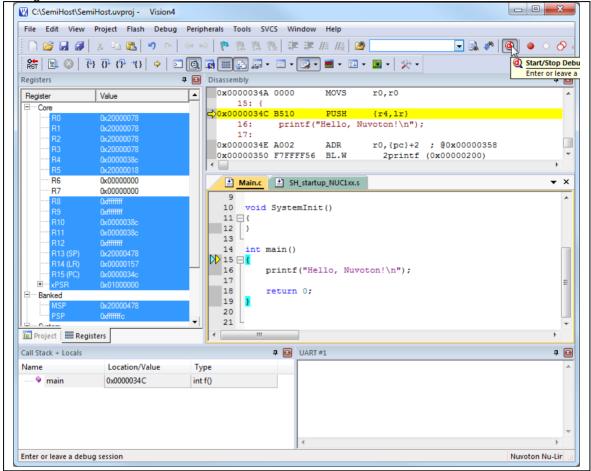


Figure 4-14 Debug Window

• 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**"  $\rightarrow$  "**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.

# nuvoTon

GCR		<b>д</b>	×
Propert	ty	Value	
+ PDI	ID	0x20014018	
+ RST	TSRC	0x0000061	Ĩ.
+ IPR	STC1	0	Ì.
+ IPR	STC2	0	Î.
+ BOI	DCR	0x0000086	Ĩ
+ TEN	MPCR	0	Î.
+ POF	RCR	0x00005AA5	Ĩ
	A_MFP	0	Î.
🛨 GPE	B_MFP	0	Ĩ
+ GPC	C_MFP	0	
⊕ GP[	D_MFP	0	Ĩ
🕂 GPE	E_MFP	0	
+ ALT	T_MFP	0	Ī
REG		0	-
PDID [Bits 31	10] RO (@ 0x50000	0000) Part Device Identification Number Regist	ter

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.

GCR		џ 🗈
Property	Value	
	0x20014018	<b>_</b>
	0x0000061	
IPRSTC1	0	
CHIP_RST	0: 0 = CHIP normal operation	
CPU_RST	0: 0 = CPU normal operation	
PDMA_RST	0: 0 = PDMA controller normal operation	
EBI_RST	0: 0 = EBI controller normal operation	
IPRSTC2	0	
BODCR	0x0000086	
■ TEMPCR	0	
PORCR	0x00005AA5	
■ GPA_MFP	0	
GPB_MFP	0	
GDC MED	0	-
IPRSTC1		
[Bits 310] RW (@ 0x5	0000008) IP Reset Control Register1	

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:

# nuvoTon

GCR		д 🔝
Property	Value	
	0x20014018	<b>^</b>
<b>∎</b> RSTSRC	0x00000061	
IPRSTC1	0	
CHIP_RST	0: 0 = CHIP normal operation	
CPU_RST	0: 0 = CPU normal operation	Ę
PDMA_RST	0: 0 = PDMA controller normal operation	5
EBI_RST	0: 0 = EBI controller normal operation	-
Setting this bit will only Controller(FMC), and ti clock cycles This bit is the protected '59h', '16h', '88h' to add	reset (Write-protection Bit) Ily reset the CPU kernel and Flash Memory this bit will automatically return to 0 after the 2 ed bit, It means programming this bit needs to ddress 0x5000_0100 to disable register protectio EGWRPROT at address GCR_BA+0x100 peration	write

Figure 4-17 Register Description

# 4.2.2 Breakpoints

The Nu-Link Keil<sup>®</sup> driver supports "Execution Break" and "Access Break" of  $\mu$ Vision IDE. You can check the  $\mu$ Vision user guide for details:  $\mu$ Vision User Guide  $\rightarrow$  Debugging  $\rightarrow$  Debug Windows and Dialogs  $\rightarrow$  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<sup>®</sup> driver will use software breakpoints, the number of breakpoints is unlimited.

• Hardware Breakpoints

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

• Flash Breakpoints

When the breakpoint is located in FLASH area, and hardware breakpoints are all used, Nu-Link Keil<sup>®</sup> 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<sup>®</sup> Cortex<sup>®</sup>-M core. The number of data breakpoints are limited (e.g. Cortex<sup>®</sup>- M0 has 2, and Cortex<sup>®</sup>-M4 has 4).

# 4.2.3 PinView Plug-in

In Debug mode, user can select "NuTool – PinView" from "**Debug**"  $\rightarrow$  "**NuTool – PinView**", and then check the correctness of pin assignment through GUI.

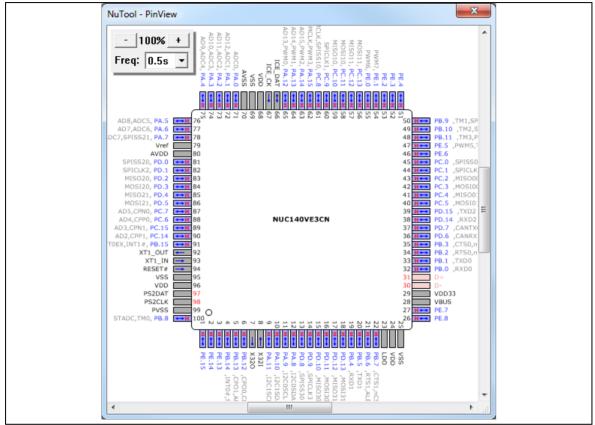


Figure 4-18 NuTool - PinView

### 4.2.4 Semihosting

The Nu-Link supports some semihosting function for user to easily output and input message in Keil<sup>®</sup> 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.

UART	#1	џ	×
Hell	.o, Nuvoton!		*
			Ŧ
_ <		1	

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\WULink\WuConsole\_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<sup>®</sup> 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.

nuvoTon	UVOTOr	٦
---------	--------	---

Stop			
Hello, Nuvoton!			
11			
Info Block	mit Text —————		Fatur
Address 0x2000024C			Enter
Address 0x2000024C	amit Text		Enter Time-out Limit (s) 5 💌
Address 0x2000024C			
Address 0x2000024C			
Address 0x2000024C Appe		Туре	

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<sup>®</sup>-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.

# nuvoTon

Core Clock: 192.000000 MHz	✓ Trace Enable	ETM Trace Enable
Capture Mode Trace After C Trace Before Trace Port Sync Trace Port with 4-bit Data	Timestamps Enable Prescaler: 1  PC Sampling Prescaler: 64*1  Periodic Period: <disabled> On Data R/W Sample</disabled>	Trace Events CPI: Cycles per Instruction EXC: Exception overhead SLEEP: Sleep Cycles LSU: Load Store Unit Cycles FOLD: Folded Instructions EXCTRC: Exception Tracing
Enable: 0x00000000	Port 24 23 Port 16 15	Port 8 7 Port 0

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.

Linker De	ebug Utilities
⊙ Use:	NULink Debugger 💌 Settings
Load /	Application at Startup 🔽 Run to main()
.\M480_1	
	M480_TP.ini
	1 = FUNC · void · DebugSetup · (void) · {
	<pre>2WDWORD(0x40000054, .0x000EEEEE);</pre>
	3 }
	4
	5 DebugSetup();

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.

								Search
ETM	index	address	opcode	instruction			source	
ETM	#0	0x00000274	4816,	LDR	RO,	[PC, #88]; 0x000002D0	LDR	R0, =0x40
ETM	#1	0x00000276	F04F0159,	MCV.W	R1,	#0x00000059	LDR	R1, =0x59
ETM	#2	0x0000027A	6001,	STR	R1,	[R0, #0]	STR	R1, [R0]
ETM	#3	0x0000027C	F04F0116,	MCV.W	R1,	#0x00000016	LDR	R1, =0x16
ETM	#4	0x00000280	6001,	STR	R1,	[R0, #0]	STR	R1, [R0]
ETM	#5	0x00000282	F04F0188,	MCV.W	R1,	#0x0000088	LDR	R1, =0x88
ETM	#6	0x00000286	6001,	STR	R1,	[R0, #0]	STR	R1, [R0]
ETM	#7	0x00000288	4812,	LDR	RO,	[PC, #72]; 0x000002D4	LDR	R0, =0x40
ETM	#8	0x0000028A	6841,	LDR	R1,	[R0, #4]	LDR	R1, [R0,#
ETM	#9	0x0000028C	F4414180,	ORR	R1,	R1, #0x00004000	ORR	R1, R1, #
ETM	#10	0x00000290	6041,	STR	R1,	[R0, #4]	STR	R1, [R0,#
ETM	#11	0x00000292	4811,	LDR	RO,	[PC, #68]; 0x000002D8	LDR	R0, =0x40
ETM	#12	0x00000294	6841,	LDR	R1,	[R0, #4]	LDR	R1, [R0,#
ETM	#13	0x00000296	F0410102,	ORR	R1,	R1, #0x0000002	ORR	R1, R1,#2
ETM	#14	0x0000029A	6041,	STR	R1,	[R0, #4]	STR	R1, [R0,#
ETM	#15	0x0000029C	6841,	LDR	R1,	[R0, #4]	LDR	R1, [R0,#
ETM	#16	0x0000029E	F0410104,	ORR	R1,	R1, #0x0000004	ORR	R1, R1, #
ETM	#17	0x000002A2	6041,	STR	R1,	[R0, #4]	STR	R1, [R0,#
ETM	#18	0x000002A4	480D,	LDR	RO,	[PC, #52]; 0x000002DC	LDR	RO, =Syst
ETM	#19	0x000002A6	4780,	BLX	RO		BLX	RO
ETM	#20	0x00000228	4810,	LDR	RO,	[PC, #64]; 0x0000026C	SCB->CI	PACR  = ((3
ETM	#21	0x0000022A	6801,	LDR	R1,	[R0, #0]		
ETM	#22	0x0000022C	F4410170,	ORR	R1,	R1, #0x00F00000	SCB->CI	PACR  = ((3
ETM	#23	0x00000230	6001,	STR	R1,	[R0, #0]		
ETM	#24	0x00000232	480F,	LDR	RO,	[PC, #60]; 0x00000270	FMC->C)	CCTL = (FM
ETM	#25	0x00000234	6CC1,	LDR	R1,	[R0, #76]		
ЕТМ	#26	0x00000236	F021010F,	BIC	R1,	R1, #0x000000F	FMC->C)	CCTL = (FM
ETM	#27	0x0000023A	F0410108,	ORR	R1,	R1, #0x0000008		
ETM	#28	0x0000023E	64C1,	STR	R1,	[R0, #76]		
ETM	#29	0x00000240	F04F4080,	MOV.W	RO.	#0x40000000	CLK->LI	OCTL  = CL -

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.

Nu-Link Driver Setup	
Core Clock:       192.00000       MHz       Image Trace Enable       ETM Trace Enable         Trace Port       Image Trace Port with 4-bit Data       Image Trace Enable       Trace Events         Sync Trace Port with 4-bit Data       Image Trace Enable       Image Trace Events       Image Core Clock:         PC Sampling       Prescaler:       Image Clock:       Image Clock:       CPI: Cycles per Instruction         PC Sampling       Prescaler:       Image Clock:       Image Clock:       CORE Clock:       CORE Clock:         Periodic       Periodic       Periodic       Closabled>       CDLD: Folded Instructions       EXCTRC:       Exception Tracing         TM Stimulus Ports       31       Port       24       23       Port       16       15       Port       0         Enable:       Dx00000000       Port 31.24       Port 2316       Port 158       Port 70       Image Clock	
OK Cancel	

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.

# nuvoTon

	main.c
L L L L L L L L L L L L L L L L L L L	180 4
	181 int32 t main(void)
	182 🖵 {
	183 ITM_SendChar('N');
	184 ····ITM_SendChar('u');
	185 ····ITM_SendChar('v');
	186 ····ITM_SendChar('o');
	187 ····ITM_SendChar('t');
	188 ····ITM_SendChar('o');
	189 ····ITM_SendChar('n');
	190
D	lebug (printf) Viewer
N	luvoton
· · · · · · · · · · · · · · · · · · ·	4
4	Call Stack + Locals Bebug (printf) Viewer

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.

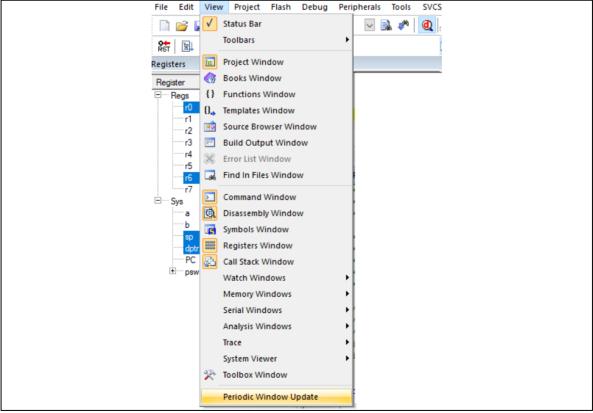


Figure 4-26 Periodic Window Update

## 4.2.8 LCDView Plug-in

In Debug mode, user can call "NuTool – LCDView" from "**Debug**" → "**NuTool – LCDView**". For details, please refer to the following link to the user manual: https://www.nuvoton.com/resource-download.jsp?tp\_GUID=UG1320220524115039

NuTool - LCDView								- 🗆 X
Project Name 🦯							ουνα	v1.04.0000
Create Mode Emulator Mode								
📑 🗄 🛃 📼 🐼 🗐								
Canvas	1							
Width: 1000 px								
Height: 500 px								
Show grid	L							
PROJ_M258	]							
M254SD3AE	1							
PID: M254SD3AE								
	]							
	Icon ID: Path ID:							
	Com: Seg: Address: [Coordinate] Top-Let		t: Bottom-l	oft: Bottor	n-Right:			
A \// 📰	[coordinate] Top Ec.	n. Top Righ	. Dottom i	Jent. Dottol	n rugnt.			
	Seg 0	Seg 1	Seg 2	Seg 3	Seg 4	Seg 5	Seg 6	Seg 7
AW∎∳ % <sup>ppm</sup> °C°F√	Com 0							
	Com 1							~
<								>

Figure 4-27 NuTool - LCDView

# 5 Firmware Update

The Nu-Link firmware upgrade steps are described below.

Run Keil<sup>®</sup> 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<sup>®</sup> driver, a dialog box will pop up informing the firmware update as follows.

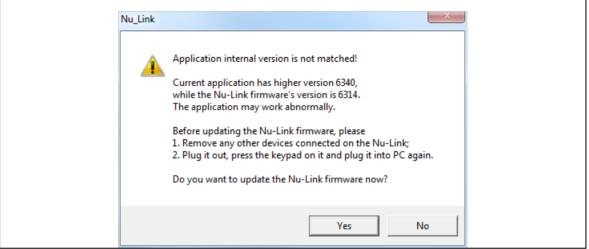


Figure 5-1 Firmware Update Selection Dialog Box

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

Update firmware	×
Update 52%	

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.

Update firmware	Nu_Link	22
Update 100%	Update firmware, OK!	
	ОК	

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<sup>®</sup> 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, 0x0000001)
    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 <sup>®</sup> 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.

Notice: Using this software indicates your acceptance of the disclaimer hereunder: THIS SOFTWARE IS FOR YOUR REFERENCE ONLY AND PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. YOUR USING THIS SOFTWARE/FIRMWARE IS BASED ON YOUR OWN DISCRETION, IN NO EVENT SHALL THE COPYRIGHT OWNER OR PROVIDER BE LIABLE TO ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.