N9H2x Keil ICE Debug User Manual

Document Information

Abstract	This document instructs the user how to debug Keil application program using J-Link ICE.
Apply to	N9H2x series

Nuvoton is providing this document only for reference purposes of microcontroller and microprocessor 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 Introduction

N9H2x series provide Keil IDE as Non-OS BSP development environment and hardware supports J-TAG debug interface. Users could use this interface to download application program to DRAM for debugging. The N9H2x Non-OS sample code uses J-Link ICE for debugging and EVB connected to J-Link ICE as shown in Figure 1-2.

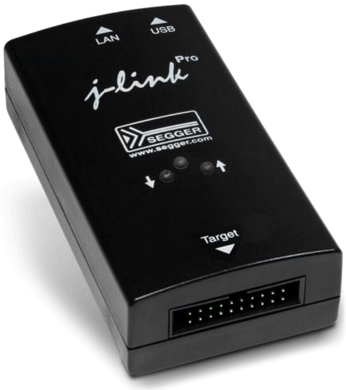


Figure 1-1 J-Link ICE

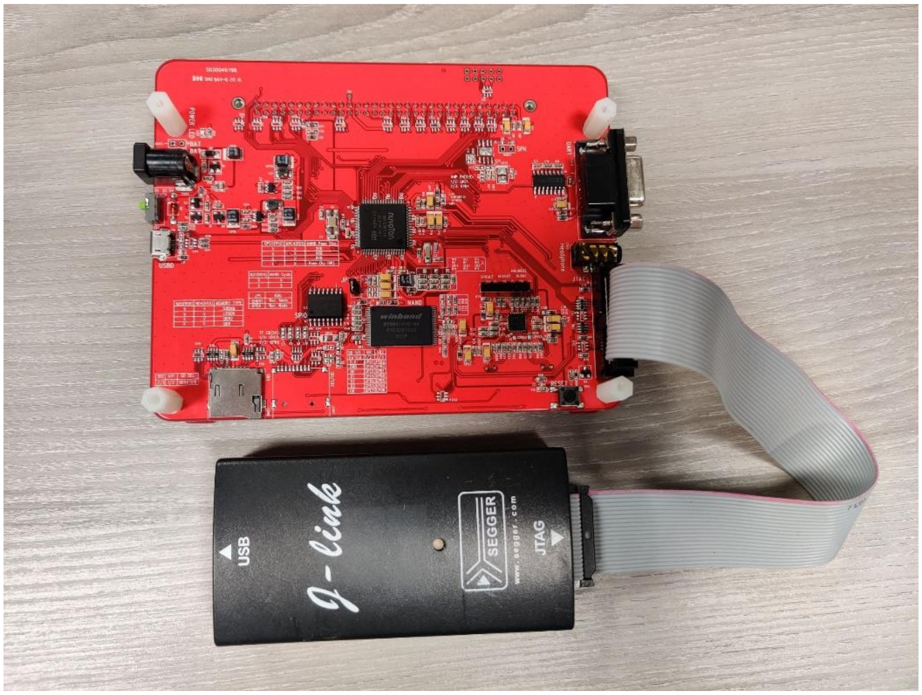


Figure 1-2 EVB Connect ICE

2 Setup Debug Environment

ARM9 based microcontrollers require the additional installation of the Keil MDK 4 Legacy Support. The user can install "NuMicro ARM9 Series Device Database Keil" program which download from the website https://www.nuvoton.com/products/microprocessors/arm9-mpus/n3290-mjpeg-series/n32901r1dn/?group=Software&tab=2 to instead of installation of Legacy Pack.

Keil supports debugging using J-Link ICE. The J-Link plug-in program can be downloaded from the website http://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack and recommend select the version above 6.88a for "NuMicro ARM9 Series Device Database Keil" program.

2.1 DEV Board Setting

N9H2x series have two boot flows, one is Normal mode and the other is Recovery mode. If user wants to use ICE for debugging, the EVB should be set to Recovery mode. Further EVB information can be found at UM_NuMaker-HMI-NK-N9H20_User_Manual_EN_Rev2.01.pdf which can be downloaded from the website:

https://www.nuvoton.com/products/microprocessors/arm9-mpus/-n9h-series/n9h20k51n/?group=Document&rt=User%20Manual&tab=2

2.2 Keil Project Setting

After opening Keil project, the user can select corresponding target depend on the DUT part number if the project has multi-targets.

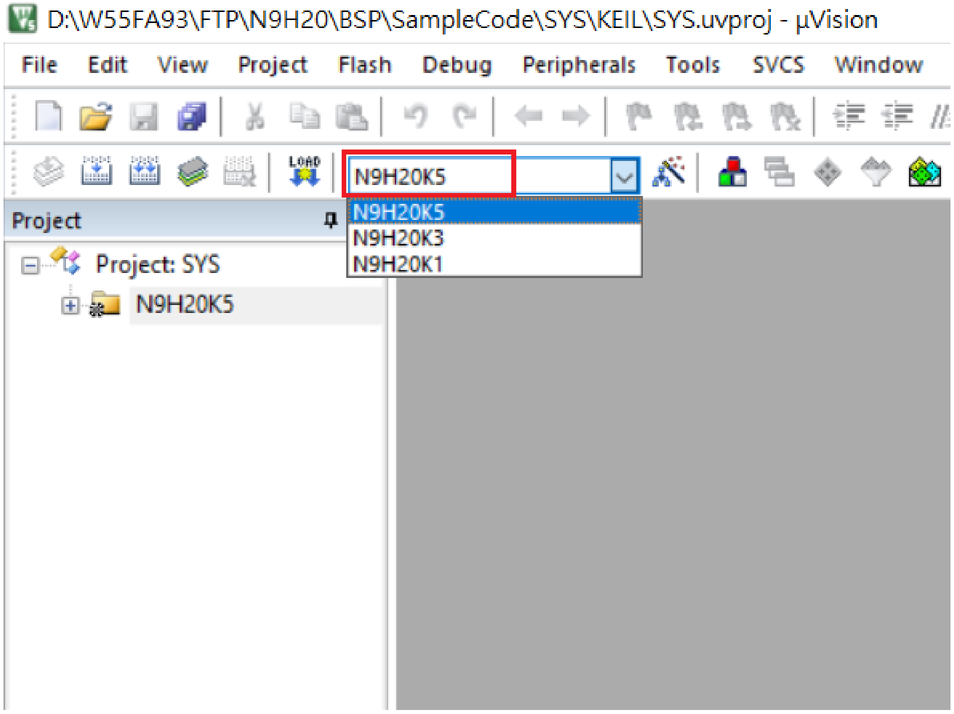
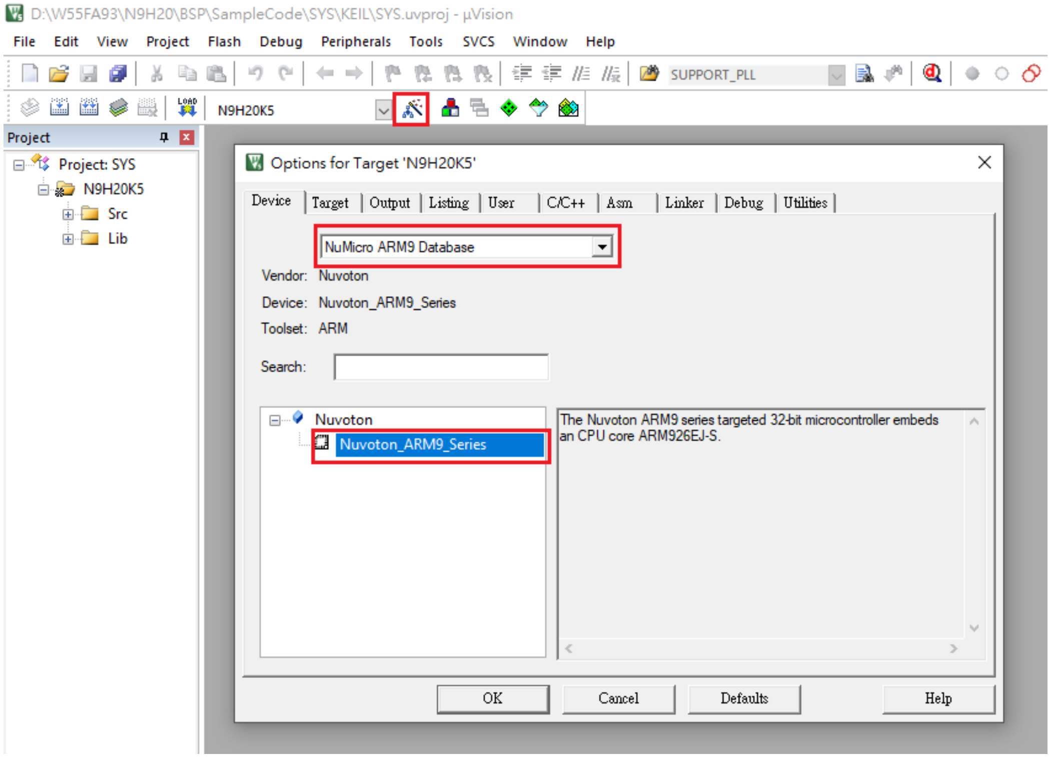

D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision

Figure 2-1 Select Target

The user can set device of project if they do not install Legacy Pack. As shown in Figure 2-2, click the **Options for Tartet** icon ➔ select **NuMicro ARM9 Database** ➔ click **Nuvoton_ARM9_Series** and then click **OK** button.


D:\W55FA93\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision

Options for Target 'N9H20K5'

NuMicro ARM9 Database

Vendor: Nuvoton

Device: Nuvoton_ARM9_Series

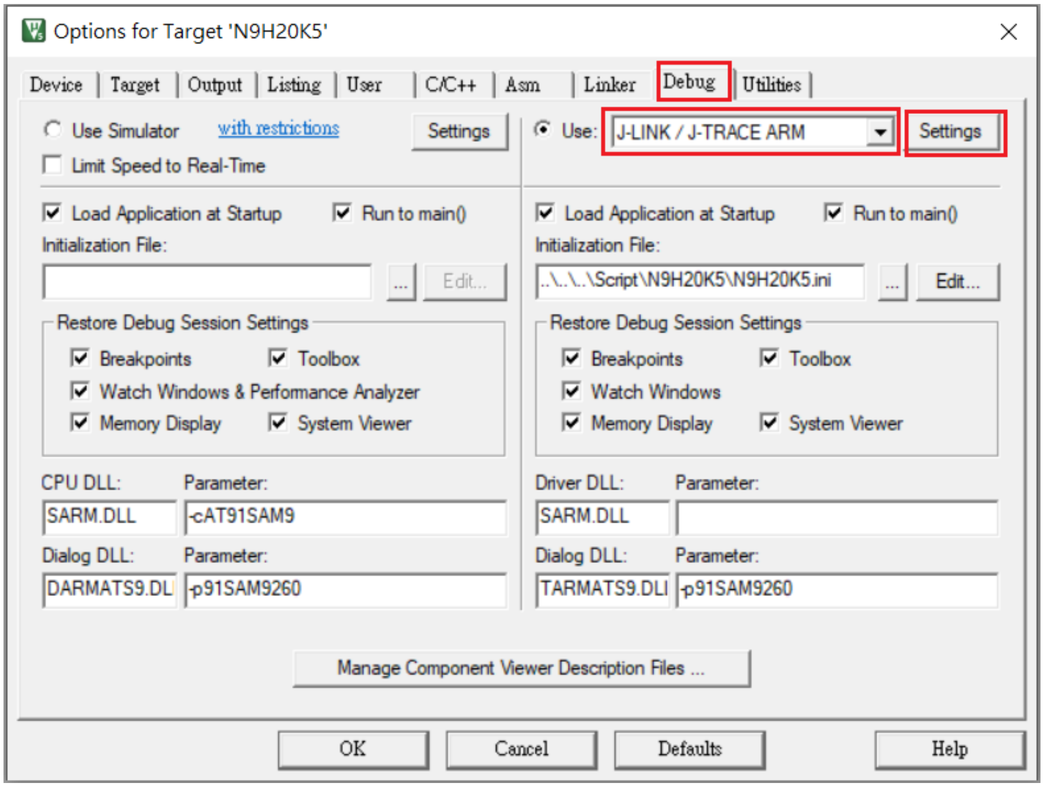
Toolset: ARM

Search:

The Nuvoton ARM9 series targeted 32-bit microcontroller embeds an CPU core ARM926EJ-S.

Figure 2-2 Device Setting

The next step is to set Debug options. As shown in Figure 2-3 and Figure 2-4 respectively, click the **Debug** button ➔ select **J-LINK/J-TRACE ARM** and then click **Settings** button to set JTAG interface configuration.



Options for Target 'N9H20K5'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

○ Use Simulator with restrictions Settings

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

... Edit...

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARM.DLL Parameter: -cAT91SAM9

Dialog DLL: DARMATS9.DLL Parameter: -p91SAM9260

● Use: J-LINK / J-TRACE ARM Settings

☑ Load Application at Startup ☑ Run to main()

Initialization File: .\..\..\Script\N9H20K5\N9H20K5.ini ... Edit...

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows

☑ Memory Display ☑ System Viewer

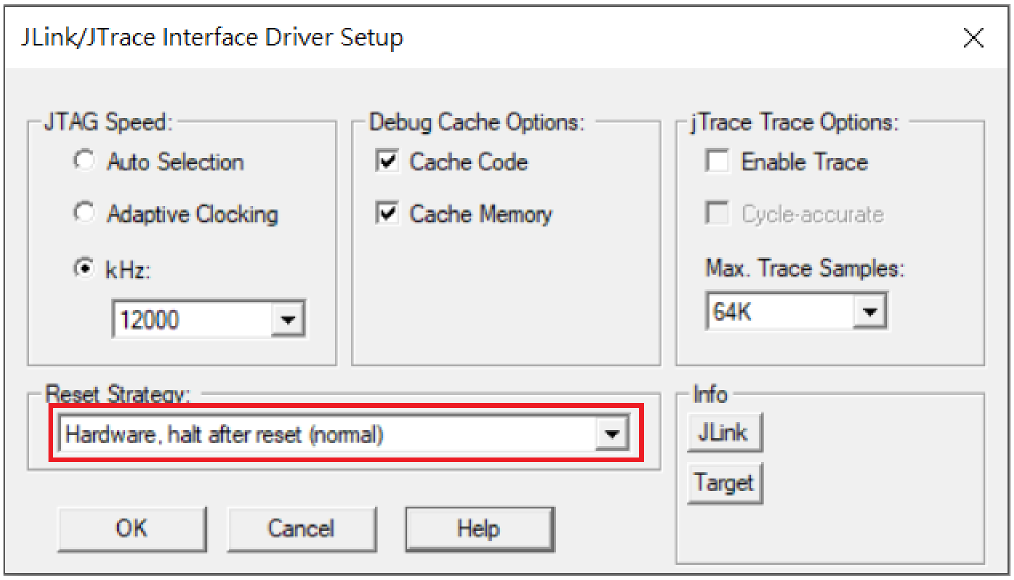
Driver DLL: SARM.DLL Parameter:

Dialog DLL: TARMATS9.DLL Parameter: -p91SAM9260

Manage Component Viewer Description Files ...

OK Cancel Defaults Help

Figure 2-3 ICE Selection



JLink/JTrace Interface Driver Setup

JTAG Speed:

○ Auto Selection

○ Adaptive Clocking

● kHz: 12000

Debug Cache Options:

☑ Cache Code

☑ Cache Memory

jTrace Trace Options:

□ Enable Trace

□ Cycle-accurate

Max. Trace Samples: 64K

Reset Strategy: Hardware, halt after reset (normal)

Info

JLink

Target

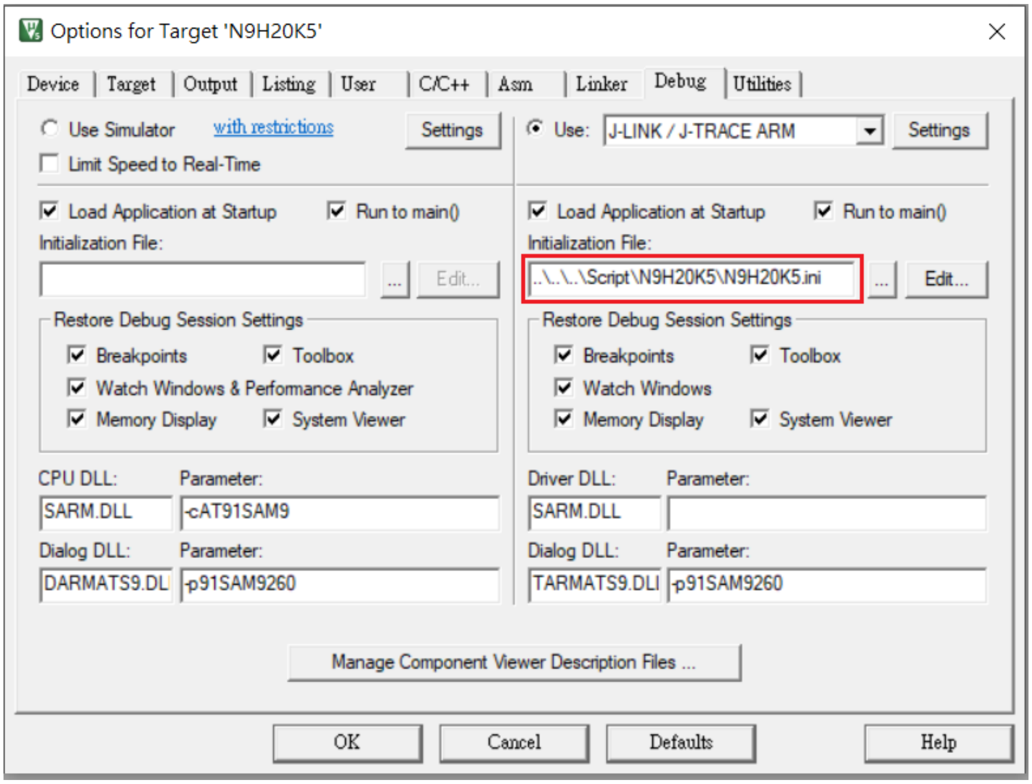
OK Cancel Help

Figure 2-4 JLink Setting in Keil MDK

The N9H2x Non-OS sample code is executed in DRAM. Before entering debug mode, it is necessary to initialize DRAM. The initialize script files are located in BSP's Script/ directory. All initialization files of N9H2x series are listed in Table 2-1. There is a unique file for each DRAM type. Please select the file according to the debug target.

Directory Name	Description
N9H20K1	N9H20K1.ini is for IP usage example, which are executed at the address 0x0.
N9H20K3	N9H20K3.ini is for IP usage example, which are executed at the address 0x0.
N9H20K5	N9H20K5.ini is for IP usage example, which are executed at the address 0x0.
N9H26	N9H26K5.ini is for IP usage example, which are executed at the address 0x0.
N9H26	N9H26K6.ini is for IP usage example, which are executed at the address 0x0.

Table 2-1 Initialization File for Keil



○ Use Simulator

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARM.DLL Parameter: -cAT91SAM9

Dialog DLL: DARMATS9.DL Parameter: -p91SAM9260

● Use: J-LINK / J-TRACE ARM

☑ Load Application at Startup ☑ Run to main()

Initialization File: ..\..\..\Script\N9H20K5\N9H20K5.ini

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows

☑ Memory Display ☑ System Viewer

Driver DLL: SARM.DLL Parameter:

Dialog DLL: TARMATS9.DLI Parameter: -p91SAM9260

Figure 2-5 Initialization File Selection

The screen is split into options for the Simulator and for the target driver.

Use Simulator

Configures the Debugger as a software-only product that simulates the instruction set of an Arm based microcontroller.

Debug Driver

Selects a driver for debugging the target hardware. For example, a J-LINK adapter driver that uses the embedded on-chip debugging hardware.

Load Application at Startup

Loads the object file specified in the field **Options for Target ➔ Output ➔ Name of Executable** into the Debugger.

Run to main()

Executes the instructions until main() is reached and stops execution. When disabled, the application stops at the first instruction. Commands defined in **Initialization File** are executed regardless of this option.

2.3 Start Debugging

To build application program in Keil MDK, click the **Rebuild** icon as shown in Figure 2-6.

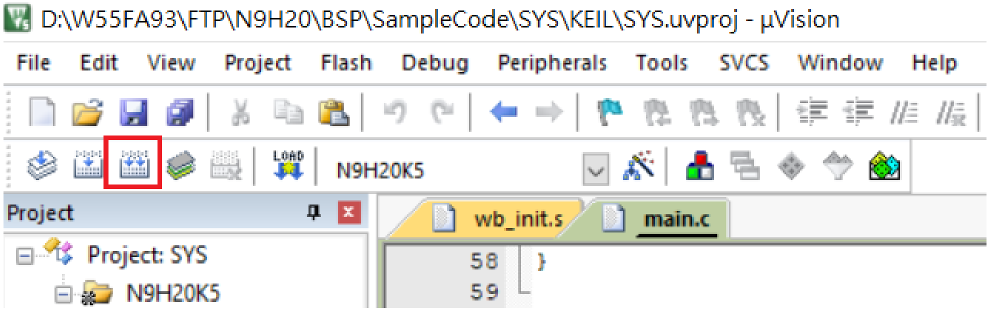

D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LOAD
N9H20K5
Project
Project: SYS
N9H20K5
wb_init.s
main.c
58 }
59

Figure 2-6 Rebuild Sample Cpde in Keil MDK

Start debugging using the menu **Debug ➔** click **Start/Stop Debug Session** to start debug with J-Link ICE as shown in Figure 2-7.

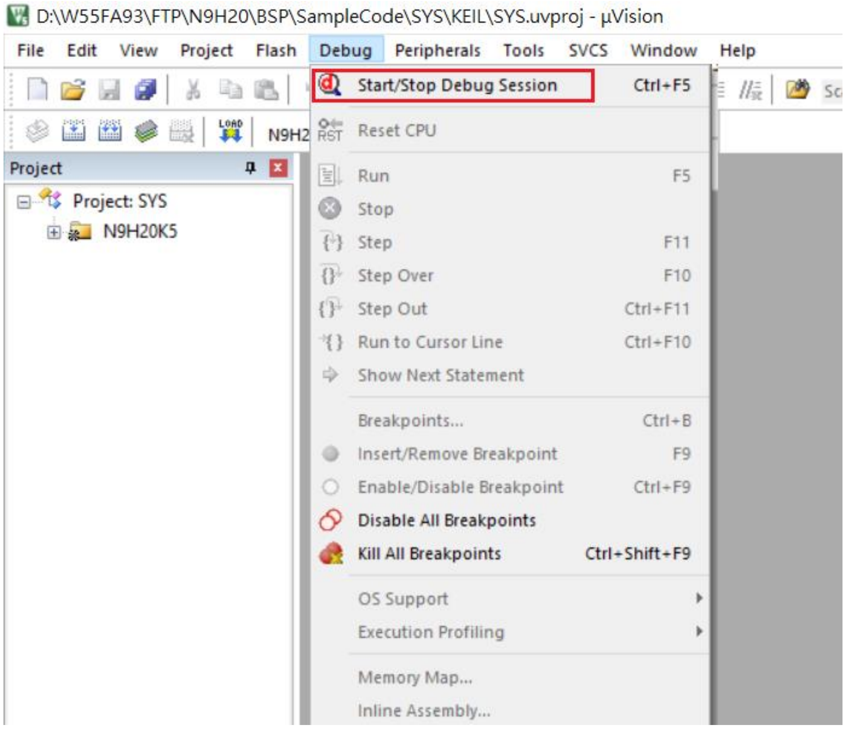
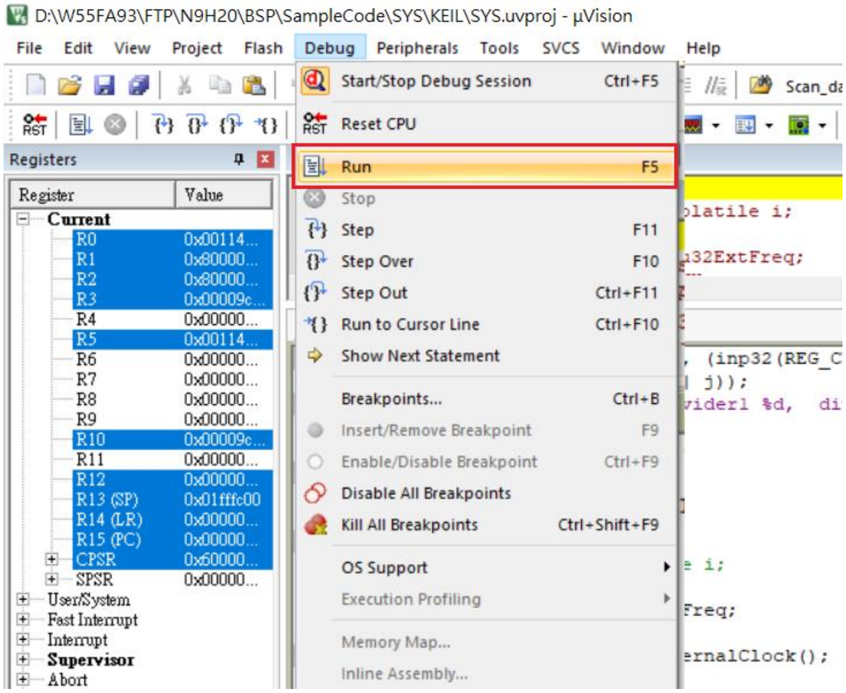

D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Project
Project: SYS
N9H20K5
N9H2

Figure 2-7 Download Binary to Device and Start/Stop Debug Session

This section introduces some simple operations for debugging.

1. Free Run the application program as shown in Figure 2-8.

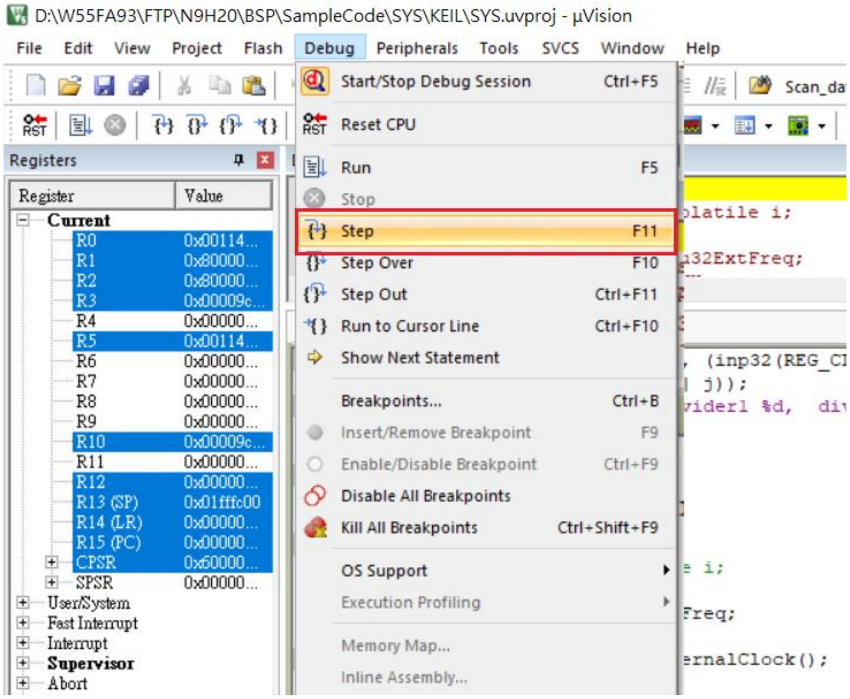

D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Registers
Scan_da

Register	Value
Current	
R0	0x00114...
R1	0x80000...
R2	0x80000...
R3	0x00009c...
R4	0x00000...
R5	0x00114...
R6	0x00000...
R7	0x00000...
R8	0x00000...
R9	0x00000...
R10	0x00009c...
R11	0x00000...
R12	0x00000...
R13 (SP)	0x01fffc00
R14 (LR)	0x00000...
R15 (PC)	0x00000...
CPSR	0x60000...
SPSR	0x00000...
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	

```
olatile i;
u32ExtFreq;
, (inp32(REG_C
| j));
viderl %d,  di
e i;
Freq;
ernalClock();
```

Figure 2-8 Free Run

2. Step & Step Over the function call as shown in Figure 2-9.



Register	Value
Current	
R0	0x00114...
R1	0x80000...
R2	0x80000...
R3	0x00009c...
R4	0x00000...
R5	0x00114...
R6	0x00000...
R7	0x00000...
R8	0x00000...
R9	0x00000...
R10	0x00009c...
R11	0x00000...
R12	0x00000...
R13 (SP)	0x01fffc00
R14 (LR)	0x00000...
R15 (PC)	0x00000...
CPSR	0x60000...
SPSR	0x00000...
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	

Figure 2-9 Step & Step Over

3. The **Memory** window shows the contents of the connected target's memory areas and allows the memory to be edited. Several memory windows can be used at a time. For example, the addresses 0xB0000200 shows content of clock control registers. Memory window and memory content as shown in Figure 2-10 and Figure 2-11 respectively.

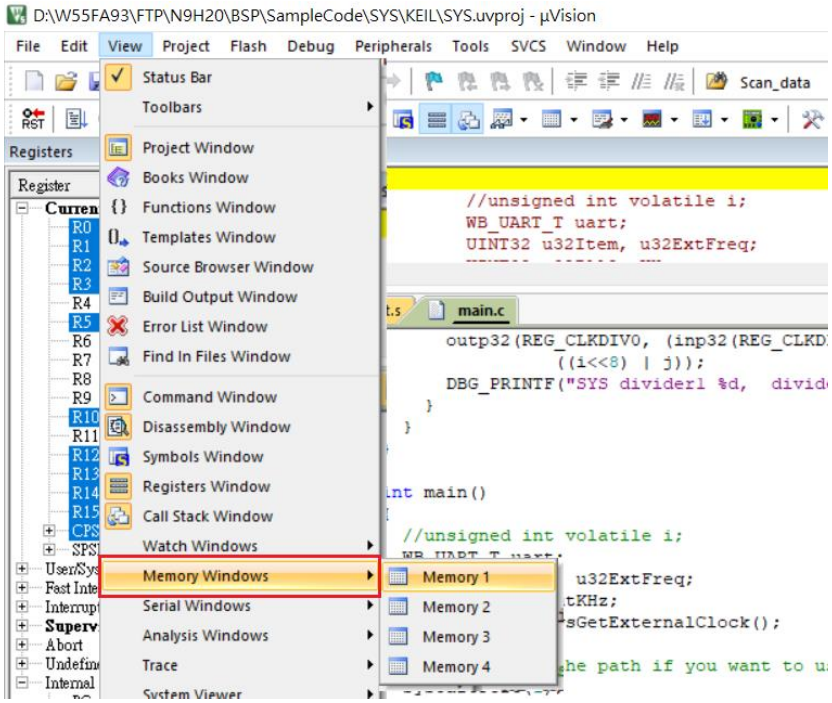
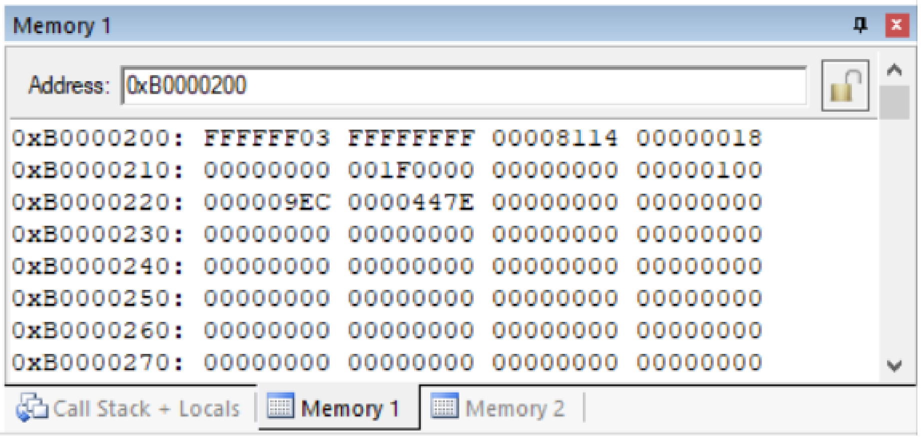


Figure 2-10 Memory Windows



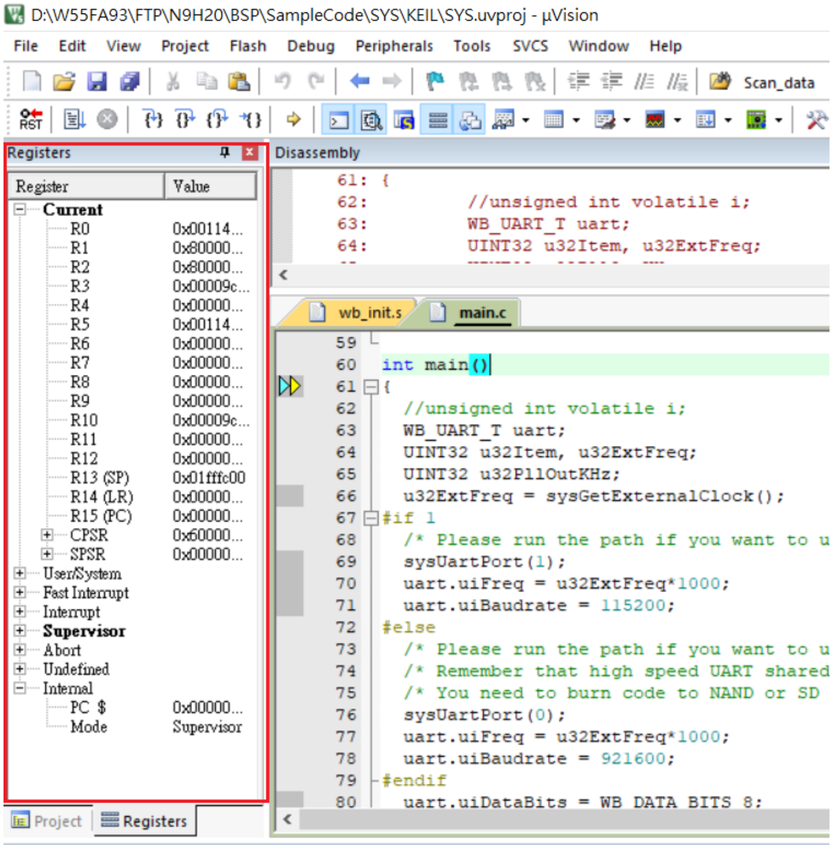
Address: 0xB0000200

```
0xB0000200: FFFFFF03 FFFFFFFF 00008114 00000018
0xB0000210: 00000000 001F0000 00000000 00000100
0xB0000220: 000009EC 0000447E 00000000 00000000
0xB0000230: 00000000 00000000 00000000 00000000
0xB0000240: 00000000 00000000 00000000 00000000
0xB0000250: 00000000 00000000 00000000 00000000
0xB0000260: 00000000 00000000 00000000 00000000
0xB0000270: 00000000 00000000 00000000 00000000
```

Figure 2-11 Content of Clock Control Registers

4. The **Registers** window includes the following registers:
   - 13 general-purpose registers R0-R12.
   - One Stack Pointer (R13 or SP).
   - One Link Register (R14 or LR).
   - One Program Counter (R15 or PC).

   The registers are available and accessible in any processor mode. When the program stops at a breakpoint, or is stepped, the **Registers** windows update to show the current values of the registers.



Register	Value
Current	
R0	0x00114...
R1	0x80000...
R2	0x80000...
R3	0x00009c...
R4	0x00000...
R5	0x00114...
R6	0x00000...
R7	0x00000...
R8	0x00000...
R9	0x00000...
R10	0x00009c...
R11	0x00000...
R12	0x00000...
R13 (SP)	0x01fffc00
R14 (LR)	0x00000...
R15 (PC)	0x00000...
CPSR	0x60000...
SPSR	0x00000...
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000...
Mode	Supervisor

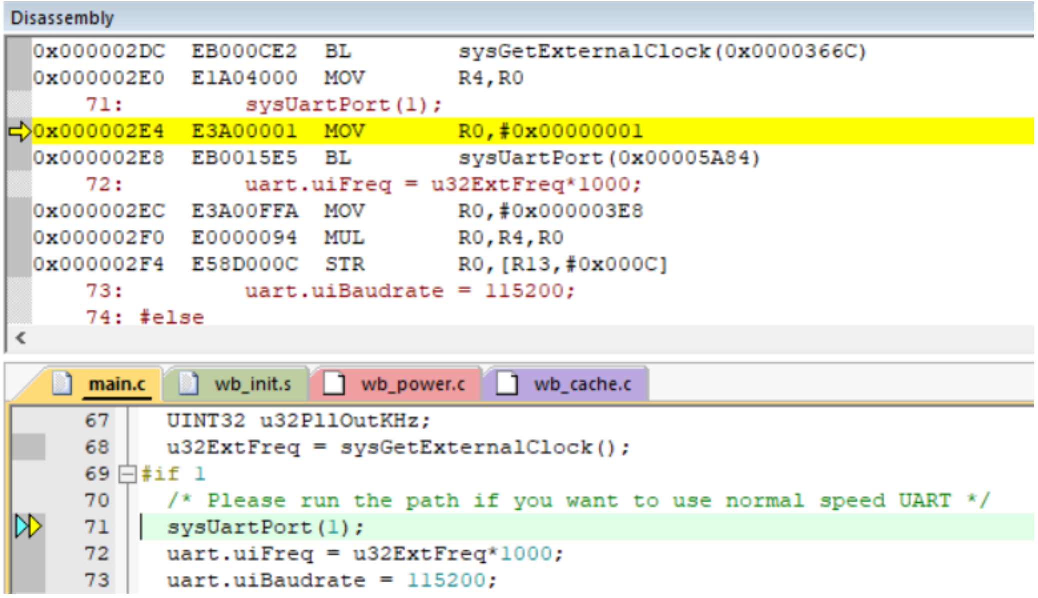
```
61: {
62:          //unsigned int volatile i;
63:          WB_UART_T uart;
64:          UINT32 u32Item, u32ExtFreq;
```

```

int main()
{
  //unsigned int volatile i;
  WB_UART_T uart;
  UINT32 u32Item, u32ExtFreq;
  UINT32 u32PllOutKHz;
  u32ExtFreq = sysGetExternalClock();
#if 1
  /* Please run the path if you want to u
  sysUartPort(1);
  uart.uiFreq = u32ExtFreq*1000;
  uart.uiBaudrate = 115200;
#else
  /* Please run the path if you want to u
  /* Remember that high speed UART shared
  /* You need to burn code to NAND or SD
  sysUartPort(0);
  uart.uiFreq = u32ExtFreq*1000;
  uart.uiBaudrate = 921600;
#endif
  uart.uiDataBits = WB_DATA_BITS_8;
```

Figure 2-12 Registers Window

5. The **Disassembly** window shows the program execution in assembly code. Open the window with the menu **View ➔** click **Disassembly Window**

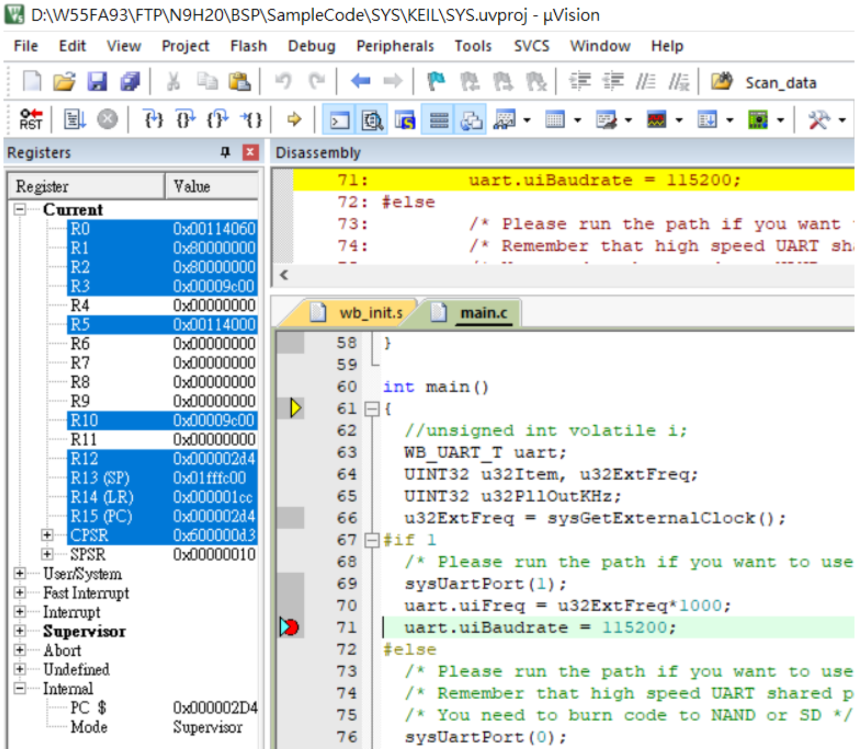


```
0x000002DC  EB000CE2  BL       sysGetExternalClock(0x0000366C)
0x000002E0  E1A04000  MOV      R4,R0
    71:           sysUartPort(1);
0x000002E4  E3A00001  MOV      R0,#0x00000001
0x000002E8  EB0015E5  BL       sysUartPort(0x00005A84)
    72:           uart.uiFreq = u32ExtFreq*1000;
0x000002EC  E3A00FFA  MOV      R0,#0x000003E8
0x000002F0  E0000094  MUL      R0,R4,R0
0x000002F4  E58D000C  STR      R0,[R13,#0x000C]
    73:           uart.uiBaudrate = 115200;
    74: #else
```

```
  UINT32 u32PllOutKHz;
  u32ExtFreq = sysGetExternalClock();
#if 1
  /* Please run the path if you want to use normal speed UART */
  sysUartPort(1);
  uart.uiFreq = u32ExtFreq*1000;
  uart.uiBaudrate = 115200;
```

Figure 2-13 Disassembly Window

6. A breakpoint can be enabled and disabled by left-clicking on its icon as shown in Figure 2-14.


D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Scan_data
RST
Registers
Disassembly
wb_init.s
main.c

Register	Value
Current	
R0	0x00114060
R1	0x80000000
R2	0x80000000
R3	0x00009c00
R4	0x00000000
R5	0x00114000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00009c00
R11	0x00000000
R12	0x000002d4
R13 (SP)	0x01fffc00
R14 (LR)	0x000001cc
R15 (PC)	0x000002d4
CPSR	0x600000d3
SPSR	0x00000010
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x000002D4
Mode	Supervisor

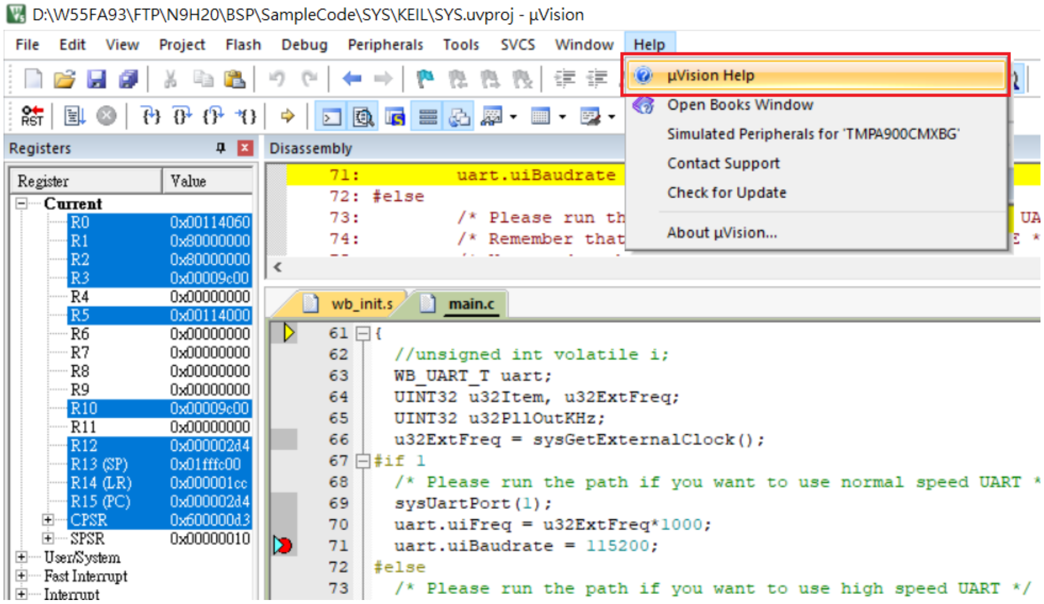
```
    71:         uart.uiBaudrate = 115200;
    72: #else
    73:         /* Please run the path if you want
    74:         /* Remember that high speed UART sh
```

```
}

int main()
{
  //unsigned int volatile i;
  WB_UART_T uart;
  UINT32 u32Item, u32ExtFreq;
  UINT32 u32PllOutKHz;
  u32ExtFreq = sysGetExternalClock();
#if 1
  /* Please run the path if you want to use
  sysUartPort(1);
  uart.uiFreq = u32ExtFreq*1000;
  uart.uiBaudrate = 115200;
#else
  /* Please run the path if you want to use
  /* Remember that high speed UART shared p
  /* You need to burn code to NAND or SD */
  sysUartPort(0);
```

Figure 2-14 Set Breakpoint

As for other operations in the debugger, the user can refer to **Help** window to get further information.


D:\W55FA93\FTP\N9H20\BSP\SampleCode\SYS\KEIL\SYS.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
µVision Help
Open Books Window
Simulated Peripherals for 'TMPA900CMXBG'
Contact Support
Check for Update
About µVision...
RST
Registers
Disassembly
wb_init.s
main.c

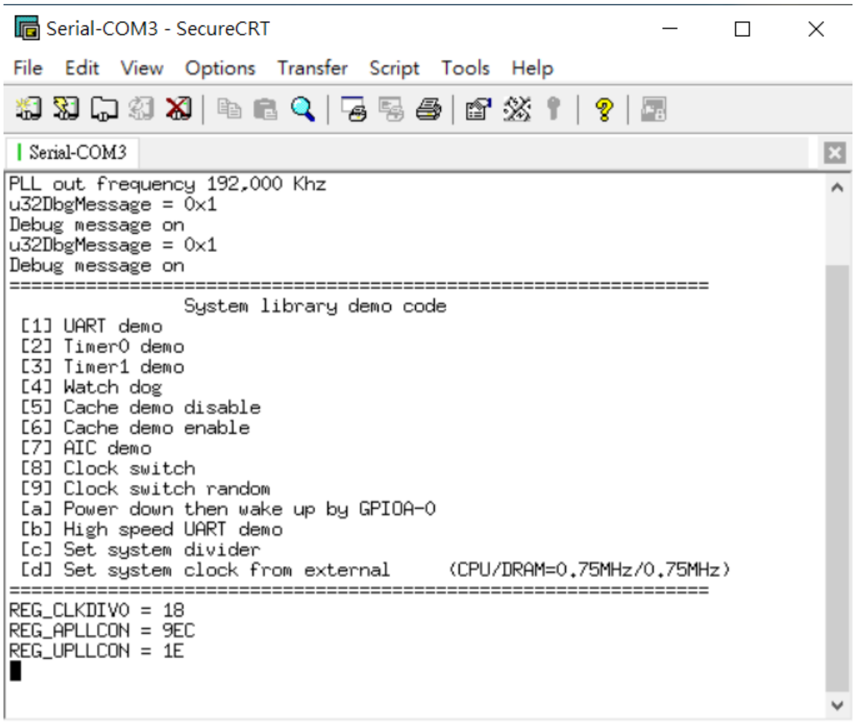
Register	Value
Current	
R0	0x00114060
R1	0x80000000
R2	0x80000000
R3	0x00009c00
R4	0x00000000
R5	0x00114000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00009c00
R11	0x00000000
R12	0x000002d4
R13 (SP)	0x01fffc00
R14 (LR)	0x000001cc
R15 (PC)	0x000002d4
CPSR	0x600000d3
SPSR	0x00000010
User/System	
Fast Interrupt	
Interrupt	

```
    71:         uart.uiBaudrate
    72: #else
    73:         /* Please run th
    74:         /* Remember that
```

```
{
  //unsigned int volatile i;
  WB_UART_T uart;
  UINT32 u32Item, u32ExtFreq;
  UINT32 u32PllOutKHz;
  u32ExtFreq = sysGetExternalClock();
#if 1
  /* Please run the path if you want to use normal speed UART *
  sysUartPort(1);
  uart.uiFreq = u32ExtFreq*1000;
  uart.uiBaudrate = 115200;
#else
  /* Please run the path if you want to use high speed UART */
```

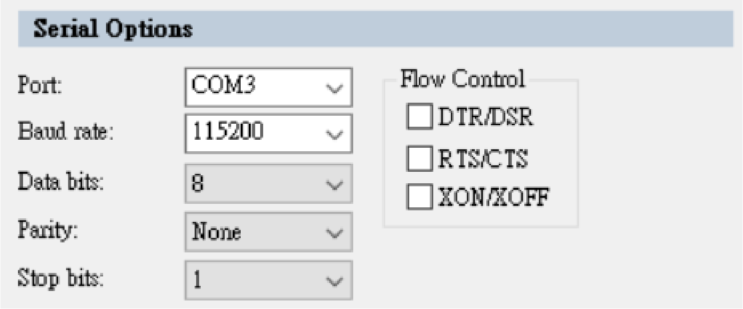
Figure 2-15 Help Window

The result of SYS sample code execution as shown in Figure 2-16 and related terminal setting as shown in Figure 2-17.



```
PLL out frequency 192,000 Khz
u32DbgMessage = 0x1
Debug message on
u32DbgMessage = 0x1
Debug message on
================================================================
                System library demo code
 [1] UART demo
 [2] Timer0 demo
 [3] Timer1 demo
 [4] Watch dog
 [5] Cache demo disable
 [6] Cache demo enable
 [7] AIC demo
 [8] Clock switch
 [9] Clock switch random
 [a] Power down then wake up by GPIOA-0
 [b] High speed UART demo
 [c] Set system divider
 [d] Set system clock from external     (CPU/DRAM=0.75MHz/0.75MHz)
================================================================
REG_CLKDIV0 = 18
REG_APLLCON = 9EC
REG_UPLLCON = 1E
█
```

Figure 2-16 Result of SYS Sample Code



Serial Options

Port: COM3

Baud rate: 115200

Data bits: 8

Parity: None

Stop bits: 1

Flow Control

□ DTR/DSR

□ RTS/CTS

□ XON/XOFF

Figure 2-17 Terminal Setting

Revision History

Date	Revision	Description
2021.07.26	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

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.