Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9Rev. 3.0 — 5 May 2025Application note

Document information

Information	Content
Keywords	AN14120, i.MX 8M, i.MX 93, i.MX 95, Cortex-M, i.MX 8MN, i.MX 8MP, i.MX 8MM, VS Code, MCUXSDK, MCUXpresso SDK, J-Link, SEGGER, Cortex-M debug
Abstract	This document describes cross-compiling, deploying, and debugging an application for the i.MX 8M Family, i.MX 8ULP, and i.MX 9 Cortex-M processor using Microsoft Visual Studio Code.



1 Introduction

This document describes cross-compiling, deploying, and debugging an application for the i.MX 8M Family, i.MX 8ULP, and i.MX 9 family Cortex-M processor using Microsoft Visual Studio Code.

1.1 Software environment

The solution could be implemented both on the Linux and Windows host. For this application note, a Windows PC is assumed, but not mandatory.

- Software versions used:
 - J-Link SEGGER V8.24
 - MCUXpresso for VS Code version 25.3.72
 - MCUX SDK 25.03.00
 - Linux BSP release 6.12.3_1.0.0 with the following images:
 - i.MX 8M Mini: core-image-minimal-imx8mmevk.wic
 - i.MX 8M Nano: core-image-minimal-imx8mnevk.wic
 - i.MX 8M Plus: core-image-minimal-imx8mpevk.wic
 - i.MX 8ULP: core-image-minimal-imx8ulpevk.wic
 - i.MX 93: core-image-minimal-imx93evk.wic
 - i.MX 95: core-image-minimal-imx95-19x19-lpddr5-evk.wic

For detailed steps on how to build these images, refer to *i.MX Linux User Guide* (document <u>IMXLUG</u>) and *i.MX Yocto Project User Guide* (document <u>IMXLXYOCTOUG</u>).

If a Windows PC is used, write the prebuild image on the SD card using Win32 Disk Imager (<u>https://win32diskimager.org/</u>) or Balena Etcher (<u>https://etcher.balena.io/</u>).

If an Ubuntu PC is used, write the prebuild image on the SD card using the below command:

\$ sudo dd if=<image_name>.wic of=/dev/sd<x> bs=1M status=progress conv=fsync

Note: Check your card reader partition and replace *sd*<*x*> with your corresponding partition.

1.2 Hardware setup and equipment

- Development kit:
 - NXP i.MX 8MM EVK LPDDR4
 - NXP i.MX 8MN EVK LPDDR4
 - NXP i.MX 8MP EVK LPDDR4
 - NXP i.MX 93 EVK for 11x11 mm LPDDR4
 - NXP i.MX 8ULP EVK LPDDR4
 - NXP i.MX 95 EVK for 19x19 mm LPDDR5
- Micro SD card: SanDisk Ultra 32-GB Micro-SDHC; Class 10 is used for the current experiment.
- Micro-USB (i.MX 8M) or Type-C (i.MX 9) cable for debug port.
- SEGGER J-Link debug probe.
- <u>MCU-link debug probe</u> This is mandatory for the i.MX 95 debug using the serial download method.

2 **Prerequisites**

Before starting to debug, several prerequisites must be met to have a properly configured debug environment.

2.1 PC Host – i.MX board debug connection

To establish the hardware debug connection, perform the following steps:

- 1. Connect the i.MX board to the host PC via the DEBUG USB-UART and PC USB connector using a USB cable. The Windows OS finds the serial devices automatically.
- 2. In *Device Manager*, under *Ports (COM & LPT)*, find two or four connected USB serial ports (COM <port_number>). One of the ports is used for the debug messages generated by the Cortex-A core, and the other is for the Cortex-M core.

Before determining the right port needed, remember:

- **[i.MX 95]**: There are four ports available in the *Device Manager*. The first port is for Cortex-M7 debug. The third port is for Cortex-A debug. The fourth port is for Cortex-M33 debug. Counting the debug ports in ascending order.
- [i.MX 8MP, i.MX 8ULP, i.MX 93]: There are four ports available in the *Device Manger*. The last port is for Cortex-M debug and the second to last port is for Cortex-A debug, counting debug ports in ascending order.
- **[i.MX 8MM, i.MX 8MN]**: There are two ports available in *Device Manager*. The first port is for Cortex-M debug and the second port is for Cortex-A debug, counting debug ports in ascending order.
- 3. Open the right debug port using your preferred serial terminal emulator (for example PuTTY) by setting the following parameters:
 - Speed to 115200 bps
 - 8 data bits
 - 1 stop bit (115200, 8N1)
 - No parity
- 4. Connect the SEGGER/MCU-link debug probe between the JTAG connector of the EVK and the PC.

If the i.MX board JTAG interface has no guided connector, the orientation is determined by aligning the red wire to the pin 1, as in Figure 1.

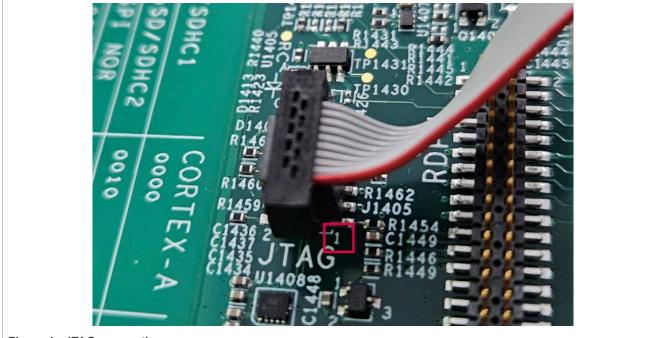


Figure 1. JTAG connection

2.2 VS Code configuration

To download and configure the VS Code, perform the following steps:

1. Download and install the latest version of Microsoft Visual Studio Code from the official <u>website</u>. If using Windows as the host OS, choose the "Download for Windows" button from the Visual Studio Code main page.

A https://code.visualstudio.com
X Visual Studio Code Docs Updates Blog
Code editing. Redefined. Free. Built on open source. Runs everywhere. Download for Windows Stable Build Web, Insiders edition, or other platforms By using VS Code, you agree to its license and privacy statement.

Figure 2. Download Microsoft Visual Studio Code

2. After installing Visual Studio Code, open it and choose the "Extensions" tab or press the Ctrl + Shift + X combination.

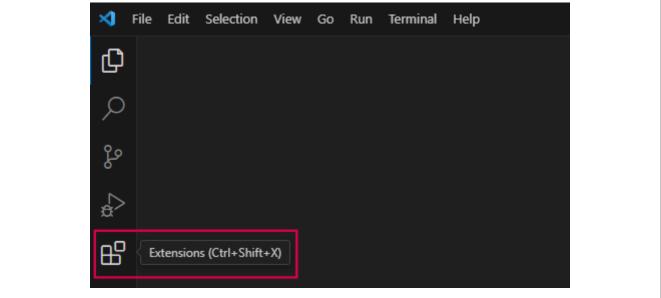


Figure 3. Microsoft Visual Studio Code Extensions tab

3. In the dedicated Search bar, type *MCUXpresso for VS Code* and install the extension. A new tab appears in the left side of the VS Code window.

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9



2.3 MCUXpresso extension configuration

To configure the MCUXpresso extension, perform the following steps:

- 1. Click the MCUXpresso extension dedicated tab from the left side bar. From the QUICKSTART PANEL, click Open MCUXpresso Installer and give permission for downloading the installer.
- 2. The installer window appears in a short time. Click *MCUXpresso SDK Developer* and on *SEGGER J-Link* then click the *Install* button. The installer installs the needed software for archives, toolchain, Python support, Git, and debug probe.

Note: For i.MX 95, to use the serial download method described in <u>Section 4 "Prepare the board for the</u> <u>debugger"</u>, install LinkServer and MCUXpresso Secure Provisioning Tool as well.

X MCUXpresso Installer v1.0	– 🗆 ×
X MCUXpresso Installer	± ❹ € :
Choose one or more categories from the list below:	Install
Software Kits	
MCUXpresso SDK Developer Installs the necessary tools for an MCUXpresso SDK developer	
Zephyr Developer Installs the necessary tools for a Zephyr developer	
Debug Probes Software	
LinkServer Installs NXP GDB-based debugging solution for LinkServer probes	
SEGGER J-Link Installs SEGGER J-Link Software	
PEmicro Installs PEmicro GDB Server	
Individual Components	
Arm GNU Toolchain Installs the Arm GNU Toolchain and additional NXP libraries built with	h it
CMSIS Toolbox Installs the CMSIS Toolbox Software	
No installs in progress [info] Successfully checked for updates. The application is up to date.	
GER J-Link software	

After all packages are installed, be sure that the J-Link probe is connected to the host PC. Then, check if the probe is also available in the MCUXpresso extension under the *DEBUG PROBES* view, as shown in <u>Figure 6</u>.

×	File	Edit	Selection	View	Go	Run	Terminal	Help		
ථ		MCUXPI	RESSO FOR VS	CODE						
	>	QUICKS	TART PANEL							
Q	>	INSTAL	LED REPOSIT	ORIES						
	>	PROJEC	TS							
وړ	\sim	DEBUG	PROBES						U	
6		/ Jink S	SEGGER							
		<mark>م</mark> ک	J-Link Com	pact PL	US, SN	I: 8510	06605			
æ>) LS נ	.inkServer Li	nkServe	r Softw	/are not	found			
₿		✓ 🖳 F	PEmicro PEn	iicro Sof	tware i	not four	nd			
Ē										
X										

Figure 6. J-Link probe available under DEBUG PROBES view

2.4 Import MCUXpresso SDK

Depending on what board you are running, build and download the specific SDK from NXP official website.

```
To build an example for i.MX 93 EVK, see Figure 7:
```

1	Hardware Off Middleware 0 Examples 0 Toolchains Off Parametrics 0 CLEAR	MCIMX93-EVK
		i.MX 93 Evaluation Kit
в	MCIMX93-EVK 1	App 🗹 🕅 Board 🗹
a		
A	мсімх93-еvk (мімх9352ххххм) 2	
⊨	MCIMX93-QSB (MIMX9322xxxxM)	
	MCIMX93AUTO-EVK (MIMX9352xxxxM)	
	MEK-MIMX8QM (MIMX8QM6xxxFF)	
	MEK-MIMX8QX (MIMX8QX6xxxFZ)	Found 180 111 591 882 HW solutions Boards Kits Processors
	MIMXRT1024-EVK (MIMXRT1024xxxxx)	
	MIMXRT1040-EVK (MIMXRT1042xxxxB)	+ ADD HW TO FILTERS
	MIMXRT1060-EVKB (MIMXRT1062xxxxB)	
	MIMXRT1060-EVKC (MIMXRT1062xxxxB)	Explore selection with
•	MIMXRT1160-EVK (MIMXRT1166xxxxx)	PINS TOOL
	MIMXRT1170-EVK (MIMXRT1176xxxxx)	
	MIMXRT1170-EVKB (MIMXRT1176xxxxx)	
	MIMXRT1180-EVK (MIMXRT1189xxxxx)	✓ 25.03.00
		BUILD SDK
	3	

To import the MCUXpresso SDK repository in VS Code, perform the following steps:

NXP Semiconductors

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

- 1. After downloading the SDK, open Visual Studio Code. Click the MCUXpresso tab from the left side, and expand the *INSTALLED REPOSITORIES* and *PROJECTS* views.
- 2. Click the *Import Repository* and select *LOCAL ARCHIVE*. Click the *Browse…* corresponding to the *Archive* field and select the recently downloaded SDK archive.
- 3. Select the path where the archive is unzipped and fill in the *Location* field.
- 4. The *Name* field can be left by default, or you can choose a custom name.
- 5. Check or uncheck Create Git repository based on your needs and then click Import.

MCUXPRESSO FOR VS CODE ····	E Import Repository ×
V QUICKSTART PANEL + Import Repository In Import Example from Repository	Import Repository
P ¹ ⁸ +8 Import Project	
	Archive: c\Users\SDK_25_03_00_MCIMX93-EVK.zip Browse
	Name: SDK_25_03_00_MCIMX93-EVK
In order to start developing your first MCOXpresso	Note: Path doesn't exist. Folder(s) will be created.
apprication, you should inst have a repository installed.	Location: c\Users\SDKs Browse Note: Path doesn't exist. Folder(s) will be created.
↓ PROJECTS	Create Git repository
Start developing your project.	Import 3
Import Example from Repository	
Import Project	

Figure 8. Import the MCUXpresso SDK repository

2.5 Import an example application

When the SDK is imported, it appears under the INSTALLED REPOSITORIES view.

To import an example application from the SDK repository, perform the following steps:

- 1. Click the Import Example from Repository button from the PROJECTS view.
- 2. Choose a repository from the drop-down list.
- 3. Choose the toolchain from the drop-down list.
- 4. Choose the target board.
- 5. Choose the demo apps/hello world example from the Choose a template list.
- 6. Choose a name for the project (the default can be used) and set the path to project Location.
- 7. Click Create.

After importing the example application successfully, it must be visible under the *PROJECTS* view. Also, the project source files are visible in the *Explorer* (Ctrl + Shift + E) tab.

3 Building the application

To build the application, press the left *Build Selected* icon, as shown in Figure 9.

NXP Semiconductors

AN14120

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

MCUXPRESSO FOR VS CODE QUICKSTART PANEL	mcimx93evk_hello_world_cm33 > C hello_world.c >
	<pre>25 26 /* 27 * Code 28 ************************************</pre>
PROJECTS McDUXpresso S D PC PC	<pre>38 39 PRINTF("hello world.\r\n"); 40 41 while (1) 42 { 43</pre>

Figure 9. Build application

4 Prepare the board for the debugger

To use the JTAG for debugging Cortex-M applications, there are a few prerequisites depending on the platform:

1. For i.MX 93

Debugging Cortex-M33 while only Cortex-M33 is running

In this mode, the boot mode switch **SW1301[3:0]** must be set to **[1010]**. Then the M33 image can be directly loaded and debugged using the debug button. For more details, see <u>Section 5 "Running and debugging"</u>.

If Linux running on Cortex-A55 is needed in parallel with Cortex-M33, there are two ways of debugging Cortex-M33:

• Debugging Cortex-M33 while Cortex-A55 is in U-Boot

First, copy the sdk20-app.bin file (located in the armgcc/debug directory) generated in <u>Section 3</u> "Building the application" into the boot partition of the SD card.

Boot the board and stop it in U-Boot. When the boot switch is configured to boot Cortex-A, the boot sequence does not start the Cortex-M. It has to be kicked off manually using the commands below. If Cortex-M is not started, JLink fails to connect to the core.

```
u-boot=> fatload mmc 1:1 80000000 sdk20-app.bin
u-boot=> cp.b 0x80000000 0x201e0000 0x10000
u-boot=> bootaux 0x1ffe0000 0
```

Note: If the system cannot be debugged normally, try to right-click the project in the MCUXpresso for VS Code and choose "Attach to debug the project".

Debugging Cortex-M33 while Cortex-A55 is in Linux

The kernel DTS must be modified to disable the UART5, which uses the same pins as the JTAG interface. If a Windows PC is used, the easiest is to install WSL + Ubuntu 22.04 LTS, and then to cross-compile the DTS.

After the WSL + Ubuntu 22.04 LTS installation, open the Ubuntu machine running on WSL and install the required packages:

```
$ sudo apt update
$ sudo apt install build-essential flex bison gcc-aarch64-linux-gnu git
```

Now, the kernel sources can be downloaded:

```
$ git clone https://github.com/nxp-imx/linux-imx
```

AN14120

```
$ cd linux-imx
$ git checkout lf-6.12.3-1.0.0
```

To disable the UART5 peripheral, search for lpuart5 node in the linux-imx/arch/arm64/boot/ dts/freescale/imx93-11x11-evk.dts file and replace the okay status with disabled:

```
&lpuart5 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_uart5>;
    status = "disabled";
    bluetooth {
        compatible = "nxp,88w8987-bt";
    };
};
```

Recompile the DTS:

```
$ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make freescale/imx93-11x11-
evk.dtb
```

Copy the newly created linux-imx/arch/arm64/boot/dts/freescale/imx93-11x11-evk.dtb file on the boot partition of the SD card.

Copy the hello_world.elf file (located in the armgcc/debug directory) generated in <u>Section 3</u> "Building the application" into the boot partition of the SD card.

Boot the board in Linux. Since boot ROM does not kick off the Cortex-M when Cortex-A boots, the Cortex-M must be manually started.

```
root@imx93evk:/lib/firmware# cp /run/media/mmcblk2p1/hello_world.elf /lib/
firmware
root@imx93evk:~# echo hello_world.elf > /sys/class/remoteproc/remoteproc0/
firmware
root@imx93evk:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

Note: The hello_world.elf file must be placed in the /lib/firmware directory.

2. For i.MX 95

• Debugging Cortex-M7 while only Cortex-M7 is running (no U-Boot or Linux running) In this scenario, build the bootloader using the mkimage tool, with the flash_m7 target, which does not include the U-Boot binaries. The M7 image inside the bootloader can be any image. It gets overwritten in the memory during debug.

\$ make SOC=iMX95 OEI=YES flash_m7

Write the bootloader on your boot device, then jump to Section 5 "Running and debugging".

- Debugging Cortex-M7 while Cortex-A is in U-Boot In this case, nothing special must be done. Boot the board in U-Boot and jump to <u>Section 5 "Running and</u> <u>debugging"</u>.
- Debugging Cortex-M7 while Cortex-A is in Linux
 - a. Switch the SW1 to OFF [0000] (JTAG side).
 - b. To disable the UART5, modify the kernel DTS. The UART5 uses the same pins as the JTAG interface. See the i.MX 93 Debugging Cortex-M33 while Cortex-A55 is in Linux section on how to disable the UART5 peripheral. Recompile and copy the resulted DTS on the boot partition of your boot device, then jump to Section 5 "Running and debugging".
- Debugging Cortex-M7 while in serial download This method uses the MCUXpresso Secure Provisioning Tool to generate a bootable image containing the OEI image, the M33 image (System Manager), and the M7 image. To use this mode, perform the following steps:
 - a. Set the boot mode switch **SW7[1:4]** to **[1001]**.

- b. Connect the MCU-link between the JTAG connector of the EVK and the PC. *Note: J-Link is not supported yet.*
- c. Connect an additional Type-C cable between the USB1 connector of the EVK and the PC.
- d. Specify the boot images. After the first build, a configuration file used by the MCUXpresso Secure Provisioning Tool is generated in .secureprovisioning/ additional_images_mx95_cm7_app_cfg.json. Edit the file with the correct images. First, get the necessary binaries.

- The DDR firmware (FW):

```
$ wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-
imx-8.27-5af0ceb.bin
$ chmod +x firmware-imx-8.27-5af0ceb.bin
$ ./firmware-imx-8.27-5af0ceb.bin
```

The DDR FW is located in the firmware-imx-8.27-5af0ceb/firmware/ddr/synopsys/ directory. Copy the lpddr5_imem_v202311.bin, lpddr5_imem_qb_v202311.bin, lpddr5_dmem_v202311.bin, and lpddr5_dmem_qb_v202311.bin in the .secure provisioning directory.

– The ELE FW:

```
$ wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-ele-
imx-2.0.1-0a66c34.bin
$ chmod+x firmware-ele-imx-2.0.1-0a66c34.bin
$ ./firmware-ele-imx-2.0.1-0a66c34.bin
```

The ELE FW is located in the firmware-ele-imx-2.0.1-0a66c34 directory. Copy the mx95a0ahab-container.img file in the .secureprovisioning directory.

- The OEI FW:

The OEI FW is not provided in binary form. This must be manually compiled.

\$ git clone <u>https://github.com/nxp-imx/imx-oei.git</u>

```
$ git checkout lf-6.12.3-1.0.0
```

\$ make board=mx95lp5 oei=ddr DEBUG=1

\$ make board=mx951p5 oei=tcm DEBUG=1

Copy the resulted binaries, build/mx95lp5/tcm/oei-m33-tcm.bin and build/mx95lp5/ ddr/oei-m33-ddr.bin, in the .secureprovisioning directory.

– The SM FW:

The System Manager is not provided in binary form. This must be manually compiled.

```
$ git clone https://github.com/nxp-imx/imx-sm.git
$ git checkout lf-6.12.3-1.0.0
$ make configence.
```

```
$ make config=mx95evk cfg
```

```
$ make config=mx95evk all
```

Copy the resulted binary build/mx95evk/m33_image.bin to the .secureprovisioning directory.

Note: Do not change/remove the cortex_m7_app and v2x_dummy entries.

```
{
  "cli_args": {
    "--additional-images": {
        "images": [
            {
            "entry_type": "oei_ddr",
            "container_set": "#1",
            "container_set": "#1",
            "extra_settings": {
               "lpddr_imem_path": "lpddr5_imem_v202311.bin",
               "lpddr_imem_qb_path": "lpddr5_imem_qb_v202311.bin",
```

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

```
"lpddr dmem path": "lpddr5 dmem v202311.bin",
          "lpddr_dmem_qb_path": "lpddr5_dmem_qb_v202311.bin",
          "oei ddr path": "oei-m33-ddr.bin"
        }
      },
        "entry type": "oei tcm",
        "container set": "#1",
        "extra settings": {
          "oei tcm path": "oei-m33-tcm.bin"
      },
        "entry type": "system manager",
        "container set": "#1",
        "extra settings": {
          "system manager path": "m33 image.bin"
        }
      },
      {
        "entry type": "cortex m7 app",
        "container set": "#1",
        "extra settings": {
          "cortex m7 app path": "dummyImage.bin"
      },
        "entry type": "v2x dummy",
        "container set": "#1",
        "extra settings": {
          "v2x dummy": "true"
      }
    ]
  "--ele-firmware": "mx95a0-ahab-container.img"
}
```

Build the project again and jump to Section 5 "Running and debugging".

3. For i.MX 8M

}

 Debugging Cortex-M while Cortex-A is in U-Boot In this case, nothing special must be done. Boot the board in U-Boot and jump to <u>Section 5 "Running and</u> debugging".

```
• Debugging Cortex-M while Cortex-A is in Linux
To run and debug the Cortex-M application in parallel with Linux running on Cortex-A, the specific clock
must be assigned and reserved for Cortex-M. It is done from within U-Boot.
Stop the board in U-Boot and run the below commands:
```

```
u-boot=> run prepare_mcore
u-boot=> boot
```

4. For i.MX 8ULP

Debugging Cortex-M while Cortex-A image needs to be loaded

For i.MX 8ULP, under normal single boot mode, it is needed to build the flash.bin using m33_image.bin in our "VSCode" repo first. The m33_image.bin can be found in {CURRENT REPO}\armgcc\debug\sdk20-app.bin. Refer to Section 6 from the Getting Started with

MCUXpresso SDK for EVK-MIMX8ULP and EVK9-MIMX8ULP in the SDK_2_xx_x_EVK-MIMX8ULP/ docs on how to build the flash.bin image.

Note: Use the $m33_image.bin$ in the active VSCode repo. Otherwise, the program does not attach properly.

Right-click and choose "Attach".

> ☆ Settings Build Project > ⊠ MCU Pristine Build/Rebuild Project > ⊕ Repository c\Users\nxtf75538\WorkSpace\U Open in Integrated Terminal > ⊕ Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard > № SEGGER 8.12a Project > № SEGGER 8.12a Project > № Pemicro Not installed Open with MCUXpresso Config Tools	> ± Settings > ☑ MCU > ☑ Build Configurations > ④ Repository cAUsers\nud75538\WorkSpaceVI > ☑ Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection > ☑ SEGGER 8.12a Project 2.11221 > ☑ Permicro Not installed Open with MCUXpresso Config Tools	✓ evkmimx8ulp_hello_world_cm33 MCUXpresso ^{cr}	NK 95 93 99	💩 Þ 🗘 😫
 Build Configurations Repository c\Users\nxt75538\WorkSpace\U Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a SEGGER 8.12a Project S LinkServer 24.12.21 Configure Open with MCUXpresso Config Tools 	> ⊞ Build Configurations Clean Project > ⊕ Repository c\Users\nxt75538\WorkSpace\I Open in Integrated Terminal > ⊡ Project Files Open in Integrated Terminal Debug Attach Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard > > ➡ SEGGER 8.12a Project > ➡ Pemicro Not installed Open with MCUXpresso Config Tools			
 Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a SEGGER 8.12a Project SE LinkServer 24.12.21 Configure Open with MCUXpresso Config Tools 	 Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard SEGGER 8.12a Project -> SEGGER 8.12a Project -> Configure Open with MCUXpresso Config Tools 	> 뛢 MCU	Pristine Build/Rebuild Project	
> ③ Repository c\Users\ndt75538\WorkSpace\I Open in Integrated Terminal > ⊡ Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard > ⊡ SEGGER 8.12a Project > ⊡ SEGGER 8.12a Project > ⊡ PEmicro Not installed Open with MCUXpresso Config Tools	 Se Repository c\Users\nxt75538\WorkSpace\T Project Files Open in Integrated Terminal Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project 24.12.21 Configure Open with MCUXpresso Config Tools Device TREE IMAGE INFO 	> 🗄 Build Configurations	Clean Project	
Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project S LinkServer 24.12.21 Configure Pemicro Not installed Open with MCUXpresso Config Tools	Debug Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project SEGGER 8.12a Project SEGGER 8.12a Project Project Project Project Project Project Project Configure Open with MCUXpresso Config Tools	> @ Repository c:\Users\nxf75538\WorkSpace\II		
Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project SEGGER 8.12a Project Project Permicro Not installed Open with MCUXpresso Config Tools	Attach Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project SEGGER 8.12a Project Pemicro Not installed Open with MCUXpresso Config Tools	> 🖻 Project Files	Open in Integrated Terminal	
Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard Copy to Clipboard SEGGER 8.12a Project SEGGER 8.12a Configure Configure Open with MCUXpresso Config Tools	Flash the Selected Target Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard Copy to Clipboard ElinkServer 24.12.21 Configure Pemicro Not installed Open with MCUXpresso Config Tools		Debug	
Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard Project SEGGER 8.12a Project SEGGER 8.12a Configure Configure Open with MCUXpresso Config Tools	Erase Flash Reset Probe Selection Copy to Clipboard Copy to Clipboard Project Is LinkServer 24.12.21 Configure Pemicro Not installed Open with MCUXpresso Config Tools		Attach	
Reset Probe Selection Copy to Clipboard DEBUG PROBES Segger 8.12a Project LinkServer 24.12.21 Configure Pemicro Not installed Open with MCUXpresso Config Tools	Reset Probe Selection Copy to Clipboard DEBUG PROBES Image INFO Reset Probe Selection Copy to Clipboard Project Project Project <td></td> <td>Flash the Selected Target</td> <td></td>		Flash the Selected Target	
Copy to Clipboard > > DEBUG PROBES Project > > IS LinkServer 24.12.21 Configure > > IMAGE INFO Open with MCUXpresso Config Tools >	Copy to Clipboard > > Image InFO Copy to Clipboard > Copy to Clipboard > > > Image InFO Configure >		Erase Flash	
> DEBUG PROBES Project > > IS SEGGER 8.12a Project > > IS LinkServer 24.12.21 Configure > > IMAGE INFO Open with MCUXpresso Config Tools >	> DEBUG PROBES Project > > Is LinkServer 24.12.21 Configure > > Image Performance Open with MCUXpresso Config Tools Open with MCUXpresso Config Tools		Reset Probe Selection	
> DEBUG PROBES Project > > IS SEGGER 8.12a Project > > IS LinkServer 24.12.21 Configure > > IMAGE INFO Open with MCUXpresso Config Tools >	> DEBUG PROBES Project > > Is LinkServer 24.12.21 Configure > > Image Performance Open with MCUXpresso Config Tools Open with MCUXpresso Config Tools		**	
> SEGGER 8.12a Project > Ø ③ > IS LinkServer 24.12.21 Configure > > IMAGE INFO Open with MCUXpresso Config Tools Ø ④	> IS SEGGER 8.12a Project > \$ () \$		Copy to Clipboard	
S LinkServer 24.12.21 Configure > Open with MCUXpresso Config Tools DEVICE TREE IMAGE INFO	LinkServer 24.12.21 Configure PEmicro Not installed Open with MCUXpresso Config Tools DeVICE TREE IMAGE INFO		Project	§ (?)
PEmicro Not installed Open with MCUXpresso Config Tools Device TREE IMAGE INFO	PEmicro Not installed Open with MCUXpresso Config Tools Device TREE IMAGE INFO			
> DEVICE TREE ~ IMAGE INFO	> DEVICE TREE ~ IMAGE INFO	> 🔛 PEmicro Not installed	Configure	
✓ IMAGE INFO	✓ IMAGE INFO		Open with MCUXpresso Config Tools	
✓ IMAGE INFO	✓ IMAGE INFO			
		> DEVICE TREE		
× No build artifact loaded	× No build artifact loaded	✓ IMAGE INFO		
		× No build artifact loaded		

Figure 10. Attach to J-Link debugger for i.MX 8ULP

• Debugging Cortex-M while Cortex-A image does not need to be loaded This method skips the loading and booting of the Cortex-A image. First, build, and write the flash.bin without the m33 image.bin, using the following command:

uuu -b emmc flash A.bin flash B.bin

Where,

- flash_A.bin is the flash.bin that contains the m33_image.bin. Build it using the following command:

make SOC=iMX8ULP flash_singleboot_m33 REV=A2

- flash_B.bin is the flash.bin that does not contain the m33_image.bin. Build it using the following command:

make SOC=iMX8ULP flash_singleboot REV=A2

Then, jump to Section 5 "Running and debugging".

Running and debugging 5

After pressing the debug button, choose the Debug project configuration and the debugging session starts.

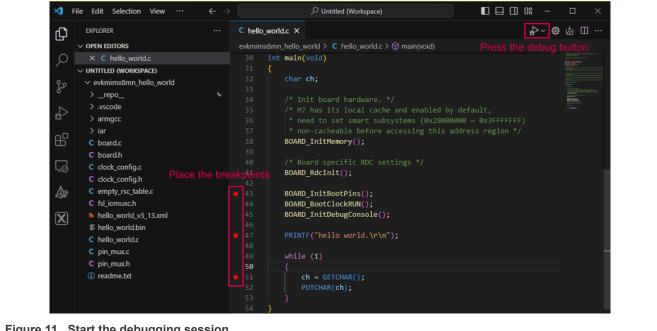


Figure 11. Start the debugging session

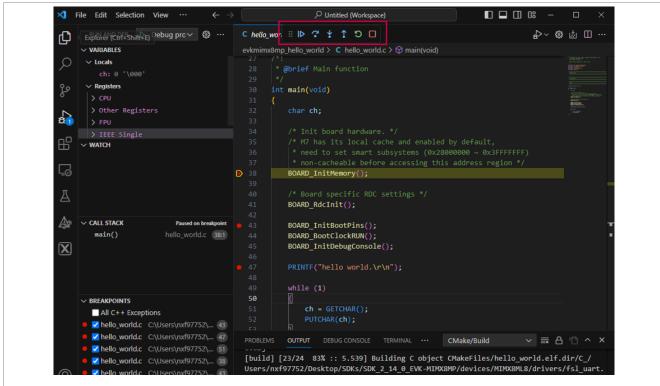


Figure 12. Debugging menu

When a debugging session starts, a dedicated menu is displayed. The debugging menu has buttons for starting the execution until a breakpoint fires up, pause the execution, step over, step into, step out, restart, and stop.

Also, we can see local variables, register values, watch some expression, and check call stack and breakpoints in the left-hand navigator. These function regions are under the "Run and Debug" tab, and not in MCUXpresso for VS Code.

RUN AND DEB 🕨 Debug pro 🗸 😳 …	C fsLdebug_console.c × □ □					
✓ VARIABLES	C:>Users > nxf97752 > Desktop > SDKs > SDK_2_14_0_EVK-MIMX8MP > devices > MIMX8ML8 > utilities > debug_console > C fsl_debug_console. > G fsl_debug_console. > G blagConsole_Vprintf(const char *, va_list)					
○ v Locals						
logLength: 0						
<pre> result: -1891565567</pre>						
fort of Gy2d9c "hello woold \n\n"	1044 #1f (defined(SDK_DEBUGCONSOLE) & (SDK_DEBUGCONSOLE == DEBUGCONSOLE_REDIRECT_TO_SDK))					
> formatStringArg: {}	144 /* See 51 debug console.h for documentation of this function. */					
	1847 int DbgConsole Printf(const char *fmt_s,)					
CPU						
	1049 va_list ap;					
> Other Registers						
G > FPU						
> IEEE Single	<pre>1052 va_start(ap, fmt_s);</pre>					
▲ > IEEE Double	<pre>1053 result = DbgConsole_Vprintf(fmt_s, ap);</pre>					
	1054 va_end(ap); 1055 -					
A	1055 return result;					
X						
V WATCH	<pre>1060 int DbgConsole_Vprintf(const char *fmt_s, va_list formatStringArg)</pre>					
	10102 int logLength = 0, result = 0;					
	<pre>1063 char printBuf[DEBUG_CONSOLE_PRINTF_MAX_LOG_LEN] = {'\0'};</pre>					
	1064 1065 if (NULL != g.serialHandle)					
	1005 11 (nucl := g_seriamanuc) 1866 {					
	1067 t /* format print log first */					
	<pre>1068 logLength = StrFormatPrintf(fmt_s, formatStringArg, printBuf, DbgConsole_PrintCallback);</pre>					
	<pre>1070 result = DbgConsole_SendDataReliable((uint8_t *)print8uf, (size_t)logLength);</pre>					
✓ CALL STACK Paused on step	1072 return result;					
DbgConsole_Vprintf(const char * fm						
DbgConsole_Printf(const char * fmt						
main() hello_world.c (47:1)	1076 int DbgConsole_Putchar(int ch)					
	PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR					
	0'}.{number="13",value="0x2001ffd0"}.{number="14",value="0x0a5"}.{number="15",value="0x3978"}.{number="25",value="0x1000000"}.{number="91",value="0x2001ffd0"}.					
	lue="0x0"},{number="94",value="0x0"},{number="95",value="0x0"},{number="96",value="0x0"},{number="97",value="0x0"},{number="98",value="0x1000000"},{n					
	{number="101",value="0x10000000"},{number="102",value="0x10000000"},{number="103",value="0x0"},{number="104",value="0x0"},{number="105",value="0x0"},{number="105",value="0x0"},					

Figure 13. Variables, registers, and call stack

6 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
- 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR 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.

7 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14120 v.3.0	5 May 2025	Updated the following sections for i.MX 95:
		<u>Section 1 "Introduction"</u>
		<u>Section 1.1 "Software environment"</u>
		<u>Section 1.2 "Hardware setup and equipment"</u>
		<u>Section 2.4 "Import MCUXpresso SDK"</u>
		<u>Section 2.5 "Import an example application"</u>
		<u>Section 4 "Prepare the board for the debugger"</u>
AN14120 v.2.0	13 March 2024	Updated <u>Section 2.4 "Import MCUXpresso SDK"</u> and <u>Section 2.5</u> " <u>Import an example application</u> " for MCUXSDK 2_15_000 version
AN14120 v.1.0	24 November 2023	Initial public release

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at <u>PSIRT@nxp.com</u>) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

 $\ensuremath{\mathsf{NXP}}\xspace$ B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners. **NXP** — wordmark and logo are trademarks of NXP B.V.

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. J-Link — is a trademark of SEGGER Microcontroller GmbH. Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Debugging Cortex-M with VS Code on i.MX 8M, i.MX 8ULP, and i.MX 9

Contents

1	Introduction	2
1.1	Software environment	2
1.2	Hardware setup and equipment	2
2	Prerequisites	2
2.1	PC Host – i.MX board debug connection	3
2.2	VS Code configuration	4
2.3	MCUXpresso extension configuration	5
2.4	Import MCUXpresso SDK	6
2.5	Import an example application	7
3	Building the application	7
4	Prepare the board for the debugger	8
5	Running and debugging	13
6	Note about the source code in the	
	document	14
7	Revision history	15
	Legal information	

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2025 NXP B.V.

All rights reserved.

For more information, please visit: https://www.nxp.com

Document feedback Date of release: 5 May 2025 Document identifier: AN14120